

Section 4: Analyzing the Workload Manager

This section discusses considerations for analyzing the Workload Manager.

- Chapter 1 presents an overview of the Workload Manager concepts.
- Chapter 2 describes subsystem transactions.
- Chapter 3 discusses internal logic of the Workload Manager.
- Chapter 4 describes the performance data that are used by CPExpert to analyze system performance from the perspective of the Workload Manager.
- Chapter 5 highlights some of the factors that must be considered when analyzing performance based upon workload data collected and recorded by RMF.

The purpose of this section is to present information that may not be generally available. The basic IBM documents dealing with the Workload Manager (the *Planning: Workload Management* document and the *Programming: Workload Management Services* document) describe the basic structure of the Workload Manager and user interaction with the Workload Manager. Some of the concepts are explained well in these documents, while other concepts have not been completely addressed.

IBM Workload Manager developers have given presentations at various professional conferences to amplify the Workload Manager concepts. This section (1) presents some of the discussion from the conferences, (2) describes concepts based on our analysis of SMF data from systems executing under Goal Mode, and (3) elaborates concepts based on private discussions with the IBM Workload Manager developers.

Not all concepts are completely understood, and this section will be improved with increased knowledge. More importantly, the results from Workload Manager parameter specification in different environments can be quite different from the results the Workload Manager developers envisioned. This particularly is true of subsystem (e.g., CICS) interaction with the Workload Manager.

As you use the WLM Component and gain additional experience with the Workload Manager, please call Computer Management Sciences with any information that might be different from what is presented in this section! For that matter, feel free to call with any questions or comments. **We appreciate feed-back.**

Chapter 1: Workload Manager Concepts

As mentioned in Section 1, the Workload Manager introduced with MVS/ESA SP5.1 is a radical departure from earlier system management concepts employed with MVS. The Workload Manager attempts to solve problems with earlier versions of MVS by removing the requirement that users provide detailed guidance to MVS on how MVS should process work. With the Workload Manager, users specify **performance goals**. The Workload Manager interprets the performance goals, and the Workload Manager tells MVS how to process work to meet the performance goals.

Some of the interaction between users and MVS is similar to earlier systems.

- Users still categorize work using workload categorization schemes similar to the schemes embodied in the IEAICSxx definitions.
- The workload is assigned to *service classes*, much as workload was assigned to performance groups in earlier versions of MVS.

The major difference between earlier versions of MVS and the Workload Manager is that users do not provide the Workload Manager with detailed specifications on how to process the work assigned to a service class. Users describe **performance goals**, and the Workload Manager adjusts system resources to meet the performance goals.

Chapter 1.1: Service Definition

Users define workload categories, establish classification rules for the workloads, assign the workloads to service classes, define performance goals and goal importance for the service classes, and establish resource boundaries for the service classes by constructing a *service definition*. Users construct the service definition using the Workload Manager ISPF application. A service definition is simply a collection of the output from the Workload Manager ISPF application. There is one service definition for an entire sysplex. The service definition consists of *service policies*, *workload classification schemes* for the workload, *service classes*, and *resource groups*. Each of these components of a service definition will be described in the following chapters.

Chapter 1.2: Service Policies

Users communicate their service requirements for workloads by defining a *Service Policy* to the Workload Manager. A service policy is simply a "named" collection of service classes and their associated performance goals and (optionally) processing boundaries for the address spaces associated with the service classes. Each of these parts of the service policy will be described in the following chapters.

A policy applies to a sysplex. Once a policy is invoked (using an operator command), the definitions contained in the policy remain in effect for the duration of the IPL. Although part or all of the policy may be overridden by a different policy invoked by an operator, any definition not overridden will remain in effect.

Chapter 1.3: Workload Classification

In earlier versions of MVS, installations classified work using the IEAICSxx member of SYS1.PARMLIB. After applying classification rules in IEAICSxx, the resulting work would be assigned to performance groups.

With the Workload Manager, the same process conceptually applies. However, instead of assigning work to performance groups, work is classified after applying classification rules and the workload is assigned to a service class¹.

Work is classified based on "qualifiers" associated with the work². Exhibit 4-1 shows the work qualifiers available with each version of MVS.

- Accounting information (AI): accounting information provided with the job or transaction.
- Correlation information (CI): correlation information is the DB2 correlation ID of the DDF server thread.
- Collection name (CN): collection name is the DB2 collection name of the first SQL package accessed by the distributed relational database architecture (DRDA) requestor in the work request.
- Connection type (CT): connection type is the DB2 connection type of the distributed data facility (DDF) server thread. The thread contains the value "DIST" indicating it is a server.
- LU Name (LU): the logical unit (LU) name originating the work.
- Netid (NET): the network identification originating the work.
- Process Name (PC): the MQWIH_ServiceName from the message's work information header.

¹The workload may be assigned to a report class instead of being assigned to a service class or in addition to being assigned to a service class. Report classes are described in Chapter 1.5.

²See IBM's *Planning: Workload Management* (Defining Classification Rules) for a complete discussion of the workload classification rules.

- Perform (PF): performance group number specified via PERFORM keyword on the JCL JOB statement or the START command, of the performance group number specified on the TSO logon panel.
- Package name (PK): package name is the name of the first DB2 package accessed by the DRDA requestor in the work request.
- Plan name (PN): plan name is the DB2 plan name associated with the DB2 server thread.
- Procedure Name (PR): the procedure name associated with the originator of the query.
- Priority (PRI): the JES priority associated with the batch job submitted through JES2.
- Sysplex name (PX): the sysplex name of the specific sysplex in which the work is running.
- Scheduling environment (SE): for JES, this is the scheduling environment name assigned to the job; for DB2, this is the scheduling environment name associated with the originator of the query.
- Subsystem instance (SI): the subsystem which generated the work. IBM-supplied subsystems consist of ASCH (APPC transaction programs), CICS (transactions processed by CICS), DDF (work requests from DB2 distributed data facility), IMS (messages processed by IMS), and JES (either JES2 or JES3).
- Subsystem specific parameter (SPM): a parameter that a subsystem can define and use to further classify work.
- Subsystem collection name (SSC): for JES, this is the XCF group name; for DB2, this is the subsystem collection name associated with the originator of the query.
- System name (SY): the system name for those address spaces whose execution system is known at classification time. This applies to ASCH, OMVS, STC, SYSH³, and TSO. JES is not eligible for this qualifier, as the system on which classification occurs might not be the system on which the job is run. Additionally, subsystem-defined transactions (CICS/IMS) and enclave-based transactions are not bound to an execution system at classification time, and are not eligible for the system name qualifier.
- Transaction class/job class (TC): the transaction class or job class.
- Transaction name/job name (TN): the transaction name or job name.

³The SYSH address space was introduced with z/OS V1R2 to describe logical partitions running LINUX or other non z/OS systems. CPU resources can be managed across these logical partitions in accordance with workload goals.

- Userid (UI): the user identification associated with the job, transaction, message, etc.

Unlike the classification scheme used with MVS prior to SP5.1 (Goal Mode), there is no set order of the classification. Users of the Workload Manager define the order in which the work qualifiers are to be applied to work arriving in the system.

MVS assigns defaults if work coming into the system is not classified by the classification scheme defined by users. The defaults are based on the type of work:

- SYSTEM (consisting of system address spaces): MASTER, GRS, DUMPSRV, SMF, CATALOG, RASP, XCFAS, SMXC, CONSOLE, IOSAS, and all other system address spaces designated "high dispatching priority" by MVS (that is, those with a dispatching priority of X'FF').
- SYSSTC (consisting of started tasks not classified by the workload classification scheme).
- SYSOTHER (for all other work not classified by the workload classification scheme).

		OS/390									
QUALIFIER	SP5	V1R1	V1R2	V1R3	V2R4	V2R5	V2R6	V2R7	V2R8	V2R9	V2R10
AI	X	X	X	X	X	X	X	X	X	X	X
CI		X	X	X	X	X	X	X	X	X	X
CN		X	X	X	X	X	X	X	X	X	X
CT		X	X	X	X	X	X	X	X	X	X
LU	X	X	X	X	X	X	X	X	X	X	X
NET	X	X	X	X	X	X	X	X	X	X	X
PC							X	X	X	X	X
PF				X	X	X	X	X	X	X	X
PK		X	X	X	X	X	X	X	X	X	X
PN		X	X	X	X	X	X	X	X	X	X
PR				X	X	X	X	X	X	X	X
PRI					X	X	X	X	X	X	X
PX											X
SE											X
SI	X	X	X	X	X	X	X	X	X	X	X
SPM				X	X	X	X	X	X	X	X
SSC											X
SY											X
TC	X	X	X	X	X	X	X	X	X	X	X
TN	X	X	X	X	X	X	X	X	X	X	X
UI	X	X	X	X	X	X	X	X		X	X

**WORKLOAD CLASSIFICATION QUALIFIERS AVAILABLE
WITH EACH VERSION OF MVS**

EXHIBIT 4-1

Chapter 1.4: Service Classes

Users assign workload to a service class and specify the performance goal and goal importance of the service class. Optionally, users also may (1) define multiple performance periods for the service class and (2) assign the service class to a resource group.

A service class can represent any collection of workload that can be classified using the workload classification schemes available with the Workload Manager.

Exhibit 4-2 shows example service classes and their associated workloads.

SERVICE CLASS	WORKLOAD DESCRIPTION
TSOPROD	TSO transactions submitted by production organizations
TSOTEST	TSO transactions submitted by test and development organizations
BATCH	Regular batch jobs
LOWPRI	Low priority batch jobs
CICSAOR1	CICS Application Owning Region 1
CICSPERS	CICS transactions from Personnel Dept
CICSADMN	CICS transactions from Administration Dept
CICSOPS	CICS transactions from Operations Dept
ST_TASK	Started tasks

EXAMPLE SERVICE CLASSES

EXHIBIT 4-2

The example shows that service classes may be similar to performance groups or domains that were defined with earlier versions of MVS.

- Service classes are much like performance groups, in that they have workloads assigned to them. As will be clear from later discussion, address spaces associated with a service class may migrate to lower service class periods based upon the duration

of service used by the address space. Since the Workload Manager manages at the *service class period* level, subsequent discussion generally will refer to service class *periods* rather than referring to service classes.

- Service class periods also are like domains from the view of the SRM in that SRM controls in such areas as the target MPL or expanded storage controls may be applied to service class periods. In fact, the Workload Manager creates a domain (for the SRM's internal control tables) for each service class period defined in a service policy.

There are some significant differences between (1) performance group periods and domains, and (2) service class periods:

- With performance group periods, users can specify detailed controls (e.g., dispatching priority, timeslicing, storage isolation, response time option, etc.). With domains, users similarly can specify detailed controls (i.e., minimum and maximum multiprogramming levels, service slopes, and expanded storage criteria).
- With service class periods, users can specify performance goals and importance of the goal. Additionally, users can assign the service class period to a resource group (described later) and specify resource minimum and maximum for the resource group.
- If CICS/ESA Version 4.1 or IMS Version 5 are installed, users can specify service classes *at the transaction level* for transactions that are served by a CICS region or IMS region. These subsystem transactions can be grouped into service classes, and the Workload Manager will attempt to achieve transaction response based on the performance goals specified for the transaction service classes.

Exhibit 4-2 shows three transaction service classes (CICSPERS, CICSOPS, and CICSADMN). These transaction service classes are considered to be "served" transaction service classes, and they are served by the CICSAOR1 "server" service class in the example.

There are several significant restrictions with assigning transactions to service classes:

- The service class describing the transactions must have a response performance goal and importance assigned; however, the service class cannot have multiple performance periods. This is because the SRM does not collect resource use at the transaction level. Rather, all resource use is accounted for at the subsystem level (i.e., at the CICS/ESA Version 4.1 region or IMS Version 5 region level).

Since resource information is not collected at the transaction level, "service units" cannot be attributed to transactions. Consequently, there is no way for the SRM to determine how much "service" the transactions have accumulated. Since the SRM cannot determine how much service the transactions have accumulated, the Workload Manager cannot migrate transactions to lower performance periods.

Further, CICS or IMS actually handle dispatching of the transactions within the CICS or IMS region. Consequently, the Workload Manager could not take action at the transaction level even if it did determine how much service the transactions accumulated. The Workload Manager can manage resources only at the CICS or IMS region level, as these are the address spaces actually using resources.

- The subsystem service class (e.g., the CICS/ESA Version 4.1 region or IMS Version 5 region) must have a performance goal and importance defined, in order for the region to start up. However, the performance goal and importance normally are used by the Workload Manager **only at start-up time** for the address space⁴.

After start-up time, the Workload Manager normally ignores the goal and importance of subsystems. After start-up time, the Workload Manager normally uses the goal and importance of the "served" transaction service classes as the basis for its resource allocation decisions.

The Workload Manager attempts to meet the performance goals of the "served" transaction service classes. In order to meet these performance goals, the Workload Manager must assign resources to the server service class (e.g., the service class of the CICS region), regardless of the goal and importance assigned to the subsystem service class.

Chapter 1.4.1: Performance goals

Users specify a performance goal for each service class. There are four types of performance goals: average response, percentile response, execution velocity, and discretionary. This chapter presents an overview of these goals, and refers the reader to other sections of this document (or to rule descriptions) for a more comprehensive discussion.

- **Average response goal.** For service classes that have an average response time goal, the Workload Manager determines whether the average response time achieved by transactions ending in the service class is greater than the performance goal. If the average response time is greater than the performance goal, the system is not meeting performance goals for the service class period. If the importance of the service class is sufficiently high, the Workload Manager may re-allocate system resources in an attempt to meet performance goals.

Please refer to Rule WLM101 for a more comprehensive discussion of average response time goals.

⁴This statement is not true if the region should become idle for some period of time. If there are no transactions executing in the region for some time, the Workload Manager will rely on the performance goal and importance associated with the region to make resource allocation decisions. This situation should normally occur only during "off shifts" or for test regions with low activity.

- **Percentile response goal.** Service classes can be defined that have a "percentile" response performance goal. A "percentile" response performance goal means that the performance goal is defined as "x%" of the transactions should complete within "y" time. For example, a percentile response goal could be that **90% of the transactions should complete within 200 milliseconds**.

For service classes that have a percentile response time goal, the Workload Manager determines whether the requisite percent of transactions are ending within the response goal. If the requisite percent of transactions is not ending within the response goal, the system is not meeting performance goals for the service class period. If the importance of the service class is sufficiently high, the Workload Manager may re-allocate system resources in an attempt to meet the performance goal.

Please refer to Rule WLM102 for a more comprehensive discussion of response percentile goals.

- **Execution velocity goal.** Installations may specify an *execution velocity goal* for a service class. An execution velocity goal is a measure of how fast work should run when the work is ready to run, without being delayed waiting for WLM-managed resources. Delays for WLM-managed resources include delays waiting for access to a CPU, delayed waiting for access to processor storage⁵, or delayed waiting for server address space creation.

With OS/390 Release 3, execution velocity can optionally include delays waiting for non-paging DASD I/O operations. Non-paging DASD I/O delays include IOS queue delays, subchannel pending delays, and control unit queue delays.

With OS/390 Version 2 Release 4, execution velocity can optionally include delays waiting for an initiator (with batch jobs in WLM-managed job classes).

The purpose of specifying an execution velocity is to allow installations to assign an overall processing importance to the work being processed, when the work has no time-related measure (that is, a response requirement is not associated with the work).

Please refer to Rule WLM103 for a more comprehensive discussion of execution velocity goals.

- **Discretionary goal.** A discretionary goal is specified for service classes that do not have a response goal or execution velocity goal⁶. A discretionary goal for a service class simply tells the Workload Manager to process the corresponding workload on a

⁵Processor storage is composed of *central storage* and *expanded storage*. The third category of storage is *auxiliary storage*.

⁶Any unit of work not assigned to a service class is assigned to the SYSOTHER default service class. A unit of work might not be assigned to a service class because the work was not identified using the workload classification scheme. The SYSOTHER default service class has a discretionary performance goal.

"discretionary" basis. The Workload Manager may allocate resources to discretionary work to optimize resource usage. However, the Workload Manager normally will not allocate resources to discretionary work if allocating the resources would cause any service class to miss its performance goal.

IBM introduced *discretionary goal management* algorithms with OS/390 Version 2 Release 6. With discretionary goal management, service class periods that are **overachieving** their goals may have their CPU resources "capped" in order to allow some CPU resources to be used by service class periods with discretionary goals.

See a more complete discussion of discretionary goal management in Chapter 1.7 of this section.

Chapter 1.4.2: Performance goal importance

Users specify how important it is that the performance goal be met: very important to not important. Five levels of importance can be specified: *Importance 1* (very high) to *Importance 5* (not important). These importance values (1 to 5) may be specified for service classes that have a response goal or an execution velocity goal.

The Workload Manager internally defines **Importance 0** to represent system tasks (the most important work) and **Importance 6** to represent discretionary work (the least important work)⁷.

Chapter 1.4.3: Service class periods

A service class may be broken into multiple *performance periods*. Each service class has Period 1 automatically defined. Optionally, installations can define up to seven additional performance periods (although typically no more than two or three additional performance periods are defined). Each performance period has its own performance goal, goal importance, and duration (except that the duration may not be specified for the last performance period).

An address space (TSO transaction, batch job step, etc.) begins execution in Performance Period 1. The address space will transition from Performance Period 1 to Performance Period 2 (and to subsequent periods), based upon the accumulation of "service" by the transaction.

⁷The goal importance for discretionary work actually is recorded as **Importance 0** in SMF Type 72 records because Workload Manager developers felt that there was no value in reporting Importance 6 externally. From an external view, the lowest importance (Importance 6) is implicit in the fact that the work is discretionary.

Except for enclaves, the "service" required by the address space is a combination of CPU resources, I/O resources, and memory resources. The actual resources used by the address space are adjusted by the CPU, SRB, IOC, and MSO service coefficients defined in the Workload Manager ISPF application panel, to yield the CPU, I/O, and memory "service" requirements of the address space.

When the address space accumulates more than a specified amount of service, the SRM will reassign the address space to the next lower performance period (if multiple periods are defined)⁸. The amount of service controlling when an address space is reassigned to the next lower performance period is specified by the value of the **Duration** specification in the ISPF definition of the service class performance period.

The normal purpose of defining multiple performance periods is to give higher priority to interactive transactions, short batch job steps, etc. Overall response is decreased (and overall throughput is increased) when address spaces requiring relatively few resources are processed at a higher priority than those address spaces requiring substantial resources.

Multiple performance periods may not be used with a service class representing subsystem transactions (i.e., a service class defined for transactions executing under CICS/ESA Version 4.1 or under IMS/ESA Version 5). This is because system service units are not accumulated by the SRM for the transactions; the service units are accumulated at the **address space level** (i.e., the CICS region or IMS message processing region). Since service is not collected at the transaction level, the SRM cannot migrate service classes to lower periods for service classes representing subsystem transactions.

Chapter 1.5: Report Classes

Work units may be assigned to a report class via the Workload Classification ISPF panels.

Report classes are similar to report performance groups in earlier versions of MVS, but there are some differences.

- The report classes are similar to report performance groups in that (1) the SRM simply accumulates performance-related and resource-related statistics about the workload, (2) the report classes are not used to control workload, and (3) more than one workload can be assigned to a report class.

⁸Note that the address space is not reassigned immediately upon accumulation of the specified amount of service. The reassignment is performed only when the SRM evaluates the address space for changed conditions. Therefore, the SRM cannot determine when the **exact** amount of service specified in the DUR parameter is used; the SRM can only determine when the DUR value is exceeded.

The SRM checks for transactions exceeding their DUR value only when the SRM gains control. Each address space normally is evaluated every three SRM seconds, and reassignment is performed only if the address space has accumulated more than the value specified for the DUR keyword. Consequently, an address space will always accumulate **more** service than specified by the DUR keyword before being reassigned.

- The report classes are different from report performance groups in that a transaction or job can be assigned to only one report class. With versions of MVS prior to MVS/ESA SP5, a transaction or job could be assigned to as many as four⁹ report performance groups, depending on the workload classification scheme contained in IEAICSxx member of SYS1.PARMLIB.

Additionally, while report classes may be defined to report on subsystem transactions (e.g., CICS transactions), the report class information will not contain **resource** data for the subsystem transactions. This is because the subsystem (e.g., CICS) does not report transaction resource use to the Workload Manager and thus no resource use information is available for the report class.

Prior to z/OS V1R2, report class information did not include response time distributions and did not include work manager (e.g., CICS or IMS) delay states. This response and delay state information is provided with z/OS V1R2.

Beginning with z/OS V1R2, the overall structure of report classes has been significantly changed. With z/OS V1R2, report classes (1) can have as many **periods** as are reflected in the service class(es) to which work units associated with report classes are assigned, (2) can be used to report on a subset of transactions running in a **single** service, and (3) can combine work units running in **multiple** service classes periods within one report class period.

- **Report classes can have multiple periods.** Prior to z/OS V1R2, a report class had a single period. With z/OS V1R2, report classes can have as many periods as appear in the associated service class(es) to which work units related to the report class are assigned. For example, if a work unit is assigned to a service class and to a report class, and if the service class has 3 periods, then the report class can have three periods.
- **Report class period related to single service class.** A report class period that is related to a subset of work in a **single** service class, is referred to as a **homogeneous** report class period. This term means that the report class period contains resource and performance information about work units in a **single** service class period, and that the performance goal and goal importance related to the report class period are obtained from a **single** service class period. As described below, with z/OS V1R2, it is possible to identify homogeneous report class periods.
- **Report class period related to multiple service classes periods.** A report class can be associated with work units that are assigned to multiple service class periods. That is, the workload classification scheme can associate work units to **multiple** service classes, while associating these work units to a **single** report class. When a report class is associated to work units in multiple service classes, the report class is referred

⁹A transaction or job can be assigned to as many as five report performance groups beginning with MVS/ESA SP5 (Compatibility Mode).

to as a **heterogeneous** report class. This term means that the report class period contains resource and performance information about work units in **multiple** service class periods.

Heterogeneous report classes can cause incorrect or misleading performance data, since the data collected is based on different goals, importance, or duration as specified for the multiple service class periods. As described below, with z/OS V1R2, it is possible to identify heterogeneous report class periods.

The heterogeneous report class will have a number of periods corresponding to the largest number of periods of any associated service class. For example, if Service Class 1 has two periods and Service Class 2 has three periods, and if work units assigned to Service Class 1 and Service Class 2 were assigned to Report Class 1, then Report Class 1 would have three periods (based on Service Class 2 having three 10 periods).

With z/OS V1R2, SMF Type 72 (field R723CRS1: Report class period flags) contains information that describes whether a report class period is homogeneous or heterogeneous, on an RMF recording interval basis. A report class period is described as heterogeneous if more than one service class period was found contributing to the report class period in an RMF recording interval.

This new information means that it is possible to detect whether a report class period reverts between homogeneous and heterogeneous between RMF recording intervals, depending on whether work units actually executed in the corresponding service class periods during the RMF interval.

Additionally, with z/OS V1R2, SMF Type 72 variable R723CLSC contains the name of the service class that last contributed information to the report class period. The performance goal and goal importance described in SMF Type 72 records for the report class period are obtained from the **last** service class period that contributed to the report class period. This new field can be used to associate the performance information to a specific service class period, which could be particularly useful if a report class period reverted between homogeneous and heterogeneous states from one RMF recording interval to the next.

Chapter 1.6: Resource Groups

A resource group is simply a "named" description of the minimum and maximum unweighted CPU service units per second that may be used by one or more service classes. A resource group applies across an entire sysplex.

One or more service classes may be associated with a resource group¹⁰. The Workload Manager will attempt to provide the specified minimum CPU service to the resource group and will restrict the resource group from using more than the specified maximum CPU service. The minimum and maximum specifications apply on a sysplex-wide basis. For example, three resource groups could be defined as:

GROUP NAME	MINIMUM CPU	MAXIMUM CPU
LIMITCPU	0	500
TOP_PRI	100000	0
4381-91E	309	309

- One or more service classes could be associated with the LIMITCPU Resource Group, and the service classes normally would be limited to using a collective **maximum** of 500 CPU service units per second. The limit on access to CPU service is termed "resource capping" in IBM documents.
- In contrast, the TOP_PRI Resource Group specifies a **minimum** CPU service units, and the Workload Manager will attempt to provide the minimum CPU service specified. Service classes could be associated with the Resource Group TOP_PRI and the Workload Manager would attempt to provide a total of 100,000 CPU units per second to those service classes. The Workload Manager would assign a high dispatching priority to service classes associated with the TOP_PRI Resource Group in an attempt to guarantee that they had access to the CPU.

Practically, of course, it is not possible with current systems to provide 100,000 CPU service units per second. The specification simply means that the Workload Manager would try to provide service classes associated with the TOP_PRI Resource Group with access to the CPU whenever the service classes wanted access. The system on which the service classes were executing might actually be capable of providing a fraction of the specified minimum CPU service.

- The 4381-91E Resource Group defines both a **minimum** and a **maximum** CPU capacity. In this example (taken from IBM's *MVS/ESA SP5.1 Planning: Workload Management* document), the minimum and maximum specifications are the CPU service units provided by an IBM-4381-91E processor. With this specification, the Workload Manager will attempt to provide exactly the CPU service that would be

¹⁰A service class representing subsystem transactions (i.e., a service class defined for transactions executing under CICS/ESA Version 4.1 or under IMS/ESA Version 5) should not be associated with a resource group. This is because CPU resources are not monitored by the SRM for the transactions; the CPU resources are monitored at the **address space level** (i.e., the CICS region or IMS message processing region). Further, CPU dispatching occurs at the address space level, rather than at the transaction level. Since CPU usage is not collected at the transaction level and CPU dispatching is at the address space level, the Workload Manager cannot control the amount of CPU resources allocated to service classes that represent transactions. Thus, while the Workload Manager ISPF application does not prevent assignment of subsystem transactions to a service class assigned to a resource group, there is no effect in doing so; the Workload Manager ignores the resource group values with regard to the subsystem transaction service class.

provided if the service classes associated with the Resource Group were running on an IBM-4381-91E.

This resource group might be appropriate for a service bureau which has "sold" a 4381-91E equivalent to some customer.

If you should make such a specification to direct the Workload Manager to provide an equivalent amount of CPU time, you probably should not specify the "exact" values listed for the appropriate processor. Keep in mind that work executing on the "equivalent" processor would have experienced MVS overhead, wait on I/O, and other conditions. The resource group minimum and maximum relate to CPU time used by service classes, and do not include MVS overhead or other overhead.

The Workload Manager implements resource group boundaries by a cooperative interaction with the SRM.

- The Workload Manager attempts to provide the minimum CPU service units to the service classes associated with the resource group. The Workload Manager implements the CPU minimum boundary by assigning a high resource allocation priority to the service classes associated with a resource group which is below its specified minimum CPU.

The high resource allocation may result in a high CPU dispatching priority, a high central storage allocation priority (to minimize page faults), etc.

- The Workload Manager implements the CPU maximum boundary by setting bits in the SRM's control blocks to indicate that address spaces in the service class are "CPU capped" and are not eligible for dispatching¹¹.
- The Workload Manager periodically sums the CPU time used by each resource group (that is, the CPU time of all address spaces assigned to the service classes associated with the resource group). The Workload Manager implements CPU capping for the resource group if the CPU time used is greater than the maximum specified for the resource group.

¹¹Capslicing is a technique somewhat similar to "timeslicing" used in earlier versions of MVS. Timeslicing is a technique by which each second of real time (adjusted by a processor-dependent constant) is divided into 64 "slices". For the duration of a timeslice, an address space may have its dispatching priority at a relatively low priority or may have its dispatching priority raised to a high priority. This timeslicing facility has been an inherent part of the SRM design. Conceptually, timeslicing as implemented by having a vector of 64 bits associated with the address space, with each bit representing a timeslice. The first bit would represent the first 1/64 slice at the start of a second, while the last bit would represent the last 1/64 slice at the end of a second of real time. If any bit is "ON", the SRM raises the dispatching priority of the address space during the corresponding time slice. The dispatching priority reverts to the original level during the next slice when the corresponding bit is "OFF". Thus, the dispatching priority of an address space can be raised and lowered from a "base" priority to a "timeslice" priority.

- The Workload Manager manipulates a "cap slice" bit vector for the resource group, to prevent all address spaces in all service classes associated with the resource group from using the CPU during the "capped" timeslice interval.
- Each "CPU capped" status represents 1/64 second of real time (adjusted by a processor-dependent constant), and uses the timeslice concept. The difference is that rather than raising the dispatching priority during the time slice as was done in earlier versions of MVS, the SRM marks the address space non-dispatchable during the time slice to restrict the service class use of CPU service.
- It is important to appreciate that **all** address spaces in all service class periods in a resource group are controlled by the **same** number of cap slices. That is, when the resource group is in its "cap slice", all address spaces associated with the resource group are capped. This cap status has no effect if the address space is waiting on some event (waiting on I/O, etc.). However, any dispatchable unit (TCB or SRB) on the dispatch queue will not be dispatched by the MVS Dispatcher if the dispatchable unit has its "cap" bit turned on.

It normally is not advisable to use resource groups. IBM provides the facility solely for special cases, and IBM does not contemplate resource groups normally being used. Resource group specifications are "preemptive" in nature, in that the Workload Manager attempts to honor resource group specifications before considering other service specifications. Consequently, **resource group specifications could nullify the rest of the Workload Manager's algorithms.**

Chapter 1.7: Discretionary Goal Management

A problem existed when using discretionary goals prior to OS/390 Version 2 Release 6: on systems in which 100% of the CPU was used by service class periods with performance goals, service class periods assigned a discretionary goal might never receive CPU service. This situation existed even though the service class periods with performance goals might be significantly **overachieving** their goals, since the Workload Manager would never allow discretionary work to have a CPU dispatching priority equal to or higher than work with performance goals.

From one perspective, this algorithm is proper; discretionary work is defined as work that has no performance goal. However, most sites want the discretionary work eventually to be processed, even though it has no performance goal. Consequently, many sites removed the discretionary goal from work and assigned a performance goal to the work.

However, there are significant advantages to assigning a discretionary goal to work: work with a discretionary goal executes with the Mean-Time-To-Wait (MTTW) algorithm.

- C Work assigned to a Mean-Time-To-Wait group competes within the Mean-Time-To-Wait group for access to the processor. Address spaces are assigned dispatching priority within the MTTW group, based upon their execution characteristics. Address spaces that execute a significant amount of CPU instructions between I/O operations are considered heavy CPU users. These heavy users receive a lower dispatching priority within the MTTW group than do address spaces requiring less CPU processing between I/O operations.

- C The philosophy behind assigning work to Mean-Time-To-Wait groups is to attempt to use as much of the overall computer system as possible. Dispatching relatively light CPU users ahead of relatively heavy CPU users ensures that the I/O complex will be used simultaneously with the CPU processor. Since both CPU and I/O are active simultaneously, more overall work will be accomplished by the computer system. This philosophy assumes, of course, that overall throughput is a major goal, rather than the turnaround of specific heavy CPU users. This philosophy is explicitly applicable to service class periods assigned a discretionary goal.

IBM addressed this problem in OS/390 Version 2 Release 6, by implementing the *discretionary goal management* algorithms¹².

With discretionary goal management, the Workload Manager identifies service class periods that have been assigned a performance goal and that are candidates for participation in discretionary goal management. Service class periods can participate in discretionary goal management if either of the following conditions apply:

- C The service class period has a response goal greater than one minute. This condition does not apply to subsystem transaction service classes (e.g., CICS or IMS transaction service classes), since these service class periods do not include address spaces.

- C The service class period has an execution velocity goal less than or equal to 30%.

The Workload Manager identifies candidate service class periods meeting either of the above conditions, that have **significantly** overachieved their performance goal. If discretionary work exists in the system, the Workload Manager may apply *internal resource capping* to the service class periods that are overachieving their performance goal. The internal resource capping operates similarly to the normal Resource Group capping described in Chapter 1.6 of this section, in that the Workload Manager will cap the address spaces for one or more cap slices. This capping restricts the amount of CPU service that can be used by address spaces in the capped service class period.

¹²CPEExpert is proud to note that Don Deese described the basic discretionary goal management design to IBM SRM/WLM developers (Steve Grabarits and Peter Yukom) at a closed working session of the Enterprise Wide Capacity Management (EWCP) project of SHARE.

The Workload Manager may apply internal resource capping when the Performance Index¹³ is less than 0.7, and stops internal resource capping when the Performance Index is greater than or equal to 0.81. If a candidate service class period with a performance goal has multiple periods, later periods are selected for capping before earlier periods (that is, capping would potentially be applied to Period 2 before capping would be considered for Period 1).

The effect of the discretionary goal management algorithm is to allow discretionary work to receive CPU cycles when work with a performance goal would otherwise significantly overachieve its performance goal.

Chapter 1.8: Long-Term Storage Protection

With Compatibility Mode, organizations can specify “storage isolation” for critical work by using the PWSS keyword in the performance group period definition in the IEAIPsxx member of SYS1.PARMLIB. Once storage isolation is specified for a performance group period, the System Resources Manager will not normally steal processor storage from address spaces assigned to the performance group period, unless the processor storage used by the address space is more than the minimum storage amount specified in the PWSS keyword.

This ability to control the potential use of processor storage by work was not available with Goal Mode prior to OS/390 V2R10. Rather, the Workload Manager implicitly controlled the assignment of processor storage, based on how well different service class periods met their performance goal and based on the relative importance of the work. Consequently, some organizations have critical work that can suffer unacceptably under Goal Mode prior to OS/390 V2R10, because there was no ability for the organization to protect storage for the work. The problem typically arises when work goes “dormant” for some period of time.

For example, when there are no transactions being processed by a CICS region for some interval, the CICS region can be considered dormant. Since the CICS region is not processing transactions, the Workload Manager can decide to steal pages from the region. This action might be taken if necessary to provide storage to other work that might require processor storage to meet its performance goal. Unfortunately, when new transactions arrive, the CICS region might have lost much of its processor storage, and considerable paging might be required to restore the stolen pages. Retrieving the pages might considerably delay the new transactions. For critical work, this delay might be unacceptable.

With OS/390 V2R10, IBM introduced a “long-term storage protection” option. This option is available with APAR OW43810 installed. With the APAR applied, organizations can

¹³The Performance Index is described in Chapter 3.4 of this section. Note that 0.81 is the Performance Index arbitrarily assigned to work with a discretionary goal.

specify long-term storage protection on the WLM ISPF “Modify Rules for the Subsystem Type” panel. This is done by using the new “Storage Critical” option on this panel.

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

Storage protection can be assigned to all types of address spaces that are classified for ASCH, JES, OMVS, STC, and TSO. When you specify YES in the new "Storage Critical" field for a classification rule, you assign storage protection to all address spaces that match that classification rule.

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

- C The service class must have a single period.
- C The service class must have either a velocity goal, or a response time goal of over 20 seconds.

CICS and IMS work have another option that is available when you are creating transaction service classes (these are, service classes for which you specify transaction response goals). For transaction service classes, you can **also** assign long-term storage protection by specifying YES in the "Storage Critical" field in the rules for specific transactions.

It is important to appreciate that when you specify YES for **one** transaction in a CICS/IMS service class, **all** CICS/IMS transactions in that service class will be storage-protected.

If a CICS or IMS region is managed as a server by WLM¹⁴ and any of the transaction service classes it serves is assigned storage protection, then the CICS/IMS region itself is automatically storage-protected by WLM. This is because storage protection is actually implemented at the *address space* level, even though you specify storage protection for a transaction service class. The CICS or IMS *transaction* is not an address space but is processed by the CICS or IMS region, and the region is the address space. Thus, all transactions processed by the region would benefit from storage protection.

As an alternative to assigning storage protection based on specific transaction service classes, you can assign storage protection to the CICS or IMS region that processes the transactions. You do this by adding or modifying the STC or JES classification rule that assigns the service class to the region.

¹⁴This would be the case when the region is managed to the response time goals of the transactions it serves.

Please note the potential problem that occurs when you exclude a CICS or IMS region from being managed as a server by the Workload Manager. This option and the potential problem are discussed in Chapter 1.10 below.

Chapter 1.9 Long-Term CPU Protection

With Compatibility Mode, organizations can explicitly control CPU dispatching priority by assigning dispatching to performance group periods, using the DP keyword in the IEAIPsxx member of SYS1.PARMLIB. This ability to control explicitly the CPU dispatching priority by work was not available with Goal Mode prior to OS/390 V2R10. Rather, the Workload Manager explicitly controlled the assignment of CPU dispatching priority, based on how well different service class periods met their performance goal and based on the relative importance of the work¹⁵. This assignment of dispatching priority based on goal attainment can result in important work having a lower CPU dispatching priority than work that is less important.

For example, suppose that a particular service class is exceeding its service goal and another less-important service class is missing its performance goal. Further suppose that CPU delay is the reason that the less-important service class is missing its performance goal. After some analysis, the Workload Manager might decide to treat the high-importance service class as a “donor” of CPU time. The result might be a readjustment of CPU dispatching priorities, with the less-important service class being given a higher CPU dispatching priority than the important service class.

Perhaps with a dynamically changed workload (for example, a surge of transactions being submitted to the high-importance service class), the CPU demands of the high-importance service class might increase. Unfortunately, it might take the Workload Manager some number of policy adjustment intervals to completely reverse its decisions with respect to relative CPU dispatching priority. Consequently, some time might lapse¹⁶ before the Workload Manager would adjust CPU dispatching priorities such that the high-importance was at a higher dispatching priority than the less-important work.

Consequently, some organizations have critical work that can suffer unacceptably under Goal Mode prior to OS/390 V2R10 because there was no ability for the organization to specify long-term CPU protection for the work.

¹⁵ There are situations, unrelated to attainment of performance goals or importance of the work, when the Workload Manager can control the dispatching of work. These situations include the infamous “small CPU users” concept in which the Workload Manager can assign a very high dispatching priority to a dispatchable unit if it believes that the dispatchable unit will consume only a small amount of CPU time before entering a wait state. Other more rational situations include giving a high CPU dispatching priority to work that holds an enqueue. In both cases, this action is taken in an attempt to quickly move the work through the CPU.

¹⁶ Recall that a policy adjustment interval is 10 seconds.

With OS/390 V2R10, IBM introduced a “long-term CPU protection” option. This option is available with APAR OW43855 installed. With the APAR applied, organizations can specify long-term CPU protection on the WLM ISPF “Modify Rules for the Subsystem Type” panel. This is done by using the new “CPU Critical” option on this panel.

When long-term CPU protection is assigned to work, WLM normally ensures that less important work will have a lower CPU dispatching priority than the work with CPU protection. This option can be useful for critical work that needs to retain high CPU dispatching priority relative to other work, because the work is extremely CPU-sensitive. This requirement would exist regardless of whether the critical work meets or exceeds its performance goal (that is, there should be no situation when less-important work would preempt the critical work from access to a CPU)¹⁷.

CPU protection can be assigned to service classes with address spaces or with enclaves. However, the service class must:

- C Have only one period, and
- C The service class cannot have a discretionary goal.

Additionally, CICS and IMS work have another option that is available when you are creating transaction service classes (these are, service classes for which you specify transaction response goals). For transaction service classes, you can **also** assign long-term CPU protection by specifying YES in the “CPU Critical” field in the rules for specific transactions.

It is important to appreciate that when you specify YES for **one** transaction in a CICS/IMS service class, **all** CICS/IMS transactions in that service class will be CPU-protected.

If a CICS or IMS region is managed as a server by WLM¹⁸ and any of the transaction service classes it serves is assigned CPU protection, then the CICS/IMS region itself is automatically CPU-protected by WLM. This is because CPU protection is actually implemented at the *address space* level, even though you specify CPU protection for a transaction service class. The CICS or IMS *transaction* is not an address space but is processed by the CICS or IMS region, and the region is the address space. Thus, all transactions processed by the region would benefit from CPU protection *while the region is being managed as a server*.

¹⁷Note that there still can exist situations when other work is temporarily given a higher CPU dispatching priority. For example, the “small CPU consumer” algorithm still applies, and work might be given a high dispatching priority because the work is holding an enqueue for which there is contention. Thus, there is no absolute guarantee that critical work will not have less-important work assigned a higher CPU dispatching priority. However, less-important work would not be assigned a higher CPU dispatching priority as a result of the normal Workload Manager policy adjustment algorithms.

¹⁸This would be the case when the region is managed to the response time goals of the transactions it serves.

If a region is **not** being managed as a server of transaction service classes with CPU protection assigned, the region does **not** have CPU protection even though the region was assigned to a service class with CPU protection. For example, suppose that Region A is assigned to a service class that has CPU protection assigned. While Region A is serving transactions that are assigned to a service class with CPU protection assigned, Region A inherits CPU protection. If Region A should not serve transactions that are assigned to a service class with CPU protection assigned, Region A does **not** have CPU protection¹⁹.

Please note the potential problem that occurs when you exclude a CICS or IMS region from being managed as a server by the Workload Manager. This option and the potential problem are discussed in Chapter 1.10 below.

Chapter 1.10: Exemption from server management

If subsystems are installed which support Workload Manager reporting (e.g., CICS/ESA Version 4.1 or IMS/ESA Version 5), installations can define service classes which describe particular transaction types and specify performance goals for the transactions in the service class. All transactions entering the system that fall into the workload category described by the service class are associated with the service class.

The concept of assigning subsystem transactions to service classes is described in detail in Chapter 2 of this section.

One problem with the subsystem transaction concept is the design that once a single subsystem transaction is assigned to a service class, *all transactions entering the system must be classified to either the defined service class or must be classified to a **default service class that must be specified***. For example, if a single CICS transaction is assigned to a transaction service class, all CICS transactions must be assigned to either that transaction service class or assigned to a default transaction service. Additionally, the default transaction service class must have a response time goal specified. Further, **all CICS regions processing the transactions** will be managed by the Workload Manager based on how well the transactions (in both the defined transaction service class and the default transaction service class) are meeting their performance goal.

This is often referred to as the “all or nothing” approach: you must either classify all subsystem transactions to transaction service classes with appropriate performance goals and goal importance, or you may not assign any subsystem transactions to transaction service classes.

¹⁹This design feature was confirmed by analyzing SMF Type 99 data showing when a server (a CICS region) had CPU protection. If the region served only transactions that did not have CPU protection, the region did not have CPU protection even though the region was assigned to a service class that had CPU protection. This is because servers (the CICS region in this case) are assigned to internal WLM service classes (\$SRMSnnn service classes) while they are being managed as servers. The internal service class acquires the CPU protection attribute based on the transaction service classes it serves, not based on the service class to which the address space belongs.

The “all or nothing” approach has caused many installations running under Goal Mode to refrain from classifying subsystem transactions, since some transactions do not lend themselves to response time goals. These installations have simply specified execution velocity goals for all CICS or IMS regions and not classified any subsystem transactions to transaction service classes. Consequently, one of the significant potential benefits of Goal Mode (the ability to manage CICS or IMS transactions based on response goals) has not been realized by these organizations.

With OS/390 V2R10, IBM introduced an “exemption from transaction response time management” option. This option is available with APAR OW43812 installed. With the APAR applied, organizations can specify whether an address space (CICS region or IMS region) will be managed based on the goals of the transactions that the region is serving, or managed based on the goals specified for the region itself. This option is exercised by using the new “Manage Region Using Goals Of:” field on the WLM ISPF “Modify Rules for the Subsystem Type” panel.

When “TRANSACTION” is entered in the “Manage Region Using Goals Of:” field, the region will be managed as a CICS/IMS transaction server by the WLM. “TRANSACTION” is the default specification. If “REGION” is entered in this field, the region will be managed based on the performance goal specified for the service class to which the region is assigned. This performance goal normally would be an execution velocity goal.

When “REGION” is specified, the WLM does not consider the region to be a “server” of transactions²⁰. Rather, the WLM server topology algorithms ignore the region when establishing server topology. Consequently, the goals for any transaction processed by the region will not be considered by the WLM when it determines whether service class periods meet goals and whether policy adjustment is necessary.

This consequence might have undesired implications if you specify goals for CICS or IMS transactions and some or all of those transactions are processed by a CICS or IMS region that has “REGION” specified in the “Manage Region Using Goals Of:” field. In this case, **performance of the transaction service class will not be considered when adjusting resource policy for the region**. This could have the undesired result of transactions not achieving the performance that you desire, simply because the transactions were processed by a CICS or IMS region that was managed based on the goals specified for the region. Alternatively, some transactions might receive better performance than desired because of the same “region-oriented” management by the WLM.

²⁰ Please refer to Chapter 2 (Subsystem Transactions) for a discussion of the servers and served concept.

Chapter 2: Subsystem Transactions

If subsystems are installed that support Workload Manager reporting (e.g., CICS beginning with CICS/ESA Version 4.1 or IMS beginning with IMS/ESA Version 5), installations can define service classes that describe particular transaction types and specify performance goals for the transactions in the service class. All transactions entering the system that fall into the workload category described by the service class are associated with the service class.

Chapter 2.1: Subsystem transaction service classes

Exhibit 4-2 shows a sample of several service classes. Included in Exhibit 4-2 are CICSOR1, CICIPERS, CICADMN, and CICOPS. While Exhibit 4-2 does not identify the nature of the service classes, suppose that the CICSOR1 service class describes a CICS region while the CICIPERS, CICADMN, and CICOPS service classes describe CICS transactions.

In the example, an installation would have defined classification rules to the Workload Manager so all CICS transactions from the Personnel department would be assigned to the CICIPERS Service Class, all CICS transactions from the Administrative Department would be assigned to the CICADMN Service Class, and all CICS transactions from the Operations Department would be assigned to the CICOPS Service Class.

The installation could define specific performance goals for transactions from the Personnel Department, the Administrative Department, and the Operations Department. The installation could specify goal importance for the different goals.

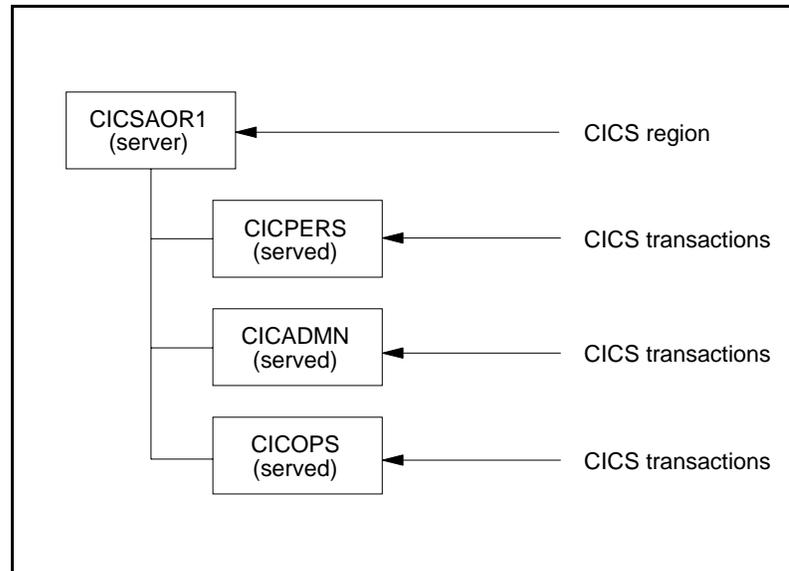
The controlling address space (e.g., the CICS region) must be in its own service class. In our example, suppose that the CICS region was placed into the CICSOR1 Service Class. The CICSOR1 Service Class would be considered a "server" and the CICIPERS Service Class, CICADMN Service Class, and CICOPS Service Class would "served" service classes controlled by the CICSOR1 Service Class.

Exhibit 4-3 illustrates the relationship between the service classes

Notice that the **transactions** comprising the CICIPERS, CICSADMN, and CICSOPS Service Classes must actually execute under control of a CICS region (CICSOR1 in our example) executing CICS of at least CICS/ESA Version 4.1. The CICS region would report transaction performance information to the Workload Manager, and the Workload Manager would attempt to manage system resources to meet the performance goal specified for the CICIPERS Service Class.

The CICSOR1 will have its own performance goals and importance. However, these performance goals and importance normally are used by the Workload Manager **only at**

address space start-up time. After the CICS region has started, its performance goals importance are ignored by the Workload Manager. The Workload Manager will allocate resources based upon the performance goals and importance of the "served" service and classes (in our example, the allocation will be based upon the performance of the CICIPERS transactions, CICADMN transactions, and CICOPS transactions)²¹.



RELATIONSHIP BETWEEN SERVER AND SERVED

Exhibit 4-3

It is important to appreciate that the Workload Manager **does not** allocate resources to the CICIPERS/CICADMN/CICOPS Service Classes, as these service classes are simply logical entities that describe transactions and they are not address spaces. Rather, the Workload Manager allocates resources to the "server" address space (the CICSAOR1 Service Class). Similarly, the Workload Manager does not measure resources consumed by the CICIPERS/CICADMN/CICOPS Service Classes, as CICS does not report this information to the Workload Manager.

One implication of the structure of the "server" and "served" service classes is that the Workload Manager will attempt to meet the performance goals of all "served" service classes that are served by the "server" service class. It does this by allocating resources to the "server" service class. **These additional resources may (or may not) be used by CICS to provide service to the service class missing its goal.**

²¹This statement is not **strictly** true. If the CICS region should become idle for an extended period (no transactions executed in the "served" service classes), the Workload Manager would use the service goal and importance specified for the CICS region service class to manage the region. Practically, of course, there would be little to manage with an idle region.

Suppose there are multiple "served" service classes associated with a "server" service class (CICPERS, CICADMN, and CICOPS in our example). If some "served" service class is failing to achieve its goal, the Workload Manager may allocate additional resources to the "server" service class. These additional resources might allow some "served" service classes to significantly exceed their performance goal and these "served" service classes may not be particularly important.

In our example, the CICS AOR1 Service Class is serving three service classes (CICPERS, CICADMN, and CICOPS).

- Suppose that CICPERS is important but that CICADMN Service Class is of lower importance.
- If the Workload Manager detects that CICPERS is not meeting its performance goal, the Workload Manager may allocate more resources to the CICS AOR1 Service Class.
- The CICS AOR1 would use the additional resources to provide service to CICPERS, to CICADMN, and to CICOPS.
- Consequently, CICADMN or CICOPS might significantly exceed their performance goals²², depending upon how the CICS AOR1 region provided internal dispatching.

To summarize this discussion, performance goals are associated with "served" service classes while resources are allocated to "server" service classes. Performance (i.e., transaction response time) is recorded at the "served" service class level, while resource use is recorded at the "server" service class level.

Chapter 2.2: Subsystem interaction with Workload Manager

Subsystems (e.g., CICS or IMS) communicate with the Workload Manager using Workload Manager Services macros. These macros are described in IBM's *Programming: Workload Management Services* document. This chapter presents a brief overview of the interaction between the subsystem and the Workload Manager.

CICS reports two separate views of the transactions: the *begin_to_end phase* and the *execution phase*²³.

²²Indeed, there is no guarantee that the additional resources would help CICPERS unless the tasks supporting the transactions assigned to CICPERS had been properly **defined to CICS** as a higher priority than CICADMN. Further, tasks might be common between CICPERS and CICADMN (for example). In this case, there is no way to help CICPERS even within CICS AOR1.

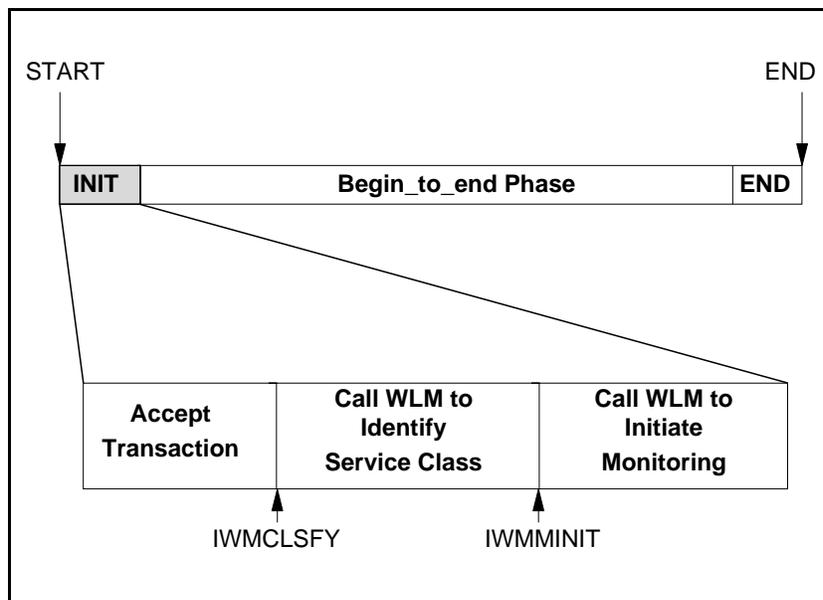
²³IMS Version 5 reports only *execution phase* samples.

- **Begin_to_end phase.** The begin_to_end phase starts when CICS has classified the transaction. This action normally is done in a CICS Terminal Owning Region (TOR).
- **Execution phase.** The execution phase starts when either CICS or IMS has started an application task to process the transaction. For CICS, this normally is done in a CICS Application Owning Region (AOR).

Some CICS transactions may never enter the execution phase, as the transactions will be completely processed in the CICS TOR. Consequently, the number of transactions completing the execution phase may be less than the total number of CICS transactions processed by the system.

CICS provides the System Resources Manager (SRM) with information about the phase (begin_to_end or execution) of transactions by executing the IWMMINIT ("Initialize the Monitoring Environment") macro. The DURATION parameter of the IWMMINIT macro tells the SRM whether the following information related to a transaction is associated with the begin_to_end phase or with the execution phase.

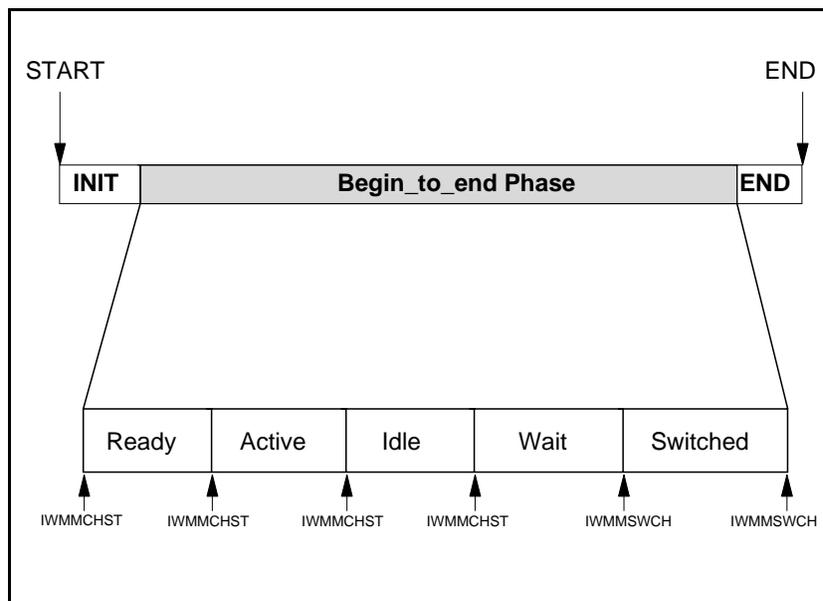
The IWMMINIT macro is issued immediately after CICS has issued the IWMCLSFY ("Assigning Incoming Work Requests to a Service Class") macro to establish a service class for a transaction. Thus, the SRM quickly knows (1) the service class to which a transaction belongs and (2) whether the transaction is in its begin_to_end phase or in its execution phase. Exhibit 4-4 illustrates the sequence of events.



TRANSACTION INITIALIZATION

Exhibit 4-4

CICS or IMS will provide the SRM with information about the "state" of the transaction by issuing the IWMMCHST ("Change State of Work Request") macro. The SRM simply sets bits in a status word to indicate the state of the transaction. Exhibit 4-5 illustrates the sequence of events.



SUBSYSTEM TRANSACTION STATES

Exhibit 4-5

The CICS subsystem work manager reports transaction delays in the following states for the "served" service class:

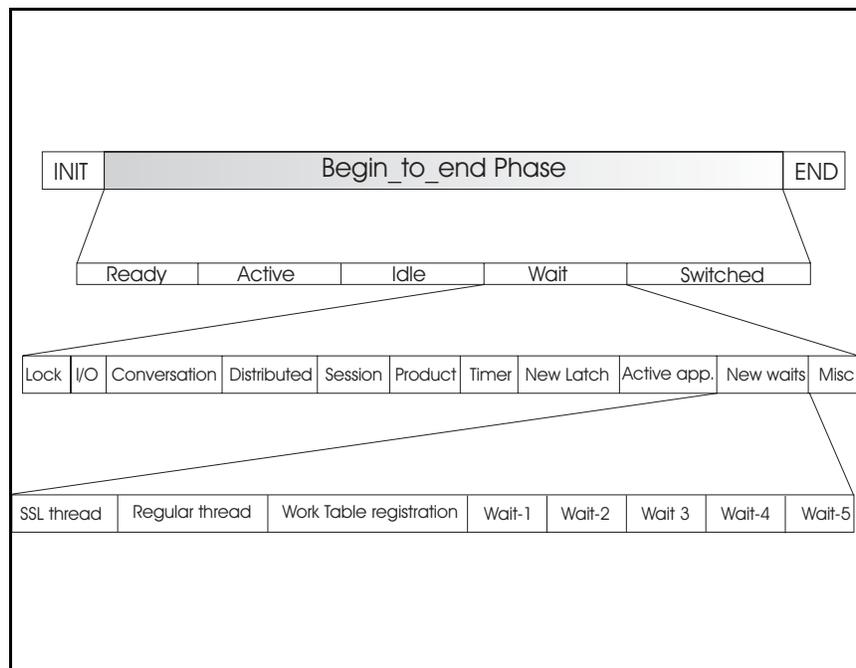
- **Ready state.** The ready state indicates that there was a program ready to execute on behalf of a work request in the "served" service class, but that the work manager has given priority to another work request. In the case of a CICS region, this means that there were more CICS tasks ready to process transactions in the "served" service class than were dispatched by CICS.
- **Active state.** The active state indicates that there was a program executing on behalf of the work request in the "served" service class, from the perspective of the work manager. In the case of a CICS region, this means that a CICS task has been dispatched by CICS to process the transaction.

However, the active state **does not mean that the task is executing** from the perspective of MVS. It simply means that the task has been dispatched by CICS. Other address spaces with a higher system dispatching priority could preempt the task dispatched by CICS, and these other address spaces could be using the CPU. The

situation in which the CICS application task is denied use of the CPU is unknown to CICS.

- **Idle state.** The idle state indicates that there were no work requests (e.g., CICS transactions) ready to run in the service class.
- **Wait state.** The Wait state indicates that a task in support of the transaction was waiting on some activity. The Wait state is broken into several categories: waiting for lock, waiting for I/O, waiting for conversation, waiting for distributed request, waiting for a session to be established (locally, somewhere in the sysplex, or somewhere in the network), waiting for a timer, waiting for another product, or waiting for an unidentified resource.
- **Switched state.** The Switched state indicates that processing of the transaction had been switched from the CICS or IMS work manager providing information to the Workload Manager. The transaction could have been switched to another CICS region (for example) in the same MVS image, switched to another MVS image in the sysplex, or switched to somewhere in the network.

The Wait State is broken into several categories. Exhibit 4-6 illustrates these categories.



WAIT STATE CATEGORIES

EXHIBIT 4-6

- **Waiting for lock.** The waiting for lock state indicates that some work request (e.g., a CICS task) was waiting for a lock.
- **Waiting for I/O.** The waiting for I/O state indicates that the work manager was waiting for some I/O request on behalf of the "served" service class. This state could be waiting on an actual I/O operation or waiting on some other function related to the I/O request.
- **Waiting for conversation.** The waiting for conversation state indicates that the work manager was waiting for a response in a conversation mode.
- **Waiting for distributed request.** The waiting for distributed request state indicates that some function or data must be routed prior to resumption of the work request.
- **Waiting for session to be established locally.** The waiting for session to be established locally means a wait for a session to be established on the current MVS image.
- **Waiting for session to be established in sysplex.** The waiting for session to be established in sysplex means a wait for a session to be established somewhere in the sysplex.
- **Waiting for session to be established in network.** The waiting for session to be established in network means a wait for a session to be established somewhere in the network.
- **Waiting for timer.** The waiting for timer means that a work request was waiting for expiration of a timer.
- **Waiting for another product.** The waiting for another product means that a work request was waiting for another product to provide some service.
- **Waiting for a new latch.** The waiting for a new latch means that a work request was waiting for a new latch. A latch is a short-duration lock.
- **Waiting for SSL thread.** The waiting for SSL thread means that a work request was waiting for a Secure Sockets Layer thread.
- **Waiting for regular thread.** The waiting for regular thread means that a work request was waiting for a regular thread.
- **Waiting for work table.** The waiting for work table means that a work request was waiting for a work table registration.

- **Waiting for unidentified resource.** The waiting for unidentified resource means that the work request was waiting, but that the work manager could not identify the cause of the wait.
- **Wait-1 to Wait-5.** Exhibit 4-6 shows five other wait states: Wait-1 through Wait-5. These states are identified in SMF documentation for z/OS Version 1 Release 4, described in SMF Type 72 (Subtype 3) as R723RW01 through R723RW05. The description given is “For future use”. A similar description is contained in the *MVS Programming: Workload Management Services* with z/OS Version 1 Release 4.

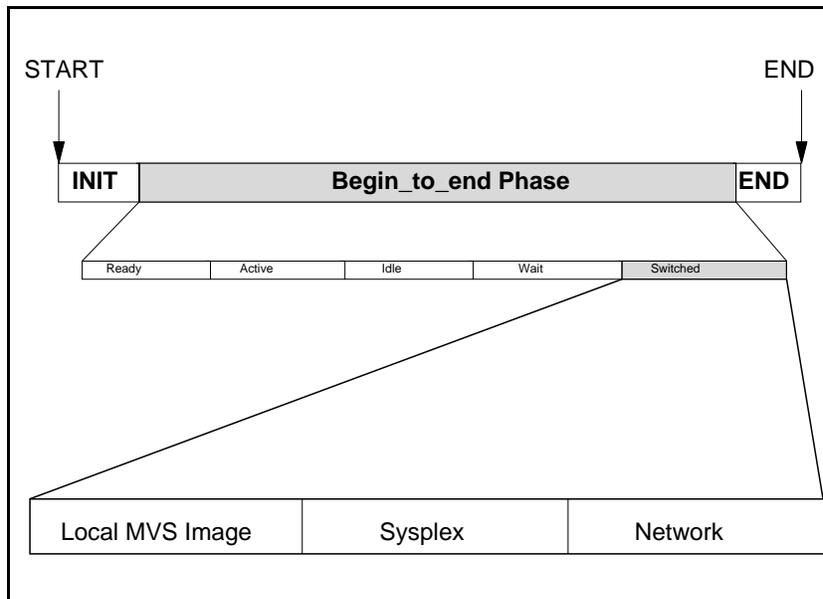
However, some insight can be gained as to the likely meaning of these wait states by examining the *MVS Programming: Workload Management Services* document from earlier releases of z/OS. With z/OS Version 1 Release 3, the *MVS Programming: Workload Management Services* document shows the following wait states related to the WMMCHST (Change State of Work Request Service macro), even though these wait states have never been reflected in SMF data:

- C WAIT, RESOURCE=BUFFER_POOL_IO indicates that the work manager is waiting since the desired data was not found in its expected buffer pool and IO is/will be initiated.
- C WAIT, RESOURCE=BUFFER_POOL_CF indicates that the work manager is waiting since the desired data was not found in its expected buffer pool and is accessing the Coupling Facility Structure for the data.
- C WAIT, RESOURCE=BUFFER_POOL_CF_IO indicates that the work manager is waiting since the desired data was not found in its expected buffer pool nor in its expected Coupling Facility Structure and IO is/will be initiated.
- C WAIT, RESOURCE=CF_IO indicates that the work manager is waiting since the desired data was not found in its expected Coupling Facility Structure and IO is/will be initiated.

Additionally, examining IMS documentation shows still other wait states that are not reflected in SMF data. Consequently, my supposition is that the SMF R723RW01 through R723RW05 fields will contain wait state information that is dependent on the work manager (e.g., DB2, IMS, SMS. etc.) that populates the variables.

The Switched State is broken into three categories, describing where the transaction has been switched. Exhibit 4-7 illustrates these categories.

Many CICS transactions are function-shipped from a CICS Terminal Owning Region (TOR) to a CICS Application Owning Region (AOR). The time spent in the TOR is considered the *begin_to_end phase*. The *execution phase* begins once the transaction is shipped to an AOR.



SWITCHED STATE CATEGORIES

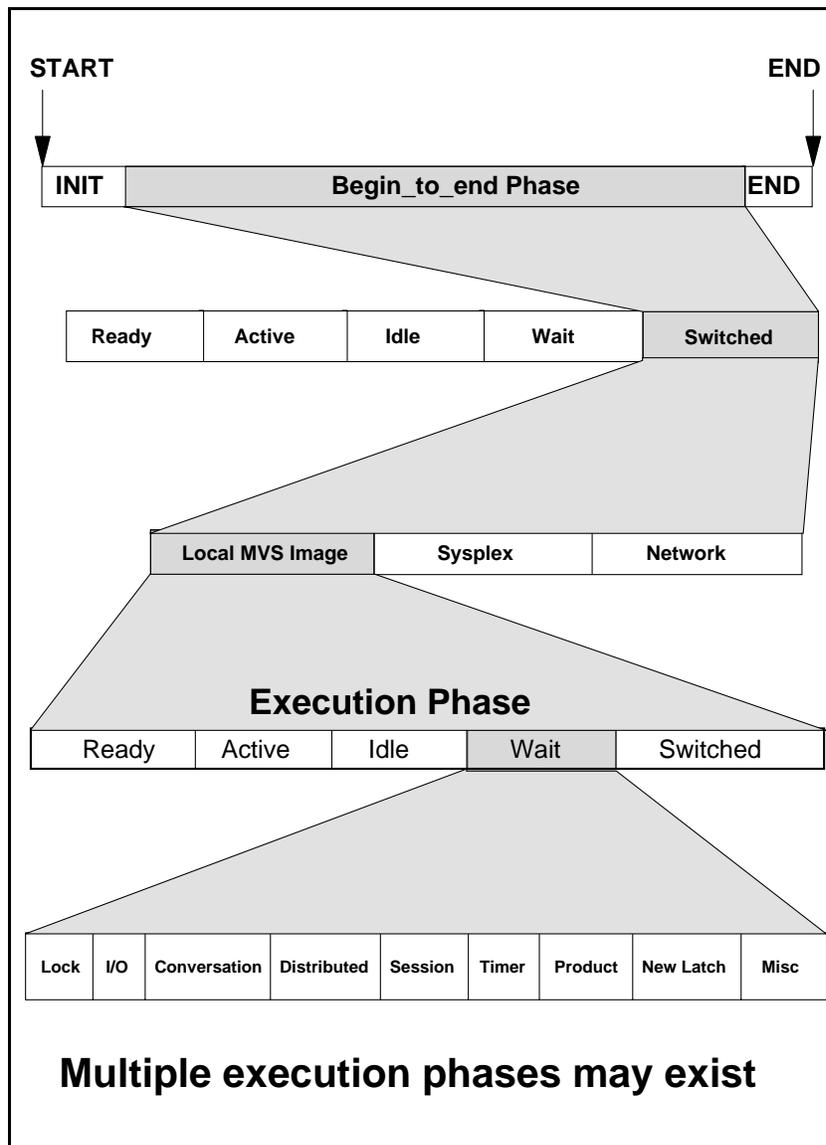
Exhibit 4-7

When the transaction is switched to another subsystem (switched from a CICS TOR to a CICS AOR, for example) in the same MVS image or switched to another MVS image in the sysplex, the subsystem from which the transaction is being shipped indicates that the monitoring environment transaction is being transferred to another subsystem. In this situation, the receiving subsystem provides transaction delay information to the Workload Manager.

Exhibit 4-8 illustrates the execution phase information for a local MVS image. The same information would be available if the receiving subsystem was on another system in the sysplex.

If the transaction is switched somewhere in the network, the Workload Manager has no more information about the status of the transaction; it is simply "switched in the network" from the Workload Manager's view.

Notice that the transaction state information provided in the execution phase is the same as the information provided in the begin_to_end phase. However, the Workload Manager is notified about the phase of the transaction. The transaction state information is maintained separately (and reported separately) for the begin_to_end phase and the execution phase.



EXECUTION PHASE STATES

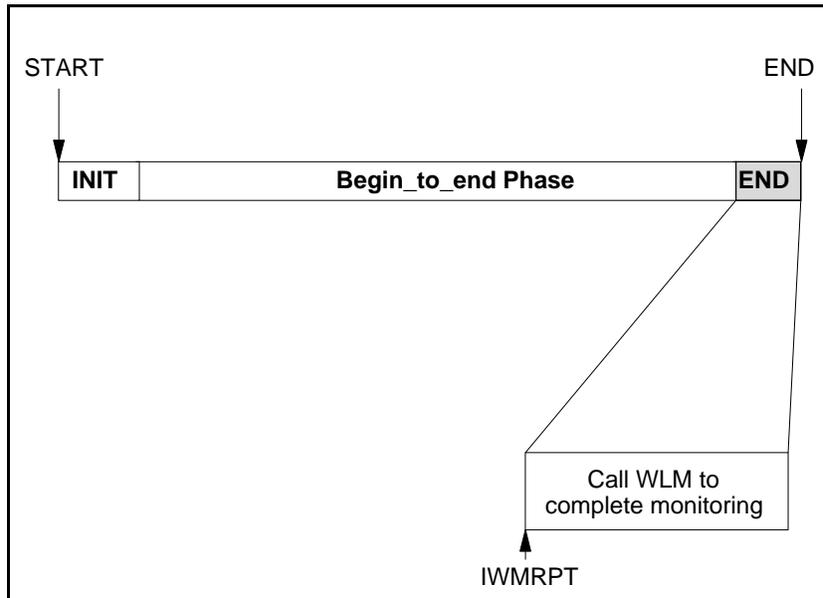
Exhibit 4-8

The level of detail about the state of a transaction in the execution phase depends upon where the transaction is switched.

- If the transaction is switched in the local MVS image to another CICS region, the transaction state information for the various CICS regions is combined in the "CICS subsystem" information.
- If the transaction is shipped to an IMS region (perhaps for data access), the IMS subsystem information is provided separately under the "IMS subsystem" information.

- If the transaction is switched to another MVS image in the sysplex, the information would be separately identified by the Workload Manager executing on that MVS image.

Once a transaction has completed the execution phase, the transaction eventually is switched back to the subsystem that received and classified the transaction (typically a CICS TOR). The `begin_to_end` phase of the transaction ends when this subsystem notifies the Workload Manager to end monitoring of the transaction. Exhibit 4-9 illustrates this sequence.



END OF BEGIN_TO_END PHASE

Exhibit 4-9

The above discussion illustrates three significant times may be associated with a CICS transaction:

- The total elapsed time of the transaction (from START to END)
- The `begin_to_end` phase (time spent in a TOR)
- The execution phase (time spent in an AOR).

Some CICS transactions may never enter the execution phase, as the transactions will be completely processed in the CICS TOR. These CICS transactions are termed "non-routed" transactions. Consequently, the number of transactions completing the execution phase may be less than the total number of CICS transactions processed by the system.

Chapter 3: Workload Manager Internal Logic

The Workload Manager operates on a cooperative basis with the System Resources Manager (SRM). The SRM provides information for the Workload Manager's analysis, the Workload Manager alters SRM control blocks based on its analysis of the information, the Workload Manager issues SYSEVENTs to invoke SRM logic, and the SRM manages system resources based on the contents of the control blocks.

The SRM, in turn, operates on a cooperative basis with other parts of MVS, such as the Real Storage Manager (RSM) and the Auxiliary Storage Manager (ASM) to manage address spaces and manage access by the address spaces to system resources.

This chapter briefly describes the internal logic of the Workload Manager and its interaction with the System Resources Manager.

Chapter 3.1: Resource Use Information

The SRM maintains control blocks to describe each address space under its control.

- The control blocks contain information describing the state of the address space (whether it is ready, waiting, logically swapped, swapped to expanded storage, swapped to auxiliary storage, etc.). The SRM updates the state of each address space as it manages the address spaces (for example, when an address space is swapped, the SRM updates appropriate control block variables).
- The control blocks contain information describing the resources held or used by the address space (the amount of CPU resources used, the number of I/O operations executed, the number of central storage frames held, the number of expanded storage frames held, etc.). Resource use information is placed into SRM control blocks by other parts of MVS. For example, the MVS Dispatcher updates CPU usage information for each dispatchable unit (i.e., TCB and SRB) associated with an address space, the MVS Real Storage Manager updates central and expanded storage information for each address space, and the MVS Auxiliary Storage Manager updates swap/page information for each address space.
- The control blocks contain information describing the page frame referencing characteristics of each address space (the highest Unreferenced Interval Count of any page in central storage, the highest Unreferenced Interval Count of any page in expanded storage, the number of pages at different values of Unreferenced Interval Count). The Real Storage Manager updates the page frame characteristics of each address space.
- The control blocks contain information used by the SRM to control each address space's access to system resources (CPU dispatching priority, the domain to which the

address space is assigned, the CPU resource cap timeslicing pattern, whether page stealing can be used by or against the address space, etc.). The SRM may update the control information or the Workload Manager may update the control information.

Chapter 3.2: Domain Control Information

The SRM maintains system-related control blocks to describe domains to which address spaces are assigned (the minimum and maximum multiprogramming levels associated with the domain, the In-target for the domain and Out-target for the domain, the number of users in the domain and their status, workload manager swap recommendation value, etc.).

When a service policy is activated, the Workload Manager causes the SRM to generate a domain for each service class period in the service policy. The Workload Manager generates domain specifications based on the defaults for the performance goals of each service class.

Chapter 3.3: Service Class Period Information

The Workload Manager generates control blocks to describe each service class period defined in the service policy. These control blocks describe the performance goal, the importance of the goal, the resource group (if any) to which the service class is assigned, etc.

Additionally, the Workload Manager generates control blocks for each service class period to describe the resources used by the service class period, delays to the service class period, etc.

The Workload Manager periodically examines the SRM control blocks describing each address space and acquires samples²⁴ describing the state of each dispatchable unit of an address space (that is, each TCB or SRB associated with the address space). The Workload Manager accumulates the samples into counters that describe the state of the address space²⁵. The samples are summarized by service class period.

- **CPU Using.** This state means that the address space was using the CPU.

²⁴With MVS/ESA SP5.1 Goal Mode, the sampling is done every 250 milliseconds. The sampling interval is recorded in SMF Type 72 records (R723MTVL).

²⁵Address spaces can have more than one dispatchable unit (that is, more than one TCB or SRB). If an address space did have more than one dispatchable unit, the Workload Manager would accumulate state samples to describe the state of each dispatchable unit.

To simplify the discussion, we shall refer to the sample states as being the "state" of the address space, with readers understanding that multiple dispatchable units could simultaneously be waiting on multiple events, or could be using the processor. Thus, for any single sampling, an address space could be counted in more than one state.

- **CPU delay.** This state means that the address space was ready to use the CPU, but was denied access because of dispatching priorities.
- **CPU Capping delay.** This state means that the address space/enclave wanted to use the CPU but it could not because it was capped at that moment. The maximum CPU service units had been consumed for the Resource Group to which the service class was assigned, and the Workload Manager had marked all address spaces or enclaves associated with the Resource Group as non-dispatchable for some cap-slice intervals. This delay does not necessarily mean that address spaces or enclaves in a capped service class had consumed the CPU service units. The CPU service units could have been used by another service class if more than one service class had been assigned to the Resource Group.

The CPU Capping delay state applies only to dispatchable units (TCBs and SRBs) actually on the dispatcher queue awaiting dispatch. The CPU Capping delay state does not apply to dispatchable units waiting for some other event (waiting for I/O, waiting for ENQ, etc.).

IBM introduced *discretionary goal management* algorithms with OS/390 Version 2 Release 6. With discretionary goal management, service class periods that are **overachieving** their goals may have their CPU resources “capped” in order to allow some CPU resources to be used by service class periods with discretionary goals. See a more complete discussion of discretionary goal management in Chapter 1.7 of this section.

- **Swap-in delay.** This state means that the address space was delayed on swap-in (the swap-in had started, but had not completed). These delays should be non-existent or extremely infrequent, unless serious problems exist with the auxiliary storage subsystem.
- **MPL delay.** This state means that an address space was ready to be swapped in, but that the SRM had not initiated a swap-in because of target MPL constraints associated with the domain. Recall that there is a domain created for each service class period. Thus, the Workload Manager had imposed MPL constraints on the service class period to which the address space belonged.
- **Private area page-in from auxiliary storage delay.** This state means that the address space was experiencing page faults in the private area and the pages were coming from auxiliary storage.
- **Common area page-in from auxiliary storage delay.** This state means that the address space was experiencing page faults in the Common Area and the pages were coming from auxiliary storage.

- **Cross-memory page-in from auxiliary storage delay.** This state means that the address space was experiencing page faults in cross-memory access and the pages were coming from auxiliary storage.
- **VIO page-in from auxiliary storage delay.** This state means that the address space was experiencing page faults in VIO and the pages were coming from auxiliary storage.
- **Standard hiperspace page-in from auxiliary storage delay.** This state means that the address space was experiencing page faults in standard hiperspace and the pages were coming from auxiliary storage.
- **ESO hiperspace page-in from auxiliary storage delay.** IBM has defined this state to mean that the address space was experiencing page faults in ESO hiperspace and the pages were coming from auxiliary storage. Pages in ESO hiperspace are, by definition, resident only in expanded storage (ESO = Expanded Storage Only), and are never migrated to auxiliary storage. IBM offers the following explanation²⁶:

"The execution delay for ESO hiperspaces is a calculated value based on the assumption that if an application does a read for an ESO hiperspace page and that page is no longer available (has been cast out), the application will read the data from DASD somewhere.

WLM/SRM takes the number of times a read failed in this way and multiplies it by the number of delay samples we expect a read of a page from DASD to represent and report the product as the execution delay samples for ESO hiperspace. This obviously is not a perfect solution, but we needed some way to get an estimate of how much delay is caused to an address space by not having enough expanded for an ESO hiperspace. Such an estimate is needed to properly manage the amount of expanded owned by the address space to the address space's goal."

- **Unknown delay.** This state means that the Workload Manager was unable to identify the cause of delay. In practice, this means that the delay was caused by something over which the SRM had no control (e.g., I/O operations not under control of the SRM, ENQ delay, etc.).
- **Idle.** This state means that the address space was idle (the address space was in STIMER wait, TSO terminal wait, or APPC wait; or that an initiator was idle).

New sampled values were introduced with OS/390 Version 1 Release 3:

²⁶IBM TALKLink RMF FORUM appended at 15:39:18 on 95/05/29 GMT (by YOCOM at KGNVMC)
Subject: Workload Activity Report. Reproduced with permission of the author.

- **I/O using.** This state means that work was found to be using non-paging DASD I/O resources. Within this context, the work is using the resources in either a device *connect* state (transferring data) or in a device *disconnect* state (seeking and latency).
- **I/O delay.** This state means that the work was delayed for non-paging DASD I/O. The delays include IOS queue, subchannel pending, and control unit queue delays.

New sampled values were introduced with OS/390 Version 2 Release 4:

- **Total delay samples.** The total delay samples is the sum of all WLM-managed delays. The total delay samples value includes batch queue delay (regardless of whether the service class period contained batch jobs that were run in WLM-managed initiators).
- **Total execution samples.** The total execution samples is the sum of the total Using samples (CPU Using samples and I/O Using samples), total Delay samples, Unknown samples, and Idle samples. The total execution samples value includes I/O using samples and I/O delay samples (regardless of whether I/O using and delays were included in the execution velocity calculations).

Please note that the Workload Manager cannot acquire address space state samples for all service classes it is attempting to manage. With CICS/ESA Version 4.1 and IMS/ESA Version 5, a service class may be defined to describe CICS or IMS transactions. The workload classification schemes can identify and assign transactions to particular service classes. Users can define performance goals and importance for these transaction service classes.

However, the transactions are not represented by the SRM as an address space. Rather, the transactions are managed by their subsystem (CICS or IMS). The **subsystem** is an address space (CICS region or IMS message processing region). The SRM maintains information (resource use and address space delay) about the subsystem address space, but does not maintain information about the transactions.

CICS and IMS subsystems communicate transaction response information to the Workload Manager (using the Workload Manager Services macros). Consequently, the Workload Manager can determine whether the transactions are meeting their performance goals.

However, the Workload Manager cannot directly manage resources to the transaction service class to help transactions achieve performance goals. Rather, the Workload Manager can only change SRM control blocks related to the address space controlling the transactions (that is, the Workload Manager changes SRM control blocks related to the CICS or IMS region).

Chapter 3.4: Performance Index

The Workload Manager acquires and analyzes information that allows it to assess whether service classes are meeting their performance goal.

- For response goals, the Workload Manager collects and analyzes **transaction response** information.
- For execution velocity goals, the Workload Manager collects and analyzes **CPU using** information and **processor storage delay** information. With OS/390 Release 3, the Workload Manager collects and *optionally* analyzes **I/O using** and **I/O delay** information.

The Workload Manager periodically assesses the performance of each service class, comparing the performance **achieved** by the service class against the performance goals **specified** for the service class. This assessment is referred to as the "policy adjustment" interval, in that the Workload Manager decides whether to adjust resource allocation policies based on whether service classes are meeting performance goals.

The comparison of performance achieved is accomplished by computing a *Performance Index* for each service class.

- For *average response* goals, the Performance Index is computed by dividing the **actual** response, by the response **goal**. If actual is less than the goal, the Performance Index will be less than one. If actual is greater than the goal, the Performance Index will be greater than one.

For example, suppose that a response goal of 100 milliseconds had been specified. Further suppose that the actual response was 50 milliseconds. Dividing the actual by the goal would yield a Performance Index of 0.5 ($50/100=0.5$). However, suppose that the actual response was 250 milliseconds. Dividing the actual by the goal would yield a Performance Index of 2.5 ($250/100=2.5$).

- For *percentile response* goals, the Performance Index also is computed by dividing the **actual** response by the response **goal**. However, since the goal is a percentage of transactions meeting the response goal, a more complicated algorithm must be used to compute the Performance Index:
 - The Workload Manager calculates the number of transactions required to meet the percentile goal, by multiplying the percentile goal times the number of ending transactions. For example, suppose that the response goal had been stated as "90% of the transactions completing in less than 100 milliseconds". If 200 transactions ended, 180 transactions ($0.90 * 200 = 180$) must end in less than 100 milliseconds to meet the goal.

- Using the response time distribution counters maintained by the Workload Manager, the Workload Manager determines the response time that was **achieved** by the number of transactions required to **meet** the percentile goal. This response time is the "actual" response used in calculating the Performance Index.

Using the above example, suppose that a distribution of transaction response showed that 180 transactions achieved a response of 150 milliseconds or less. 150 milliseconds would be the "actual" response used in the calculation of the Performance Index. The Performance Index would be computed as actual/goal, or $150/100 = 1.5$ (this particular example results in performance worse than the goal, since 90% of the transactions failed to achieve 100 milliseconds or less).

- For *execution velocity* goals, the Performance Index is computed by dividing the **goal** by the **achieved** velocity. If actual is greater than the goal, the Performance Index will be less than one. If actual is less than the goal, the Performance Index will be greater than one.

For example, suppose that an execution goal of 30% had been specified. Further suppose that the actual execution velocity achieved was 50%. Dividing the goal by the actual would yield a Performance Index of 0.6 ($30\%/50\%=0.6$), indicating that the service class had met its goal. However, suppose that the actual execution velocity achieved was only 15%. Dividing the goal by the actual would yield a Performance Index of 2.0 ($30\%/15\%=2.0$), indicating that the service class had not met its goal.

- For *discretionary* goals, the Performance Index is arbitrarily established as 0.81 by the Workload Manager. The 0.81 Performance Index value means that service class periods with a discretionary goal are always considered to be meeting their goal from the view of the Workload Manager.

Exhibit 4-10 summarizes the results from the above discussion. A Performance Index less than one implies that a performance goal **has** been met, while a Performance Index greater than one implies that a goal has **not** been met. A Performance Index of exactly one implies that the performance goal has been **exactly** met.

<p>PI > 1 => performance worse than goal PI = 1 => exactly on goal PI < 1 => performance better than goal</p>
--

PERFORMANCE INDEX EFFECT

Exhibit 4-10

The Performance Index can be used to compare the performance of service classes, regardless of the type of performance goal specified for the service class²⁷. This approach has the appeal of creating a **single number** that can be used to evaluate the performance of service classes based upon how well or how poorly the service classes meet their performance goals.

Chapter 3.5: Policy Adjustment

The Workload Manager periodically assesses the performance of each service class, comparing the performance achieved by the service class against the performance goals specified for the service class. This assessment is referred to as the "policy adjustment" interval, in that the Workload Manager decides whether to adjust resource allocation policies based on whether service classes are meeting performance goals. The policy adjustment is performed every 10 seconds.

As the initial step of policy adjustment, the Workload Manager implements the specifications of any resource groups (minimum and maximum CPU service units). **This step is performed regardless of the importance of the service class periods associated with the resource group.**

The Workload Manager initially assesses performance based on the *sysplex performance index* computed for each service class period. This assessment is done at each goal importance level. Policy adjustment actions are evaluated for the worst-performing service class period at the highest goal importance, then the next worst-performing, etc. It is important to realize that only one service class period will be "helped" by the policy adjustment algorithms per policy adjustment interval²⁸.

If the Workload Manager has evaluated the performance of all service class periods at the highest goal importance *based on sysplex performance index* and no action has been taken, the next step depends on whether APAR OW25542 has been applied.

- **OW25542 has not been applied.** With the normal logic, the Workload Manager will examine the performance of all service class periods at the next-highest goal importance *based on sysplex performance index*. The Workload Manager will continue analyzing performance at successively lower goal importance levels, based on sysplex performance index. After the performance of all service class periods with goals have been analyzed with no action, the Workload Manager will perform the analysis beginning with the highest goal importance, **using the local performance index** as the measure of performance,.

²⁷A discretionary goal has an implied performance index of 0.81, which means that service classes with discretionary goals will always be considered as achieving their service goal.

²⁸Recall that the policy adjustment interval is 10 seconds of elapsed time.

- **OW25542 has been applied.** With OW25542, the Workload Manager will examine the performance of all service class periods at the highest goal importance **using the local performance index** as the measure of performance. The Workload Manager will continue examining performance at successively lower goal importance levels, analyzing performance based on sysplex performance index followed by an analysis of performance based on local performance index.

Both the original design of the Workload Manager and the fix for OW25542 operate under a basic assumption: that a sysplex consists of multiple systems configured in a symmetric manner, and that service class periods can operate on any system in the sysplex. If the workload being processed consists of transaction service classes such as CICS transactions managed by CICSplex/SM and routed to any system in the sysplex to be processed in cloned CICS regions, this view of the sysplex makes sense.

From this perspective, all systems in the sysplex can be viewed collectively as a pool of resources and the performance of the transactions can be evaluated based on how well the transactions perform on the sysplex. If a service class period is not meeting its performance goal *on the sysplex*, action may or may not be necessary at a local system level. Consequently, **sysplex performance index** is the basic measure of performance used in the Workload Manager design.

Unfortunately, this logic does not work in all situations. Consider a site that has established a service class for TSO trivial transactions. The TSO users might log onto, for example, two systems: SYSA and SYST. The users on SYSA might represent production work while the users on SYST might represent TSO testing (and might not be as important to the site as the production work).

The first service class period encountered with a Performance Index greater than one is selected for attention, as a Performance Index greater than one means that the performance goal is not being met. There are two exceptions to this selection:

- The Workload Manager will not attempt to reallocate resources if the Performance Index is only slightly above one²⁹. This prevents application of the resource allocation algorithms where only marginal performance improvement is to be expected.
- If a service class cannot be "helped" after the policy adjustment algorithms are executed and analyzed, the Workload Manager will set a "skip counter" for the service class. The skip counter initially is set to three and decremented each policy adjustment interval, until the skip counter reaches zero. While the skip counter for a service class is above zero, the Workload Manager will ignore the service class unless performance significantly degrades.

²⁹The actual value used is 1.1 (that is, the Workload Manager will not attempt to reallocate resources if the Performance Index is 1.1 or below).

The reasoning here is that there is little point in continuing to try to help a service class if the first attempt fails. It is better to wait awhile with the expectation that system conditions will change.

The Workload Manager waits 20 seconds (2 policy adjustment intervals) before the service class is again subjected to the policy adjustment algorithms. If performance seriously degrades for the service class, the skip logic is ignored, and the service class will be selected for policy adjustment if it has a sufficiently high importance.

Once a service class has been selected for consideration of policy adjustment, the Workload Manager considers the service class to be a "goal receiver" of resources and begins looking for a service class that can be a "donor" of resources. A potential donor is selected in inverse order by which a receiver is selected (that is, the least important service classes are examined and a service class from that group with the lowest Performance Index is selected as a potential donor).

Within the same Goal Importance, the Workload Manager will not take resources from a service class to help another service class unless the expected performance improvement to the "receiver" service class outweighs the expected performance degradation to the "donor" service class.

A variety of algorithms are applied to assess whether the receiver service class can benefit from additional CPU dispatching priority, additional processor storage, etc. These algorithms are illustrated in IBM's *Programming: Workload Management Services*, Chapter 4 (Using SMF Record Type 99).

Additionally, if the "goal receiver" and potential "donor" are at the same goal importance level, the harm to the donor service class by losing the resources is evaluated. Within a goal importance level, the Workload Manager reallocates resources only if the projected benefit outweighs the projected harm. This evaluation is not done if the potential "donor" is at a lower importance level than the "receiver". The resource allocation is made regardless of the potential harm to the "donor" as the goal importance takes precedence.

If policy adjustment algorithms conclude that resource adjustment would be beneficial, the Workload Manager adjusts the policy by altering the SRM's internal controls so the SRM will manage system resources according to the new controls.

Only **one** receiver is helped during a single policy adjustment interval, although multiple donors may be required to provide the necessary resources. Only one receiver is helped since the Workload Manager must acquire additional data to evaluate whether the particular resources helped performance for the receiver.

Exhibit 4-11 illustrates a sample ordering of service classes within Goal Importance and Performance Index (the service classes are arbitrarily numbered 1-12 for the sake of

brevity). This exhibit will be used to illustrate the above concepts, **assuming that OW25542 has not been applied.**

SERVICE CLASS	SYSPLEX	
	IMP	PERF INDEX
1	1	1.1
2	1	0.8
3	1	0.7
4	1	0.6
5	2	4.2
6	2	3.9
7	2	1.0
8	2	0.9
9	2	0.7
10	3	6.0
11	3	5.3
12	3	1.2

SAMPLE ORDERING OF SERVICE CLASSES

Exhibit 4-11

- The Workload Manager will examine the service classes in the order shown (top to bottom) since Service Class 1 has the highest Goal Importance³⁰ and this service class has the highest Performance Index within Goal Importance 1.
- The Workload Manager takes no action on Service Class 1 since the Performance Index is 1.1, and the Workload Manager does not wish to try to help a service class that is very close to achieving its performance goal.
- The Workload Manager next examines Service Class 2. This service class and all subsequent service classes at Goal Importance 1 are skipped, since the Performance index for these service classes is less than one (indicating that the service classes are achieving their performance goal).
- The Workload Manager will next examine Goal Importance 2, beginning with Service Class 5. This service class has a Performance Index of 4.2, so the Workload Manager will attempt to help the performance of this service class. Service Class 2 will be selected as a potential "receiver" for resources.

³⁰Recall that Goal Importance 1 is the highest, while Goal Importance 5 is the lowest. Goal Importance 6 is used internally by the Workload Manager for discretionary work since discretionary work is the lowest importance of all.

- The Workload Manager will select Service Class 12 as a potential "donor" of resources to Service Class 5. Service Class 12 was selected because it was the service class with the lowest Goal Importance and the lowest Performance Index.
- The Workload Manager will arbitrarily take any necessary resources from Service Class 12 in an attempt to help Service Class 5.
- If Service Class 12 does not hold enough resources to help Service Class 5, the Workload Manager will select Service Class 11 as a potential donor, and then select Service Class 10 as a potential donor.
- Suppose that Service Class 10, Service Class 11, and Service Class 12 do not hold sufficient resources to help Service Class 5. The Workload Manager will then select Service Class 9 as the next potential donor. Service Class 9 would be selected because it was the service with the next lowest Goal Importance and the lowest Performance Index.
- Once the Workload Manager selected Service Class 9, the Workload Manager would take resources from Service Class 9 **only if the projected performance improvement for Service Class 5 exceeds the projected performance degradation to Service Class 9**. The reasoning here is that the service classes are at the same Goal Importance, and resources should be reallocated to service classes at the same importance only if there is a "net gain" for the action.
- Suppose the Workload Manager concluded that the performance of Service Class 5 could not be significantly improved after analyzing Service Class 6 through Service Class 12. The Workload Manager would set a "skip counter" for Service Class 5 and select Service Class 6 as a potential "receiver" of resources. Unless the performance of Service Class 5 **significantly** deteriorated, the Workload Manager would skip Service Class 5 for three subsequent policy adjustment intervals.

The reasoning here is that (1) Service Class 5 could not be improved with current system conditions and (2) perhaps system conditions would change after some elapsed time. The three policy adjustment intervals equates to an elapsed time of 30 seconds real time.

The Workload Manager may adjust resource allocation policy in the following areas:

- CPU dispatching priorities and CPU capping
- MPL targets for domains (recall that a domain is created for each service class period).
- Swap protect time (the time address spaces will be protected in processor storage before being eligible for swap to auxiliary storage).

- Expanded storage policies (swap working set pages, VIO pages, hiperspace pages, and stolen pages/swap trim pages).
- Protected storage targets (central storage protection, central storage restriction, expanded storage protection, expanded storage restriction).

With OS/390 Release 3, the Workload manager may adjust resource allocation policy in the I/O priority queuing area. I/O priority queuing is used to control non-paging DASD I/O requests that are queued because a device is busy.

- Without I/O priority queue management, the Workload Manager sets I/O priorities equal to the dispatching priority for the service class period.
- With I/O priority queue management, the Workload Manager will dynamically adjust the priority of I/O requests based on how well the service class is meeting its goal, and whether the device can contribute to meeting the goal. The Workload Manager will consider I/O priority queue adjustment only if you have specified *I/O priority management = yes* in the Workload Manager Service Coefficient/Service Definition Options panel.

If policy adjustment algorithms conclude that resource adjustment would not be beneficial, the Workload Manager selects another potential receiver service class (in order of ascending importance and descending Performance Index).

Chapter 4: Workload Manager Performance Data

The primary source of performance information about the performance of service classes is contained in SMF Type 72 (Subtype 3) records³¹. The Type 72 records contain a **wealth** of information, with each record describing a service class period in great detail. The Type 72 records include:

- Workload Manager control information: service policy name and activation time, service coefficients, workload group name, etc. This information is collected from the service policy in effect when the SMF Type 72 record was written.
- Service class served data (for subsystems serving other service classes): the name of each service class served and the number of times the service class was served. This information is collected from the service policy in effect when the SMF Type 72 record was written.
- Resource group information (if a service class is associated with a resource group): resource group name, CPU minimum, and CPU maximum. This information is collected from the service policy in effect when the SMF Type 72 record was written.
- Service class period data is available in several categories
 - Description of service class period: type of performance goal, value of the performance goal, importance of the goal, and performance period duration. This information is collected from the service policy in effect when the SMF Type 72 record was written.
 - Resource data: CPU service units, I/O service units, central storage service units, page-in counts by several categories from auxiliary and expanded, storage residency time, swap count, and service times.

The resource values are not available for "served" service classes (e.g., CICS transactions), but are available for the "server" subsystems (e.g., the CICS regions serving the transactions). Please refer to Chapter 2 of this section for discussion of the served and server service classes.

- Transaction data: transaction elapsed and execution times, and transaction ending counts.

³¹The SMF Type 99 records contain extremely detailed information about the internal logic and decisions of the Workload Manager. The Workload Manager creates a "trace" of its logic as it examines service classes, makes resource adjustment decisions, makes policy adjustment decisions, working set management decisions, etc. The Type 99 records are written every 10 seconds, and may be extremely large. IBM strongly recommends that Type 99 records be collected for **only** a short time in response to specific needs. Since the Type 99 records are not normally collected, CPEXpert does not consider them to be a primary source of performance data. In contrast, the Type 72 records are routinely collected at virtually every site.

- Samples describing address space states: CPU using samples, execution delay samples, and other samples related to address spaces in the service class.
- Response time distribution data (for service classes with response goals): a distribution of response time into 14 counters of transactions ending with response times relative to the performance goal.

The following table shows the percentages as a function of the response goal, recorded into each counter. Each percentage reflected in the table describes a "counter" recorded by RMF in SMF Type 72 records, and reflects a count of the transactions with response times as the indicated percentage of the response goal. For example, a count of "13" in the second counter (50%-60%) would indicate that 13 transactions ended with a response time of between 50% and 60% of the response goal.

COUNTER GOAL	PERCENT OF
1	LESS THAN 50%
2	50%-60%
3	60%-70%
4	70%-80%
5	80%-90%
6	90%-100%
7	100%-110%
8	110%-120%
9	120%-130%
10	130%-140%
11	140%-150%
12	150%-200%
13	200%-400%
14	OVER 400%

**RESPONSE TIME DISTRIBUTIONS
SMF TYPE 72 RECORDS**

Exhibit 4-12

The last counter contains a count of transactions that exceeded 400% of the performance goal.

Additionally, for subsystems supporting Workload Manager interface (initially, CICS/ESA Version 4.1 and IMS Version 5), the following work manager/resource manager information is available in the Type 72 records:

- Work Manager/Resource begin_to_end phase state samples: active state, ready state, idle state, waiting states, and switched states.
- Work Manager/Resource execution phase state samples: active state, ready state, idle state, waiting states, and continuation states.

If transactions in the service class are served by both CICS and IMS, the Type 72 records contain begin_to_end phase and execution phase information for CICS and only execution phase information for IMS.

These subsystem states are described in Chapter 2 of this section.

Chapter 5: RMF Data Analysis Considerations

This chapter highlights some of the factors that must be considered when analyzing workload data collected and recorded by RMF in SMF Type 72 records.

These factors do **not** preclude a comprehensive analysis of performance data and usually do not prevent insight into the causes of unacceptable performance. However, the factors must be recognized and accounted for both by CPExpert in analyzing data and by the user in reviewing CPExpert's results. The factors stem from (1) the way in which the SRM defines transactions, (2) the way in which RMF collects transaction counts and times, and (3) operator actions that cause unique data recording.

Chapter 5.1: SRM Transactions

Within MVS, the concept of a "transaction" is based upon the SRM's definition of a transaction. This definition differs depending upon whether the transaction statistics relate to TSO, relate to batch, or relate to a started task (such as CICS). The Workload Manager uses the SRM services to acquire information and modifies SRM control blocks to adjust system policies. Consequently, the SRM's definition of a transaction is relevant to understanding RMF data.

- TSO transactions begin (1) whenever terminal input is entered and the line is not continued, (2) the 3270 field mark key separates commands on the same input line, (3) a command's output is detained while waiting for an output buffer, or (4) a command is taken from the TSO internal stack.
- Batch transactions correspond to a job in most cases. Batch jobs are single transactions counted in their service class, unless they transition to a different performance period. In this case, the batch transaction is "ended" and a new transaction begins with the new step and the new performance group.
- Most started tasks are counted as a single transaction for their entire execution time (unless they employ special techniques to communicate "transaction" information to the SRM, or unless SRM counters are in danger of overflowing). Consequently, individual transactions in support of started tasks are not managed or counted by the SRM³².

For example, CICS transactions are managed by CICS, rather than the SRM. The SRM views the entire CICS region as a single transaction that ends when the CICS region ends.

³²With CICS/ESA Version 4.1, CICS communicates some transaction information to the Workload Manager if the CICS transactions have been placed into their own service class. Similarly, with IMS/ESA Version 5, IMS communicates some transaction information to the Workload Manager. Consequently, the Workload Manager has some information about transactions, but the SRM does not collect resource use at the transaction level, as the SRM manages and collects information on address spaces.

With versions of MVS prior to MVS/ESA SP5.1 (Goal Mode), it was convenient to think of a transaction as being synonymous with an "address space" since only one transaction (from the SRM's view) could be active in an address space at one time. Of course, subsystems (e.g., CICS, DB2, IMS) could process many transactions, but as explained above, the SRM would view the subsystem as a single transaction.

With MVS/ESA SP5.1, the SRM still views the address space as a single transaction. This view is still relevant for MVS components except subsystems (that is, the view is relevant for TSO transactions, batch jobs, etc.). RMF records transaction information from the SRM's view (e.g., transaction counts, transaction active time, transaction elapsed time, etc.).

With MVS/ESA SP5 (Goal Mode), subsystems can communicate transaction information to the Workload Manager via Workload Manager Services macros. These transactions are individual CICS transactions and individual IMS transactions. The Workload Manager accumulates transaction information for the subsystems, and RMF records the subsystem information in SMF Type 72 (Subtype 3) *Work Manager/Resource Manager Section*. The Work Manager/Resource Manager data will be discussed later in Section 5.4. Initially, we will restrict our discussion to transactions from the SRM's view.

A transaction becomes "ready" from the SRM's view when the transaction starts. Transaction active time and transaction elapsed time begin at this point.

The address space supporting a newly-ready transaction is in a swapped out state (except for CICS or IMS regions if they have been made non-swappable). The address space must be brought into the system and made a part of the multiprogramming set³³. The address space may remain swapped out for some period of time before being swapped into storage.

After some swap delay, the address space supporting the transaction is made resident in storage (or the address space supporting the transaction's control blocks are updated, in the case of logically swapped address spaces). Once resident, the address space may be swapped out of storage for a large number of reasons (these reasons are globally recorded in SMF Type 71 records and include exchange swaps, unilateral swaps, etc.). If the swap was uncontrolled by the address space, the time is counted as transaction active time (the address space is still ready to execute).

Thus, the **transaction active time** is the address space resident time plus the time the address space was swapped out but ready to execute. The **transaction elapsed time** is the total time from transaction start to transaction end, regardless of whether the address space was ready to execute.

³³This discussion ignores the implications of logical swapping, in which the transaction is submitted by an interactive user, and the address space may actually reside in storage even though "logically" swapped out.

All transactions are counted by the SRM when they end, with the above definition of "ending" transactions. Transaction counts and the elapsed times of transactions are recorded **when the transaction ends**. This method of recording can have a profound impact on transaction counts and times as recorded by RMF, and on averages based upon these transaction counts and times.

Chapter 5.2: RMF Transaction Information

Transaction-related information is acquired by RMF at the end of a recording interval. The information is recorded in SMF Type 72 (Subtype 3) records. RMF **normally** records this information at user-specified intervals (e.g., users commonly specify RMF recording every 15 or 30 minutes)³⁴.

There are two implications with the information recorded into SMF Type 72 records: (1) RMF normally collects and records information at regular intervals but (2) the transaction counts and times are available to RMF only when transactions end. These two facts can distort all computations based upon transaction counts or times.

- **Average response time.** Most analysts compute average response time by dividing the transaction elapsed time (SMF72TTM) by the transaction count (SMF72TTX). These SMF variables are applicable for MVS prior to Goal Mode; with Goal Mode, the variables are R723CTET and R723CRCP, respectively. These are the values used to compute response time in the RMF *Workload Activity Report*.

The transaction elapsed time represents the entire transaction elapsed time, including long wait (e.g., time enqueued, waiting for a response to WTOR, etc.). The long wait part of some transactions can be extremely lengthy. The lengthy elapsed times of only a single transaction can completely distort the average transaction time. In fact, transactions can be in a long wait condition for many minutes. In some situations³⁵, the average response resulting from using SMF72TTM (or R723CTET) can exceed the RMF recording interval in which the transactions finally end!

This situation is not usually serious for the computing the average response times of a TSO Period 1 service class. TSO Period 1 is intended to service interactive transactions. These transactions are very short relative to the RMF recording interval, and have less of an opportunity to "spill over" into a RMF recording interval different from which they started. Additionally, there usually is a large number of transactions ending in TSO Period 1 for any RMF interval. Thus, there is a greater chance of "evening out" the average number transactions ending.

³⁴There are important exceptions to this normal recording, as discussed later.

³⁵For example, such distortions might occur if TSO were used by a single operator on the midnight shift, and extensive Enqueue time was experienced.

This same reasoning normally applies to TSO Period 2, although more care must be taken with the results.

The situation becomes more serious for TSO Period 3 or Period 4 transactions and for batch jobs. The results can be seriously misleading if average resource per transaction is computed for these workloads based upon SMF Type 72 workload information.

Fortunately, IBM provides **response time distributions relative to the performance goal** for all service class periods that have response performance goals. Thus, although average response time may not be particularly meaningful, the response time distributions can be analyzed to understand the response experienced by the service class period.

- **Average resources used.** The resources used by transactions (e.g., CPU service units, memory service units, etc.) and the resource counts (e.g., swap sequences and page-ins) are **continually accumulated**. This information is acquired by RMF in the recording interval **in which the resources were used**.

In computing the average resource per transaction, the resource is divided by the count of ended transactions (SMF72TTX or R723CTET). As explained above, the count does not necessarily represent the transactions actually using resources. There could be many transactions (e.g., long-running batch jobs) in the multiprogramming set. These transactions could use significant resources, but their count would not be reflected in computing the average resources used unless they ended in the RMF recording interval.

There are several additional scenarios that complicate the average resource computations:

Resources are accumulated in the performance **period in which they are used**. Transactions are counted in the performance **period in which they end**. For example, the resources required to support the TSO service class Period 2 transactions while they were in Period 1 are attributed to Period 1.

However, the TSO Period 2 transactions are not counted in Period 1; they are counted in Period 2 if they end in Period 2 (however, if they migrate to Period 3, they are not counted in Period 2 but would be counted in Period 3 if they end in Period 3). This situation has the effect of inflating average resources used in Period 1 by whatever was used by transactions ending in Period 2. (Of course, the same applies for resources used by transactions ending in Period 3 and Period 4.)

If numerous transactions pass through Period 1 to (for example) Period 4 within a single RMF measurement interval, the average service rate for Period 1 transactions could be extremely large. In fact, the average service rate could be far more than the computer

system was able to deliver! Such data anomalies can give analysts concern if they are not aware of the causes.

It is possible to "back out" some of this resource use by estimating the resources used by TSO transactions ending in Period 2, Period 3, etc. However, there is no guarantee that these transactions will end in the same RMF measurement interval, and thus no knowledge of how much should be "backed out" of the resources.

CPEXpert exploits this situation for service classes having multiple periods and having a relatively short response goal for Period 1. CPEXpert analyzes the average service used in Period 1 compared to the DUR value for Period 1. If the average service used is much greater than the DUR value, conclusions can be made about non-interactive transactions executing in Period 1.

Chapter 5.3: Operator Actions

An additional problematic situation occurs if the service policy is changed (even if the service policy is changed simply to implement policy overrides)³⁶. When the service policy is changed, RMF writes out SMF Type 72 records before the new service policy takes effect.

When the service policy is changed, all ongoing transactions are ended for any service class effected, and the transaction is restarted in the first period of their service class.

RMF collects workload and resource information before writing out its records. Therefore, changing the service policy could have the effect of creating a large number of "ended" transactions in a short RMF recording interval. This large number would seriously distort the workload averages computed for the period.

After writing out the information acquired before the new service policy takes effect, RMF will synchronize the writing of SMF Type 72 records with the writing of other system records at the user specified interval (e.g., every 15 minutes). Thus, if the service policy is changed, there will be two SMF Type 72 records written for the RMF recording interval in which the service policy change occurred. Either of these SMF Type 72 records could represent a **very** short interval if the policy change occurred just after the previous recording interval or just before the current recording interval.

There potentially is a serious performance implication of changing the service policy. The Workload Manager will require some elapsed time to accumulate statistics about the performance of service classes in the new service policy. Depending upon the number of

³⁶The same effects occur if a mode change is made (switching between Compatibility Mode and Goal Mode). This discussion would become cluttered if all change references used "service policy change or mode change" to describe the effects. Of course, the effect would be more serious if a mode change were made. However, mode changes are likely to be infrequent once users are comfortable with Goal Mode operation.

service classes and types of performance goals, some significant elapsed time could be required for the Workload Manager to adjust resource allocation to the new service policy.

Additionally, all ended transactions would restart in Period 1 of their service class. This could have the effect of suddenly placing non-interactive transactions into Period 1 of the service class, and these transactions would compete with interactive transactions for system resources. As a result, there could be some interval of poor response to the interactive users.

Chapter 5.4: SRM Sampling of Subsystem State

With CICS/ESA Version 4.1 or IMS Version 5, these subsystems will provide the Workload Manager with information about the state of the transaction (active state, ready state, waiting state, etc.) by issuing the IWMMCHST ("Change State of Work Request") macro. The Workload Manager simply sets bits in a status word to indicate the state of a transaction.

The SRM periodically samples the status word associated with each transaction³⁷, and updates counters representing the state of transactions executing in the service class. There is a status word for the begin_to_end phase and a status word for the execution phase, and separate sets of counters are maintained for the various begin_to_end states and execution states for each service class.

The SRM also keeps a count of the number of samples that it takes of the begin_to_end phase and of the execution phase.

The counts of various samples are recorded in the "Work Manager/Resource Manager State Section" of SMF Type 72 records.

The SRM also includes the elapsed time of transactions (R723CTET) and the count of transactions (R723CRCP) in the SMF Type 72 records. Based on the transaction elapsed time and transaction count, CPExpert can compute the approximate number of samples that the SRM **should** take of the begin_to_end phase of transactions.

To illustrate the computation, suppose that a single transaction were to execute in a service class, and further suppose that the transaction elapsed time was 1 second. During this second of elapsed time, the SRM should take a sample every 250 milliseconds (4 samples per second), or 4 samples of the begin_to_end phase³⁸ of the transaction of the 1-second transaction. If two transactions with individual elapsed times of 1 second were to execute

³⁷With MVS/ESA SP5.1, the SRM takes its samples every 250 milliseconds.

³⁸For the moment, we can ignore the time required by MVS to assign the transaction to a CICS region, the time for the CICS region to issue the IWMLCLSFY macro, the time for the Workload Manager to classify the transaction to a service class, and the time for the CICS TOR to issue the IWWMINIT macro. These times normally are very small.

in the service class, the SRM should take 8 samples (1 second average elapsed time * 2 transactions * 4 samples per second = 8).

Thus, the computation of the number of samples that the SRM **should** take in any RMF measurement interval is simply the total elapsed time of transactions, times the sampling rate. The result from this computation should always be more than the number of samples the SRM actually took of the begin_to_end phase, since the begin_to_end phase does not start until after the transaction has entered the system and has been classified to a service class, and the begin_to_end phase ends before the transaction is finally marked "ended" by the SRM.

However, the SRM updates the elapsed time of transactions only when the transactions end. Suppose that a never-ending transaction executed in the service class. The SRM would initialize the begin_to_end phase and observe subsequent state changes in the begin_to_end phase (and perhaps in the execution phase). However, the SRM would never see the transaction complete and thus would not update the elapsed time of the transaction.

A similar situation occurs with long-running transactions. These transactions can span RMF measurement intervals; the SRM would initialize the begin_to_end phase and observe subsequent state changes in the begin_to_end phase (and perhaps in the execution phase) in one RMF interval. The elapsed time of the transaction might not be recorded until a subsequent RMF interval.

These anomalies can cause response time calculations to be misleading. More importantly, the Workload Manager algorithms may be less effective if never-ending or long-running transactions are in the same service class as interactive transactions. This is because the Workload Manager's computation of response times may be distorted by the long-running transactions.

CPEXpert detects many of these situations and either adjusts its analysis of performance problems or produces rules to alert you to the problems.

For example, Rule WLM110 to Rule WLM113 describe the results of CPEXpert's analysis to detect never-ending or long running transactions executing in a service class with interactive transactions.