Workload Manager Goal Mode—for the Trailing Edge

Stephen L. Samson
Senior Technical Staff Member
Candle Corporation

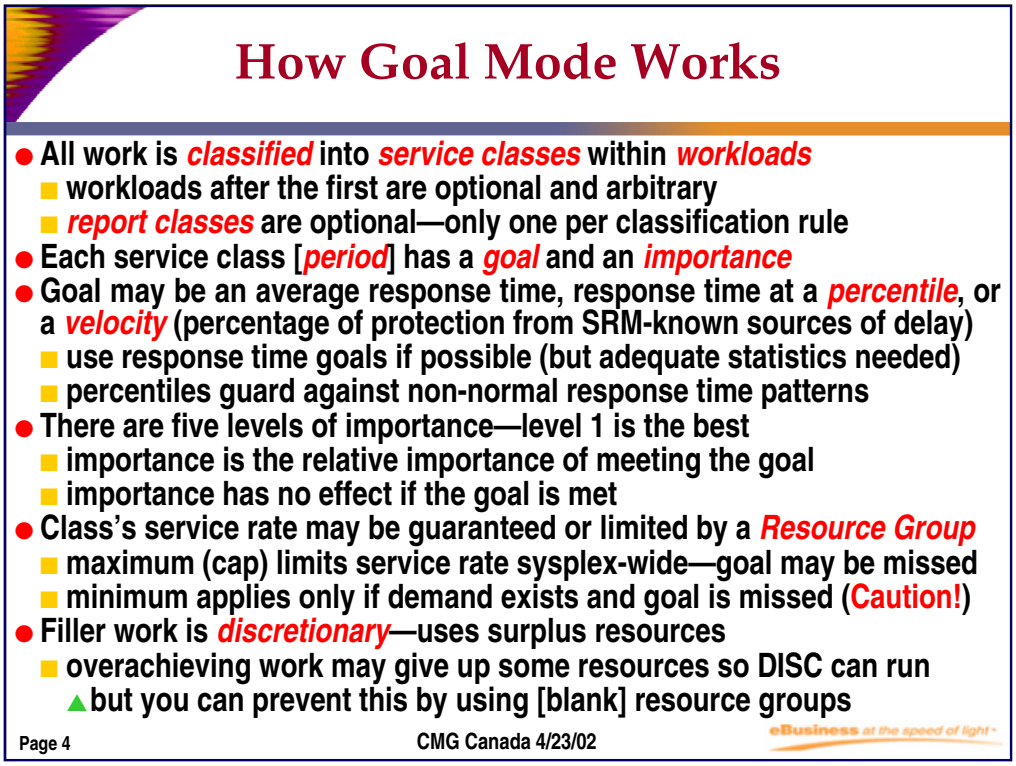
steve_samson@candle.com
steve.samson@attglobal.net

CMG Canada
April 24, 2002
Toronto, Ontario

Topics

- Goal mode vocabulary
- Approaches to conversion
- Pre-conversion preparation
- The IBM Goal Mode Migration Aid (GMMA)
- Detailed conversion steps
- Anatomy of a Service Definition
- Getting CICS to play nicely with others
- Summary

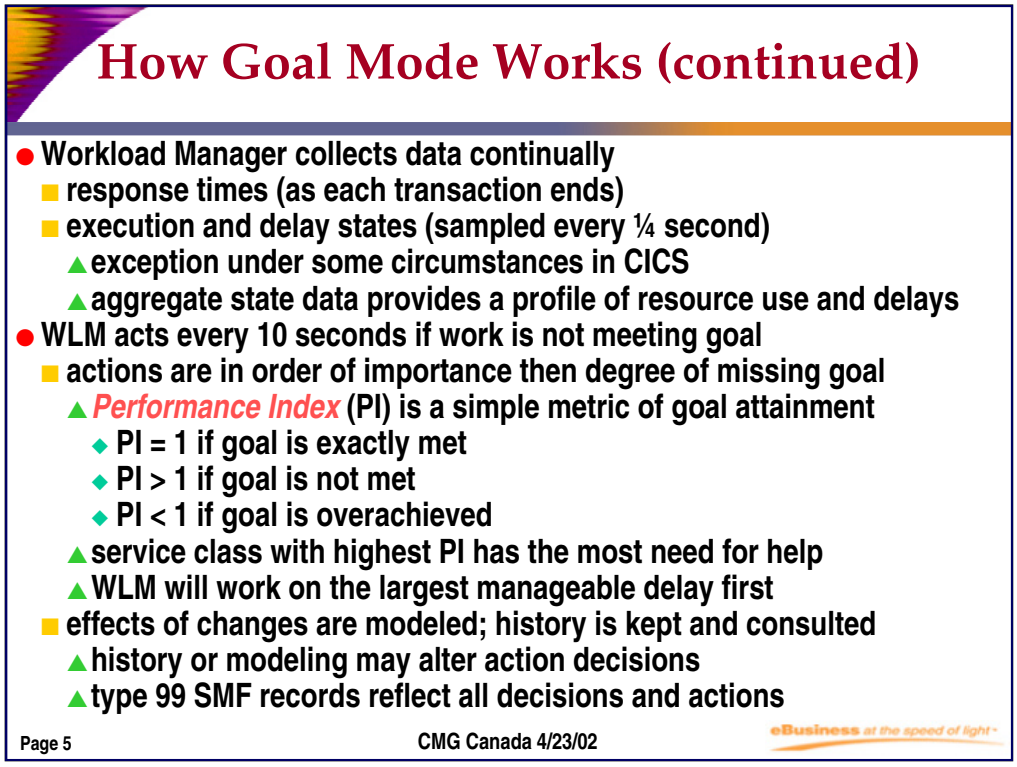
Goal Mode Vocabulary



Going to goal mode requires that all the work in the system (or sysplex) be classified through rules in an active service definition. Each work unit—job, TSO session, Unix transaction, individual CICS or IMS transaction, …—goes into exactly one service class and, optionally, one report class. The service classes are grouped into arbitrary reporting entities called workloads.

A portion of the service definition, the service policy, defines the attributes of each service class: the number of periods in the service class, the goal and importance in each period, and (optionally) the resource group to which it belongs. It is also possible to create and name additional service policies to apply to unusual combinations of workload and calendar. A new policy can be activated without changing the service definition. The service class definitions in the new policy replace [only] their counterparts in the running policy; hence the name override is applied to additional service policies.

Percentile response time goals are independent of response time distributions and are unaffected by outliers (transactions with atypical response times). Velocity goals are not really workload-oriented; in fact, they need to be adjusted after significant configuration or workload changes. A system that becomes heavily loaded may not be able to deliver the high velocities that were common in easier times. Lower velocities may be more realistic and may work as well. (How can we know if the difference between 35% delay and 45% delay can affect the performance of a workload in any measurable way?) On the other hand if a CPU is upgraded, 25% CPU delay may represent far more contention than it did on a less powerful system.



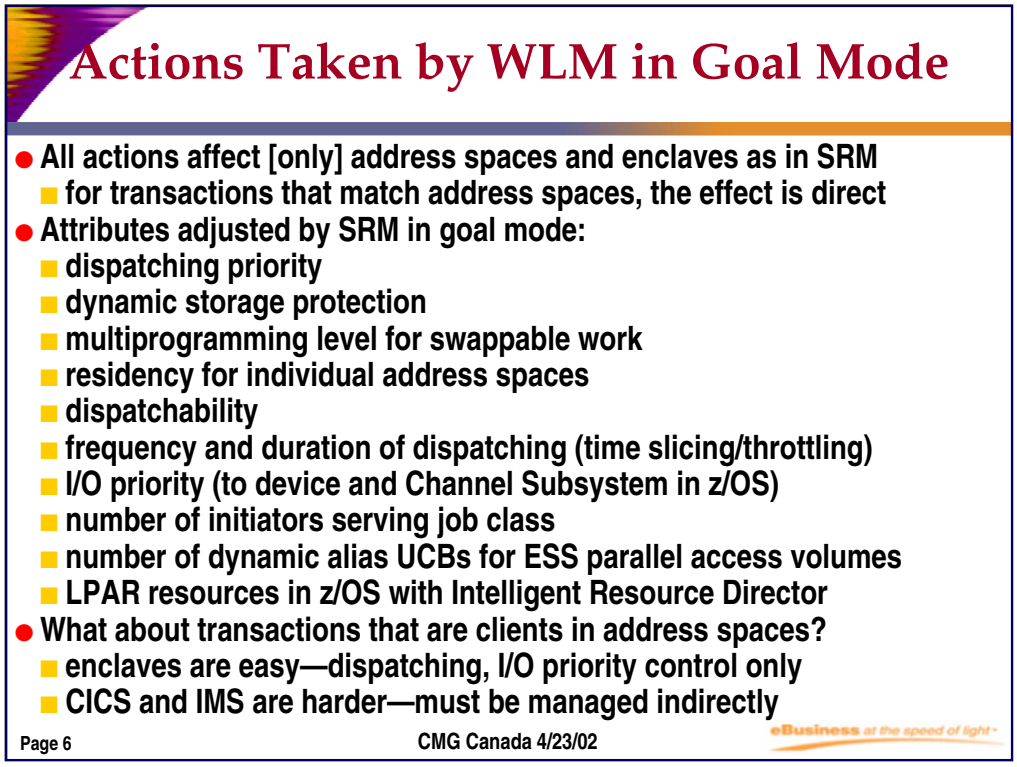
Workload Manager does what is needed to meet goals and satisfy resource group limits based on data that it collects and calculates. The first kind of data collected is response times of work that has response time goals. It comes from a variety of sources, either as communicated directly to the system (e.g. TSO/E via SYSEVENTs) or as explicitly communicated to WLM by the subsystem, as by CICS.

WLM also collects execution and delay state data by sampling. The ¼ second sampling interval would have been unthinkably short for a performance monitor in the past, but current CPU speeds and short path lengths add up to little WLM sampling overhead. What is gathered in such sampling leads to an execution and delay profile for each address space and enclave in the system.

Every 10 seconds, WLM checks to see if all goals are being met. If they are not, the service classes are ranked in descending order by Performance Index within importance level. From the top down, a service class will be helped if it can be helped.

If all service classes are meeting goal, those that are far overachieving their goals may become resource donors so that discretionary work may run.

Before taking action, WLM consults its history to see if it has already tried the action and whether it helped in the [recent] past. WLM also models the effect of a proposed change such as an MPL increase to ensure that the cost to the system is exceeded by the projected benefit.

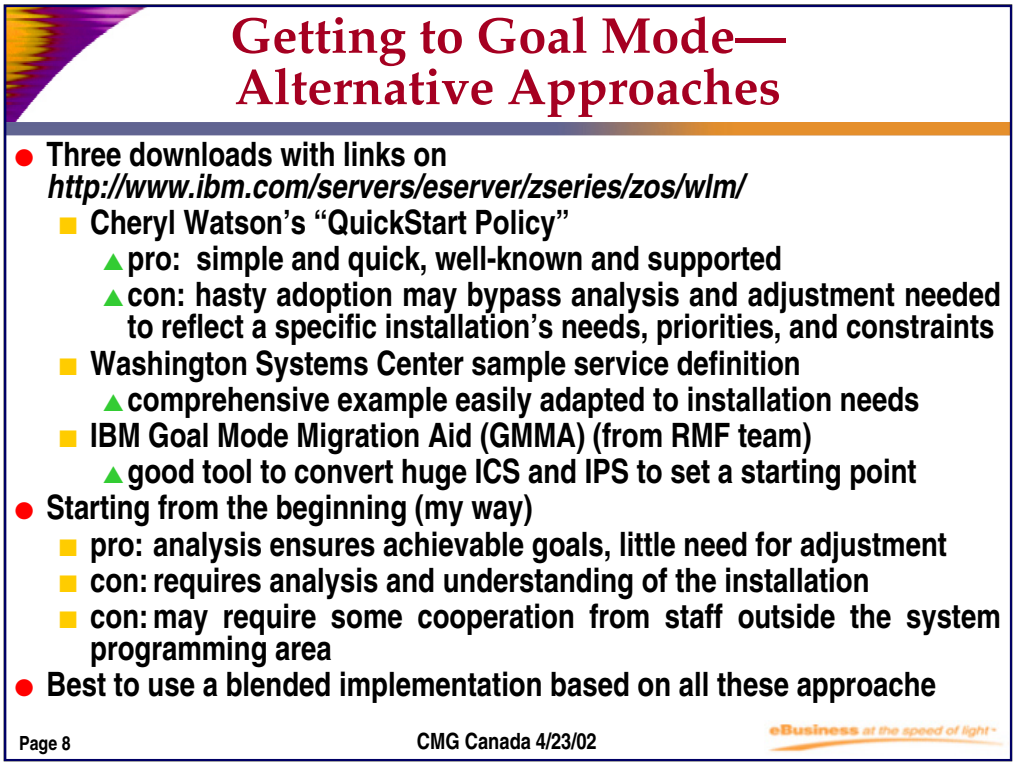


WLM in goal mode is a level of management above SRM as we have known it. It directs dynamically the settings of dispatching priority, I/O priority and storage protection (similar to storage isolation). Working set management is fully integrated and is far more effective than in compatibility mode.

Some new capabilities are added, such as rapid adjustment of dispatching priority to slow down address spaces in service classes that exceed their resource group service rate cap. Also new is the propagation of I/O priority to the device level with the Enterprise Storage Server (Shark).

Although goals may be set for transaction service classes, all that WLM/SRM can do is to adjust the attributes of address spaces—or of enclaves. Enclave transactions are known to WLM throughout their existence. CICS and IMS transactions are not directly visible to WLM; information is passed from their subsystems, which own the resources.

Approaches to Conversion



Several approaches have been developed for getting to goal mode. The first of these is Cheryl Watson's well-known "QuickStart Service Policy". If Cheryl's complete procedure is followed in detail, the results are usually favorable. Unfortunately, some try to do the job too quickly and simply copy the basic recommendations without analysis or particularization for the site. If you do choose to follow this approach, do the whole job.

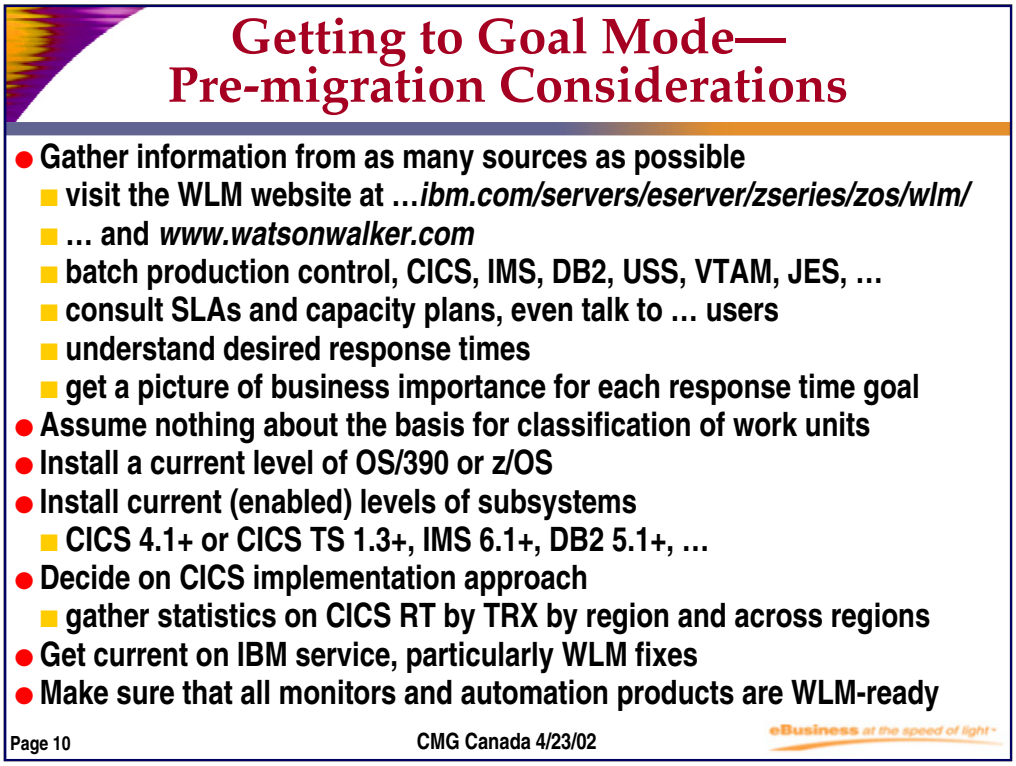
IBM has made an RMF-based conversion aid available. Based on the experience with an IPS conversion aid in the MVS/ESA 4.2 time frame, it may not be more than a starting point. The difference in approach between compatibility mode and goal mode makes little more than syntactically correct translation likely. Levels of importance and response time goals cannot be easily inferred from an IPS. However, the IBM tool is very useful where there is a very long and detailed ICS or IPS.

Another new IBM offering is a sample service definition from the Washington Systems Center. It may be downloaded from a link on the Workload Manager web page.

The approach recommended here starts from scratch and requires analysis before jumping into goal mode. As long as compatibility mode provides acceptable performance, a hurried conversion to goal mode makes no sense. Please take the time to do it right!

Of course, the four approaches are not mutually exclusive. Pick and choose the elements you like and come up with your own procedure for getting to goal mode.

Pre-conversion Preparation



While it may be possible to study the SRM parameter sets in SYS1.PARMLIB and associated data sets, and from that study infer required performance characteristics and their levels of business importance, why bother?

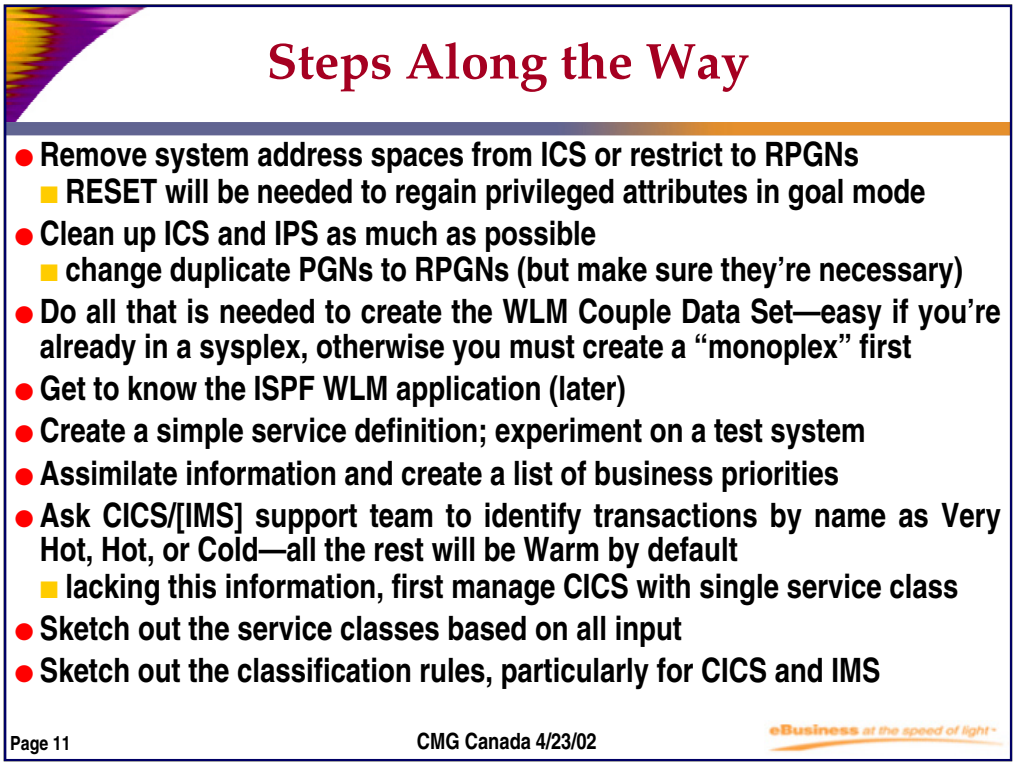
In many cases the people who put those parameters together are no longer accessible, and the underlying assumptions may no longer be current.

Instead (or in addition), go to the direct sources—those who are responsible for the applications and subsystems and find out from them what is expected and needed today. Even "current" service level agreements (SLAs) may not reflect current volumes, current performance, and current expectations. To a large extent, a service level agreement is only an approximate substitute for a real live service policy.

In this search for information, try to keep an open mind and accept what the users and their representatives provide as correct. For instance consider how DB2 Distributed Data Facility (DDF) transactions should be classified. The rules include no less than 11 possibilities including such arcane DB2-ish things as Correlation Information, and Collection, Package, Procedure, and Plan names. So what did one customer's DB2 wizard choose as the basis for classification? NET-ID! In other words, where the transaction originated is the most important characteristic. We implemented it that way, and it worked to everyone's satisfaction.

Many important WLM-related fixes are now available. OW37742 is crucial to anyone running a critical response time workload on a heavily loaded system. There was a PE against its PTF in the past; that should be remedied by now. OW 44517 is a recent APAR that explains a wide variety of bizarre symptoms. Make sure to get the PTF for it.

A final comment on readiness—make sure that fixes are current, and that support products are current. There are no excuses left.



Ever since MVS/ESA 5.1, a "system" address space loses its "high priority" attribute if it is placed by the ICS in a control performance group. Until OS/390 Version 2, that attribute could not be regained without an IPL. When a system with such a compromised address space is switched to goal mode, the address space may fall through the STC classification rules and be treated as a bottom feeder.

If a critical address space like CONSOLE is locked out as a result, two or more IPLs may be necessary to repair the damage—at least one to correct the ICS and another to come up clean.

As of OS/390 Version 2, the same thing happens, but it is possible (if CONSOLE is responding) to RESET an address space explicitly to the SYSTEM or SYSSTC service class. It is also possible to classify address spaces to the SYSTEM or SYSSTC service classes directly. (Only properly designated address spaces will actually go to SYSTEM.) A subsystem parameter (SPM) value of SYSTEM or SYSSTC in a classification rule can direct system address spaces to other service classes but this is a foolish and dangerous thing to do unless special circumstances apply. This facility can and should be used to assign report classes to system address spaces.

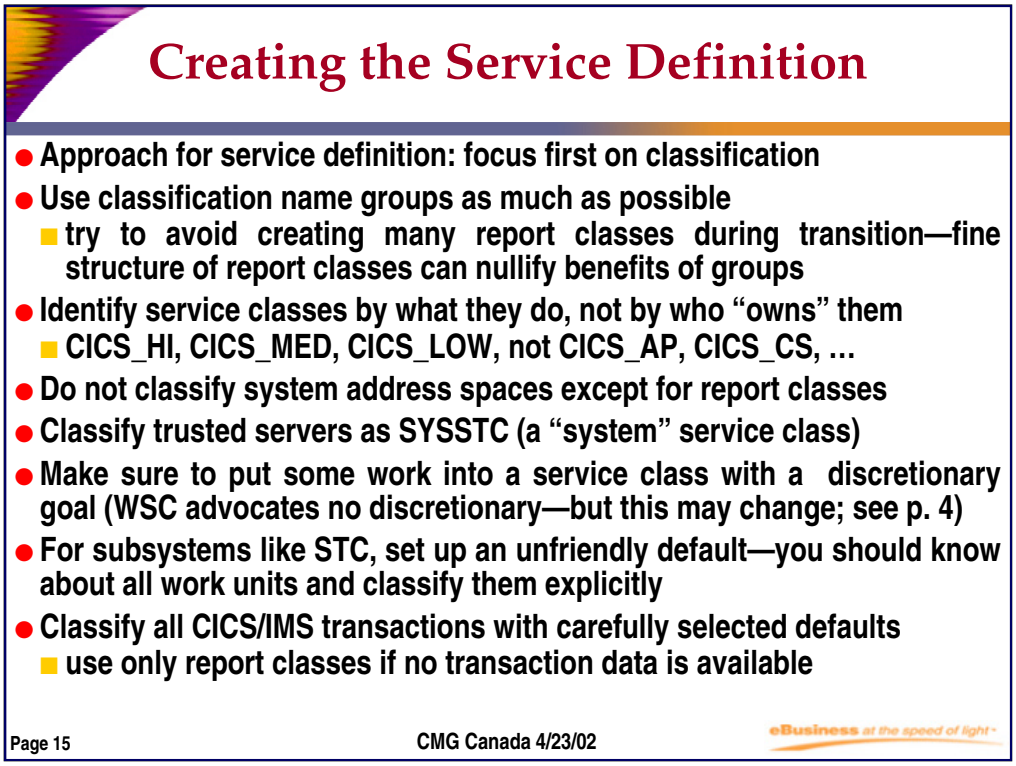
Although it is not something to do on Day One of goal mode, using WLM-managed initiators is worth exploring, especially in shops with a high volume of short routine batch jobs that arrive randomly throughout the day. Some have managed all their batch this way with good results and have achieved simplification of the operators' batch role.

The IBM Goal Mode Migration Aid (GMMA)

GMMA Highlights

- What is it?
  - a semi-automated tool for assisting goal mode migration
- What does it do?
  - parses ICS and IPS to create a skeleton service definition
  - integrates RMF data to set initial goals
  - integrates ICS and/or IPS comments to distinguish test/production *etc.*
- What doesn't it do?
  - create classification rules for non-ICS subsystems
  - create classification rules for CICS/IMS transactions
  - create meaningful names for classification groups or service classes
- Who should use it?
  - those with very detailed ICS rules—long lists of jobs or STCs
- Where is the work done?
  - on a PC using Excel with uploads and downloads
  - after the upload of the service definition, use of GMMA is finished

Detailed Conversion Steps

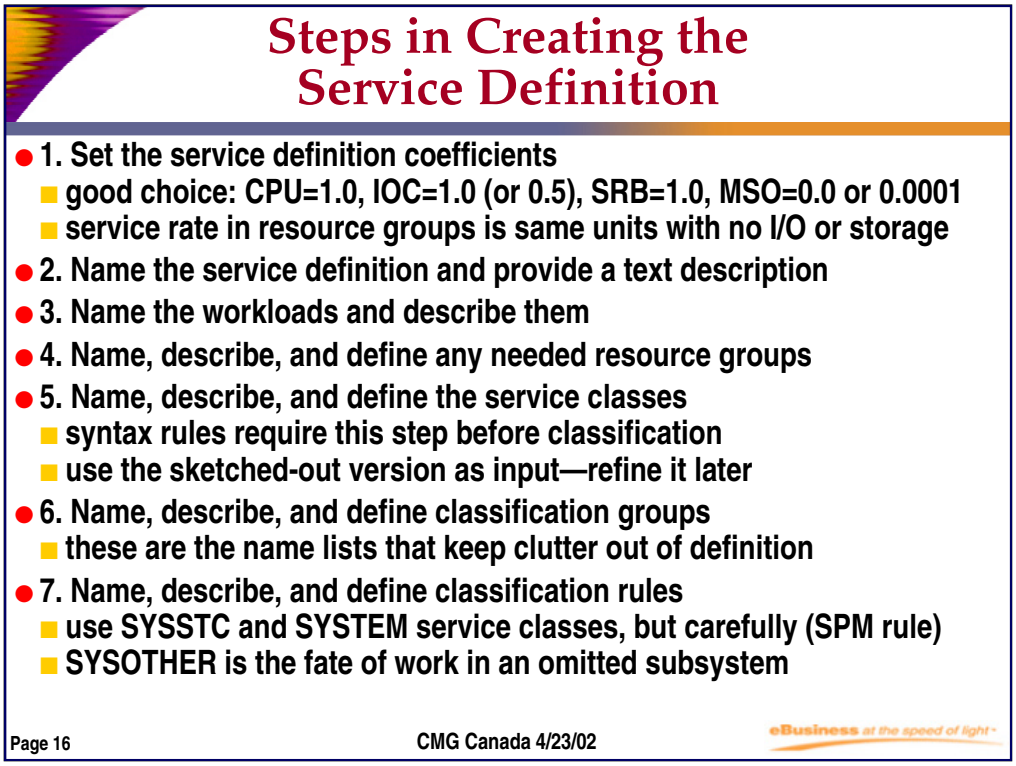


The service definition is mostly about classification, but it also includes a base service policy—the definition of service classes with their goals. It might be reasonable to say, "OK, I'll do classification first and then turn to the goals." However, the syntax rules of IWMARIN0 to not permit this order of definition.

The ISPF application seems to have been written by someone who learned to write one-pass compilers, in which all objects must be defined before they are used. Therefore classification comes last, specifically after the service classes are defined, and you are forced to define the service classes before you write the rules that direct work to them. You should have enough defined in your sketched version to make a start at this point—refine it later.

There has been some confusion about classification for CICS. CICS address spaces are classified in the subsystem in which they run, to service classes with velocity goals. CICS transactions are classified in the CICS subsystem, to service classes with response time goals. The address space velocity goals are ignored when the address spaces are acting as servers to transactions;.

The advice about not classifying system address spaces is absolute for Version 1 of OS/390. For Version 2, it may be relaxed if the system address spaces are classified explicitly as SYSTEM or SYSSTC as appropriate.



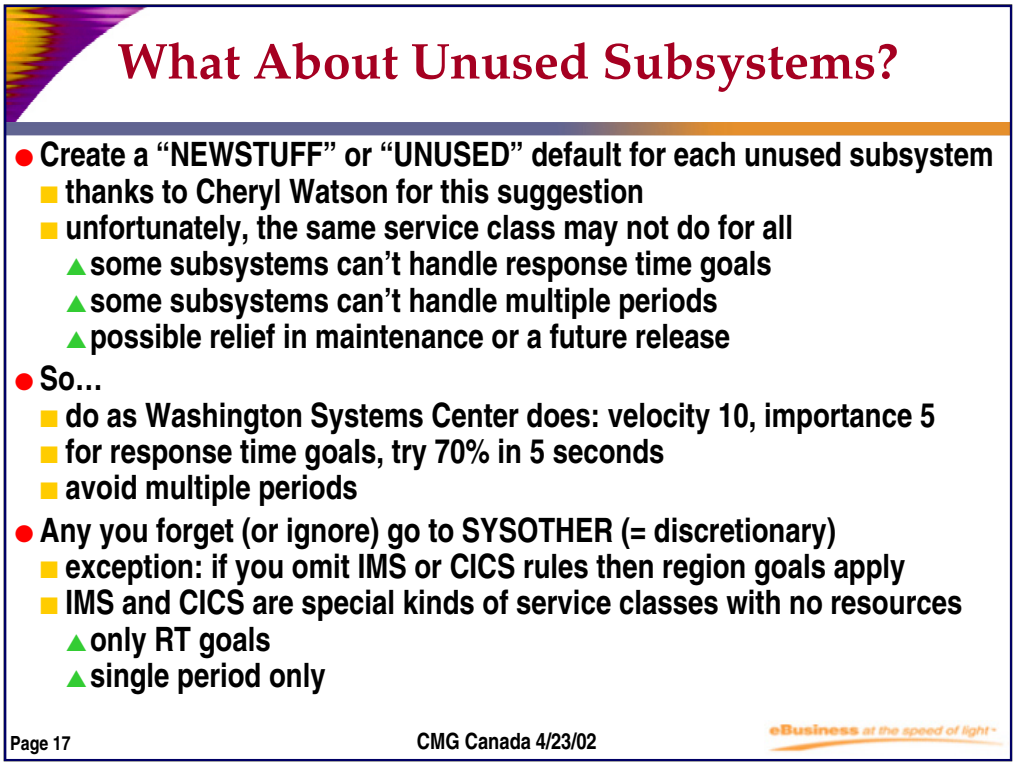
This slide shows the required order of proceeding through the ISPF application (IWMARIN0). The first order of business is to set the service definition coefficients (SDCs). The simplest setup recognizes that the service rate specified for resource groups is raw unweighted CPU service—therefore to put other service rates and service accumulations on the same footing requires CPU and SRB coefficients of 1.0. As has been demonstrated elsewhere, MSO is a flawed metric and should be eliminated from the service formula; thus it should be set to 0.0. (Some still cling to the idea that MSO is worth something and shouldn't be totally ignored.) I've seen recommendations for IOC as either 1.0 or 0.5; either should do based on consistency with prior practice.

The next two steps are routine. Step 4 allows you to define resource groups, most commonly to limit service to a set of service classes. Step 5 requires full specification of each service class. If you're not quite ready to set all goals at this point, leave some or all service classes as DISCRETIONARY until you've pondered classification.

Step 6 allows you to offload complexity (or at least volume) in the classification rules by creating lists (groups) of object names that will be referenced in classification. Groups can be defined for eight different kinds of objects, including transaction name, transaction class, and user-id.

The final step in the definition process is classification itself, in which each work unit that becomes known to WLM is placed in a service class and (optionally) a report class. (The report classes do not need to be defined beyond their appearance in the classification rule.)

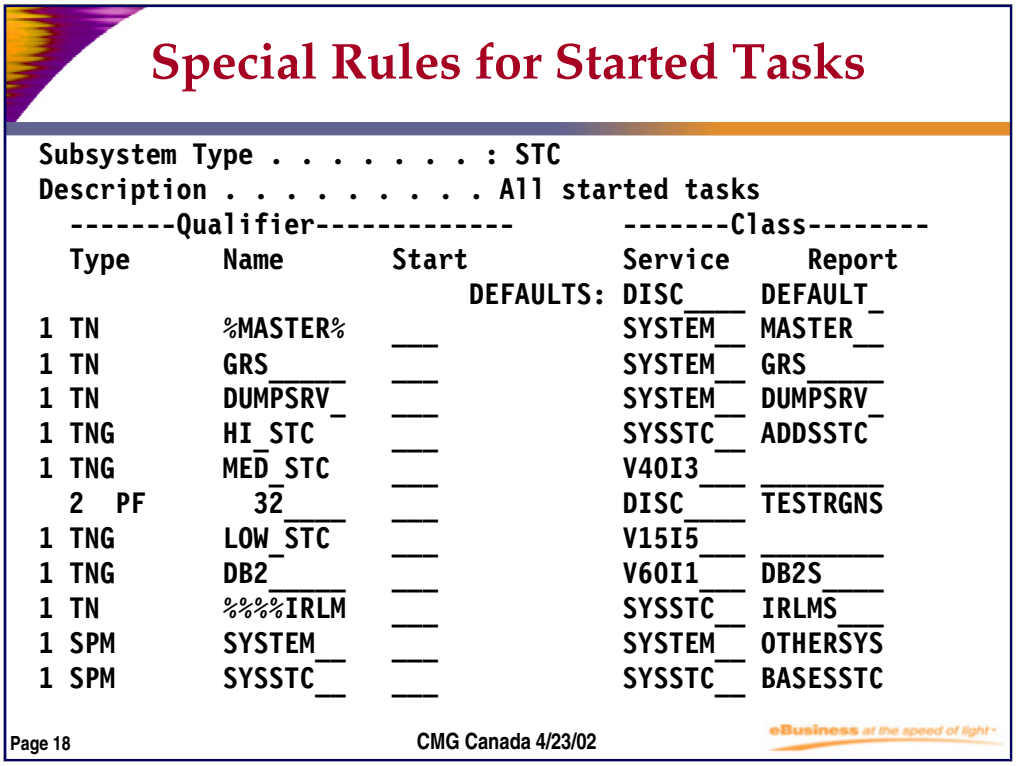
After the definitions are complete, they may be refined. Establish a change control discipline of not editing the active service definition—download it to a new name first.



One of the hallmarks of effective performance management is resilience—a system that is prepared for the unexpected. Preparing WLM to deal fairly with work it hasn't seen before is not difficult. For the current release, see which subsystems (types of work managers) are supported. For each of those that is not a part of your current workload, create a one-rule classification entry that directs the work to a non-hostile service class—i.e. not SYSOTHER. When someone starts using Component Broker or MQ, the transactions will be treated reasonably well.

Then, each time a new release is installed, just check for any additional types of work. The originally proposed service class, as suggested by Cheryl Watson, was NEWSTUFF, something like the TSO service class illustrated on page 20. (The service class name is irrelevant, except that it can't begin with "SYS".) Unfortunately, different kinds of work cannot all use the same service class. The MVS Planning: Workload Management publication spells out which kinds of goals go with which kinds of work.

In the Washington System Center sample service definition, subsystems of no current interest all point to the same default service class, UNCLASS, a single period service class with velocity 10 and importance 5.



The STC subsystem is crucial to successful performance management in OS/390 or z/OS, so it's worth some time to consider a few sample rules. Note that this example applies to OS/390 Version 2 and later.

As usual, we begin with a default. (That is not possible in Version 1.) Any STC that is not explicitly classified by one of the rules goes to a service class called DISC, presumably with a discretionary goal. The first three explicit rules are there to define separate report classes for three key system address spaces. Next is a TNG to enumerate the installation's additional choices for SYSSTC, followed by another TNG to list the STCs that go to a service class called V40I3, suggesting a velocity of 40 and an importance of 3. All rules to this point have been first-level rules; now there is a second-level rule defining exceptions to the preceding level 1. Those STCs on the MED_STC list but assigned to PERFORM=32 via JOB or START go to DISC. The report class assigned suggest that these are test [CICS] regions. The next two TNG rules are unremarkable, and the TN rule assigns STCs named as ????IRLM to SYSSTC.

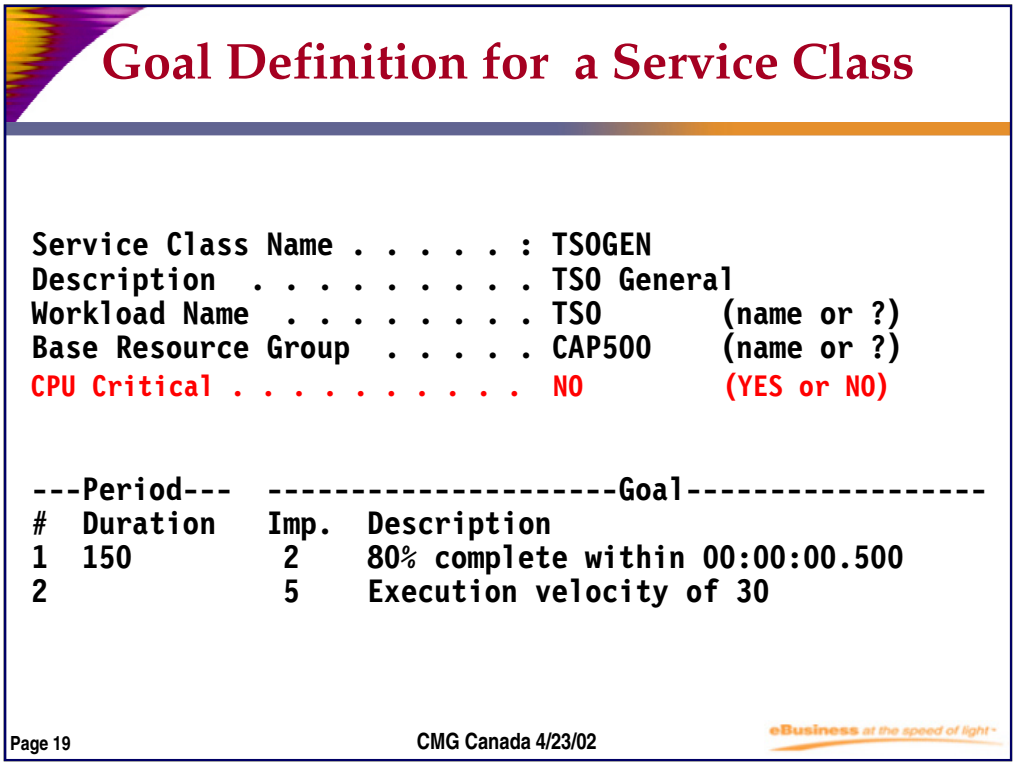
Finally, two SPM rules ensure that system tasks are classified properly. The first SPM (subsystem parameter) rule says that "anything else that is known to the system as belonging to the SYSTEM service class goes there." The second SPM rule puts in the SYSSTC service class all those address spaces that are system-defined as going there. The two SPM rules are necessary to eliminate the need to name all the system address spaces in a TNG (transaction name group) so that they do not go to the low-priority default service class. (If a name were to be omitted from such a list it would go to DISC.)

All else goes to the default of DISC.

For a version 1 system, the default service class is left blank, the SPM rules are not allowed, and a final rule of

"1 TN * … DISC DEFAULT"

has an effect similar to that of the default in the Version 2+ environment. In either case, any STC showing up in report class DEFAULT should be investigated and properly classified.



Here is a simple service class definition for a general TSO service class. We take advantage of percentile response time goals to simplify TSO down to only two periods. If you have very well-defined special circumstances, one or more added periods may be needed. My advice—start simple.

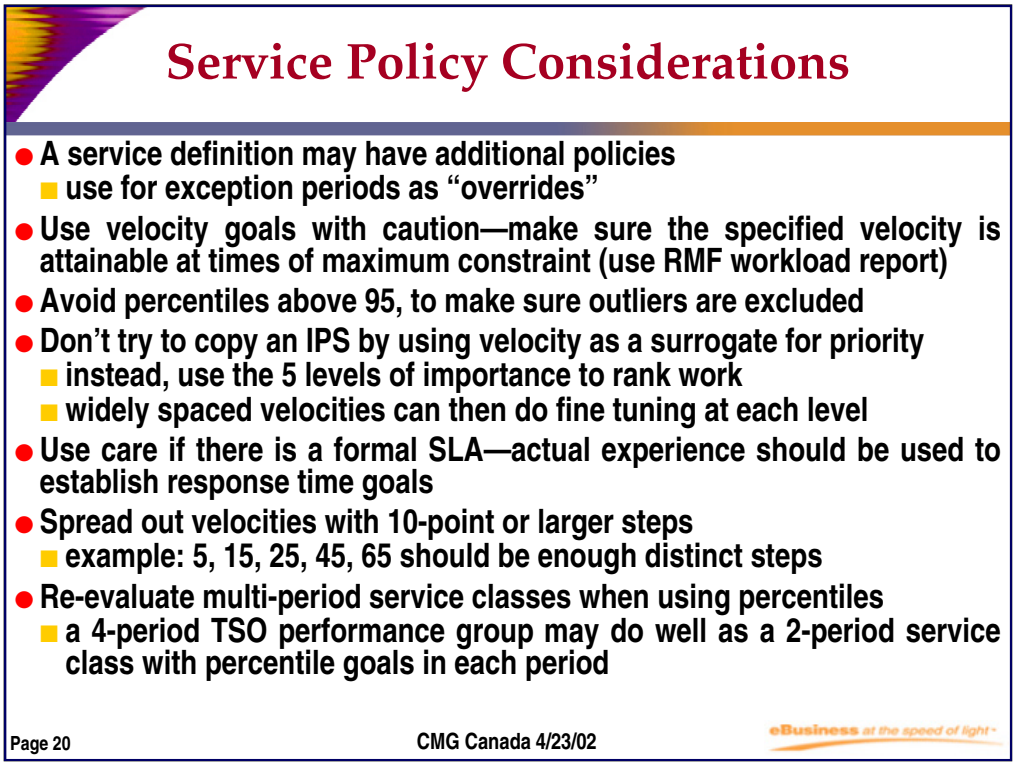
For purposes of illustration, this service class is shown as belonging to a resource group, RGCAP5K. Taking the name literally, this resource group would be defined with a minimum service rate of 0 and a maximum of 5000 service units per second—about one full processor of a G5. If service class TSOGEN and others in the resource group exceed 5000 unweighted CPU service units per second across the sysplex, address spaces in the service class will be "throttled" to stay within the cap, even if goals are missed. "Throttling" is accomplished by rapidly alternating the address spaces between dispatchable and non-dispatchable states.

The service class is not denoted as CPU Critical. Indeed, it cannot be since it is not a single-period class.

Note the durations—they are lower than was typical in the past. This is because the Service Definition Coefficients are lower, as explained on page 17.

The goal for the first period is percentile response time. Excluding 20% of the first period completions means that, for example, a few trivial transactions delayed by synchronous data set recalls will not contribute to an inflated measured response time.

Second period has a velocity goal of 30 percent at importance 5. A discretionary goal here would be acceptable if long-running TSO transactions are less important to this installation, or if, for some reason, they are to be discouraged.



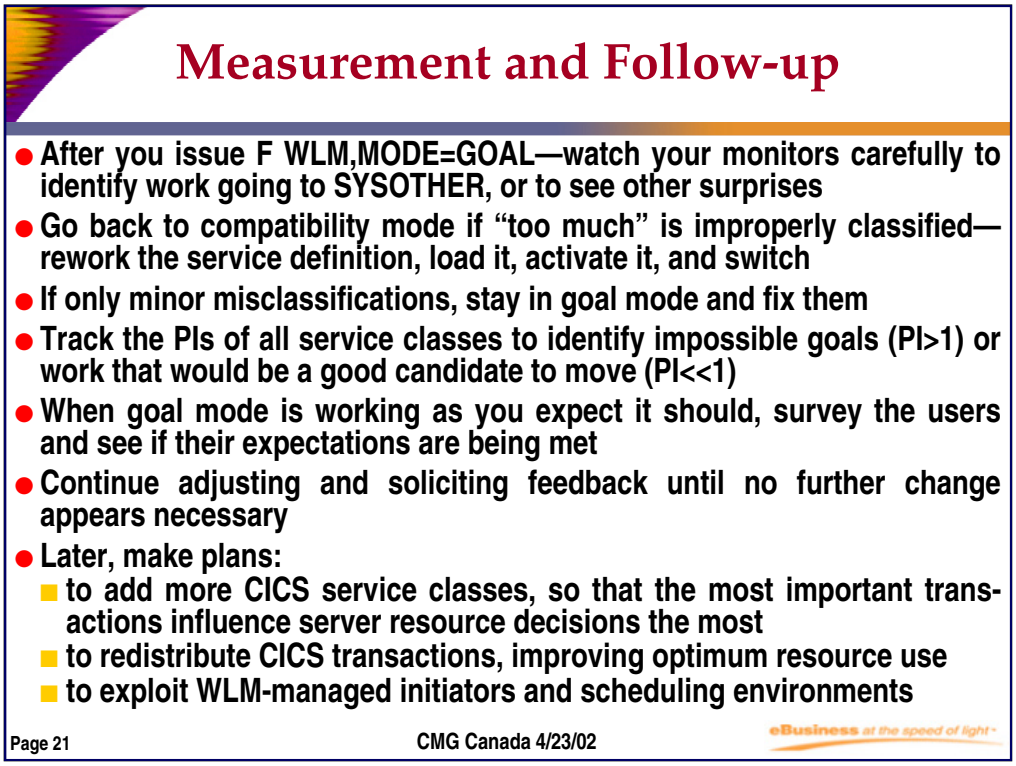
A service policy is a set of service class definitions. Included in the service definition is a base service policy, but additional policies may be defined and then invoked as overrides to the current active policy.

When setting a velocity goal based on prior experience, remember that any velocity you choose based on a time of satisfactory service becomes a lower limit in goal mode. Anything lower than the goal velocity will show up as a missed goal. Therefore make sure to choose a suitably low goal velocity—the minimum observed during a period of good service, perhaps reduced by a few percent as a safety factor.

Percentile goals are designed to eliminate the effect of anomalously long response times on resource management. Going beyond 95% might start including some extreme values, thus defeating the purpose of percentiles.

Perhaps the surest way to fail in goal mode is to to try to map compatibility mode dispatching priorities to velocities. There is a very weak correlation between velocity and dispatching priority. Importance is a much better match to priority, but it would be far better to analyze what the IPS implies in terms of business importance and expected performance and leave the literal details behind.

Service level agreements (SLAs) can be a good guide to goal-setting—provided that actual experience and expectations are close to the SLA values. When in doubt, go by experience.

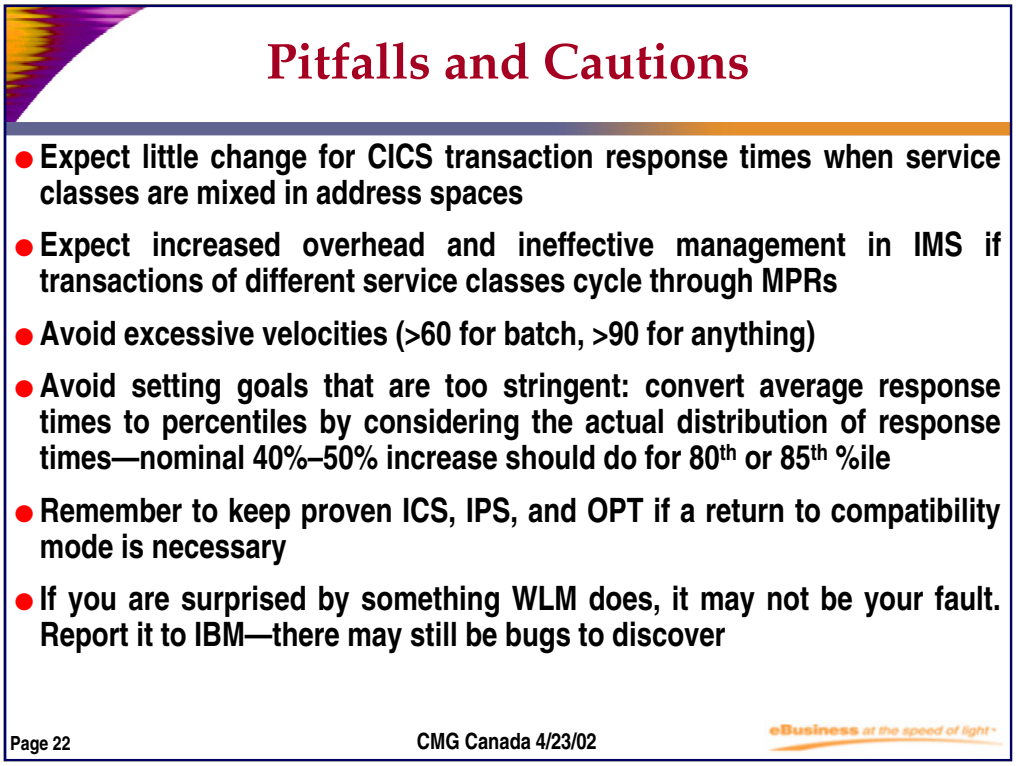


The best switch to goal mode is the one that is unnoticed. However, the person responsible for going to goal mode should watch closely the first few times. At a minimum, make sure that no system address spaces are misclassified, and that there is no flood of work units going to SYSOTHER. After a few minutes in goal mode, if there is more than about five minutes' work needed to correct minor misclassifications or overly ambitious goals, switch back. Doing this should be planned, and it is not a sign of failure. Correct the errors and switch back to goal mode. Now you should start to regard goal mode as established and be reluctant to go back. The first switch out of goal mode should be the last unless something nasty happens.

The basic monitoring technique should be to watch performance indices. A PI greater than 1.0 means that a goal can't be met. PIs much less than 1.0 mean that a goal is too easy to meet and that WLM can't take resources away from that service class. In turn, that usually identifies an opportunity to move a service class and its transactions out of a CICS address space that is dominated by high importance work.

Survey your users and make sure they have not observed a negative change in performance.

If you do observe overachieving CICS service classes, create a plan to redistribute transactions and thus make the work in each server address space more homogeneous. Once you are safe and secure in goal mode, make plans to implement WLM-managed initiators. You should also plan at the same time to implement Resource Affinity Scheduling so that jobs will run where you want them to.



Based on reported experience, the most likely goal mode failures occur on very heavily loaded systems with very important transactions that require very short and consistent response times. On such a system, make sure that the load is carefully monitored to make sure that the goals can be attained at "peak of peak" times. Goal mode does not supersede capacity planning. The fix to APAR OW37742 should help with goal attainment for such critical workloads. Make sure that all HIPER WLM fixes are installed. As of V2R10 with the enabling PTFs, such work might be a candidate for CPU Critical or Storage Critical designation.

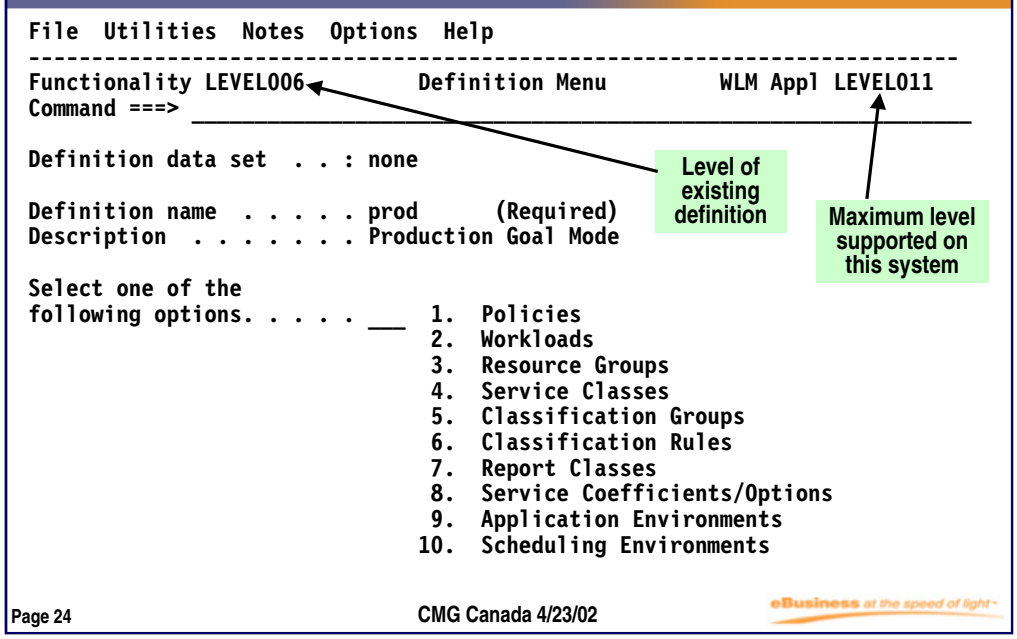
CICS transaction response times are measured and visible (as service classes) in goal mode. To the extent that the more important service classes dominate the activity in server address spaces, their response times will be managed through the management of resources directed to the server address spaces. The more mixed the population of the CICS regions, the less effective will be the management.

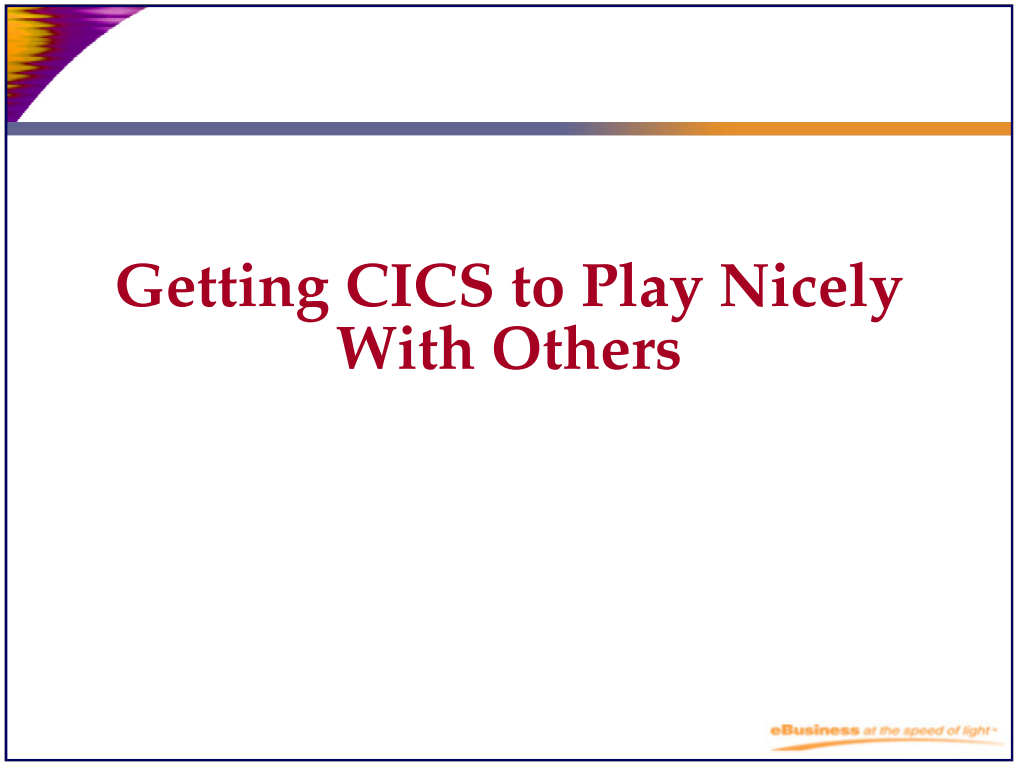
In IMS, the effect of mixing transaction service classes in MPRs is the need to wait for adjustment of dispatching priorities when there are transitions from one service class to another as transactions come and go.

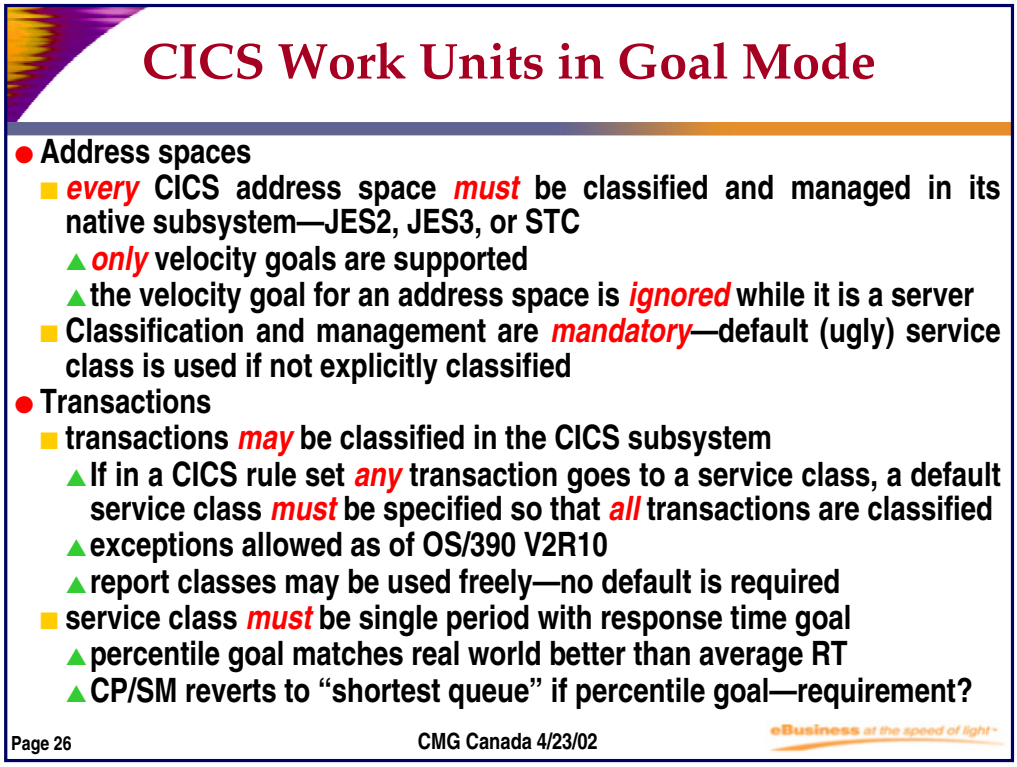
When moving from observed average response times to percentile goals, the same response time number at say, the 85th percentile, is a more stringent goal than that response time as an average response time goal. The response time number should be increased by 40% to 50% unless specific knowledge of the response time distribution is available and indicates otherwise.

Anatomy of a Service Definition

Service Definition Elements (as seen in IWMARIN0)




Getting CICS to Play Nicely
With Others
eBusiness at the speed of light

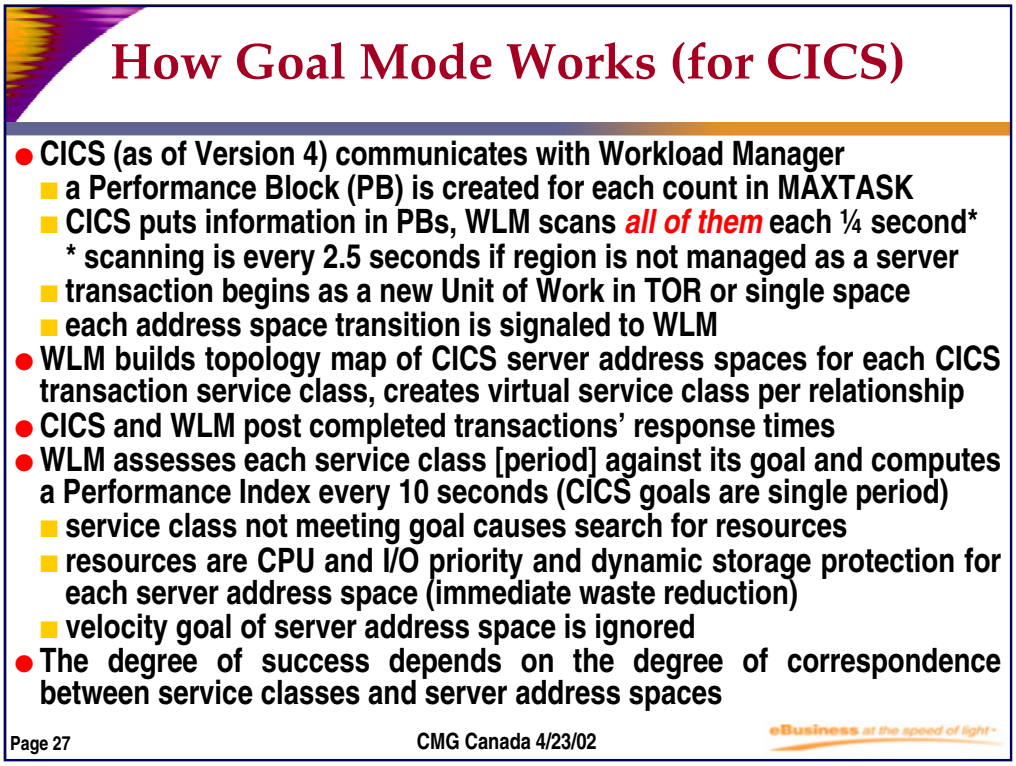


I see many queries based on a misunderstanding of CICS work units and classification restrictions. Let's consider some WLM differences between address spaces and transactions:

Attribute	Address space	Transaction
Classification needed	Mandatory	Optional
Subsystem in which classified	STC or JES	CICS
Default SC	Optional in subsystem	Optional if no other SC
Report classes	Optional	Optional
Consequences of omission	Ugly default SC	No server management
Goal types permitted	velocity	1-period response time (avg. or percentile)
Resource Group	allowed	ignored
Owns resources?	yes	no
Response time reporting?	no	yes
Resource use reporting?	yes	no

Another frequent area of confusion is that "subsystem instances" apply to originating address spaces only (usually TORs) and use the VTAM appl-id in the rule.

Many of the difficulties may be overcome through the judicious exclusion of selected server address spaces from server management.

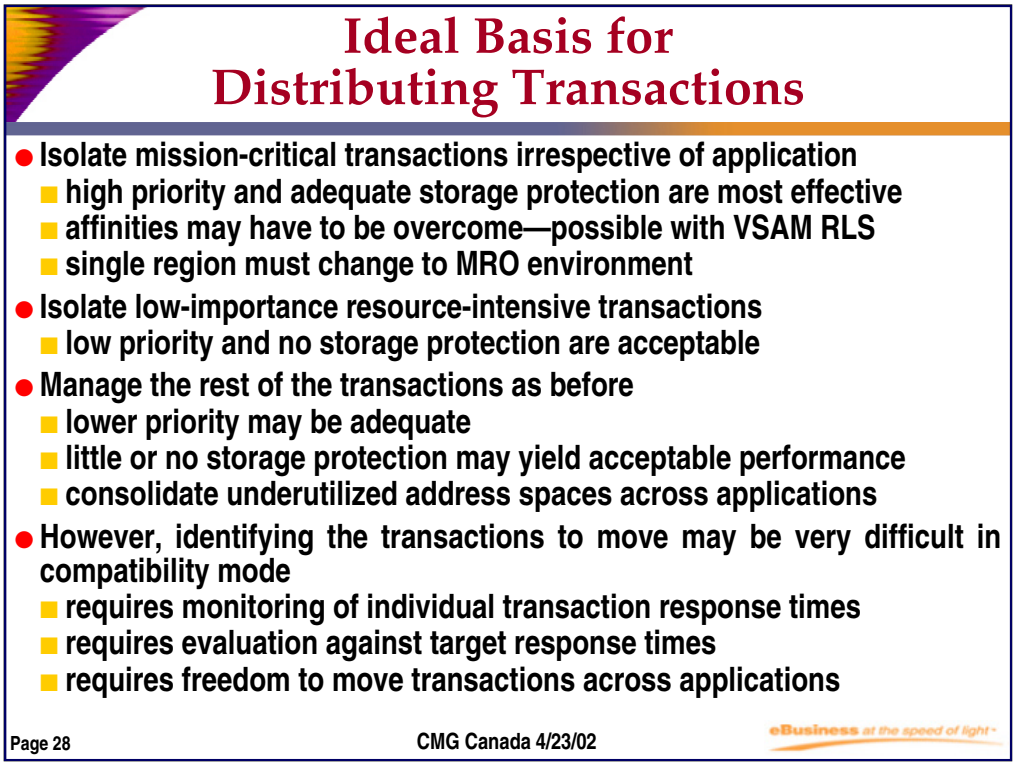


The adjustments made by WLM (SRM) affect only address spaces and enclaves. The intent of adjusting the dispatching priority, I/O priority, and storage protection for server address spaces (as for CICS) is to influence the performance of the served service classes so that they meet their goals.

A server address space is classified in the subsystem in which it starts, to a service class with a velocity goal. CICS transactions are classified in the CICS subsystem, to service classes with response time goals. Understanding this distinction is very important to success with CICS in goal mode.

When WLM goes through its analysis cycle it creates virtual service classes for the interactions of transaction service classes with their respective server address spaces. Evaluation is done for the transaction service classes but the resources live, and are doled out, in the address spaces. The velocity goal of a CICS server space is ignored once it takes on the role of server.

The key "CICS problem" with goal mode is that the relationship between transaction service classes and server address spaces is ill-defined. The association of transactions with address spaces is a consequence of application implementation decisions by those who were remote in time and knowledge from those implementing goal mode. If the transactions within an address space are homogeneous in execution profile, response time, and importance, they should be in the same service class. If there is a one-to-one correspondence between transaction service classes and server address spaces (AORs) then the performance of the transactions in a service class will be very well controlled by WLM. To the extent that that is not the case, less important transactions will get a free ride and tie up resources that more important work outside of CICS should be able to use.



The ideal is easy to infer from the available information. Assuming we had the freedom to move transactions from region to region, we'd put the most important, response-critical transactions from all applications in one or more dedicated regions, so that the need for high priority and storage protection could be focused on just those transactions that merit such coddling. We might also consider moving the very unimportant transactions that are significant resource consumers to one or more regions of their own, so that they might execute at low priority with no protection.

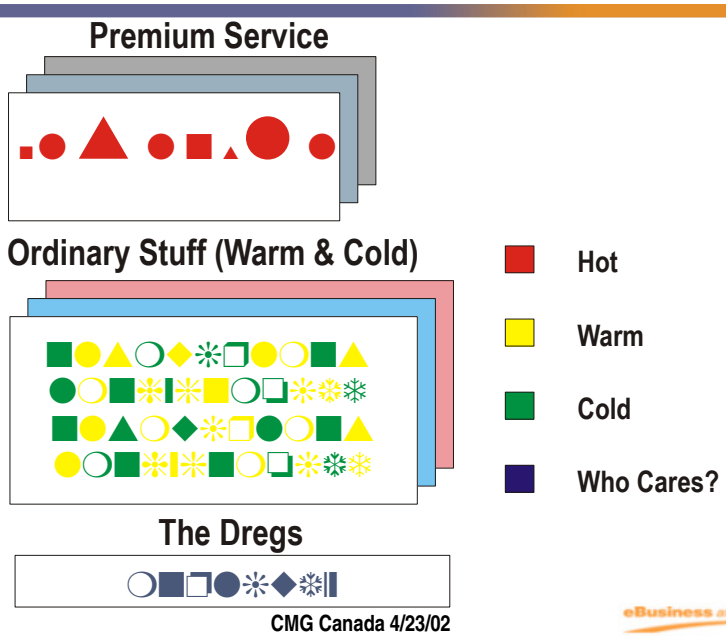
The vast majority of transactions could be left as is, but the now more homogeneous regions could run at lower priorities than before, with less (or no) protection.

Once the transactions had been rearranged, the original regions could be consolidated and reduced, again combining transactions across applications.

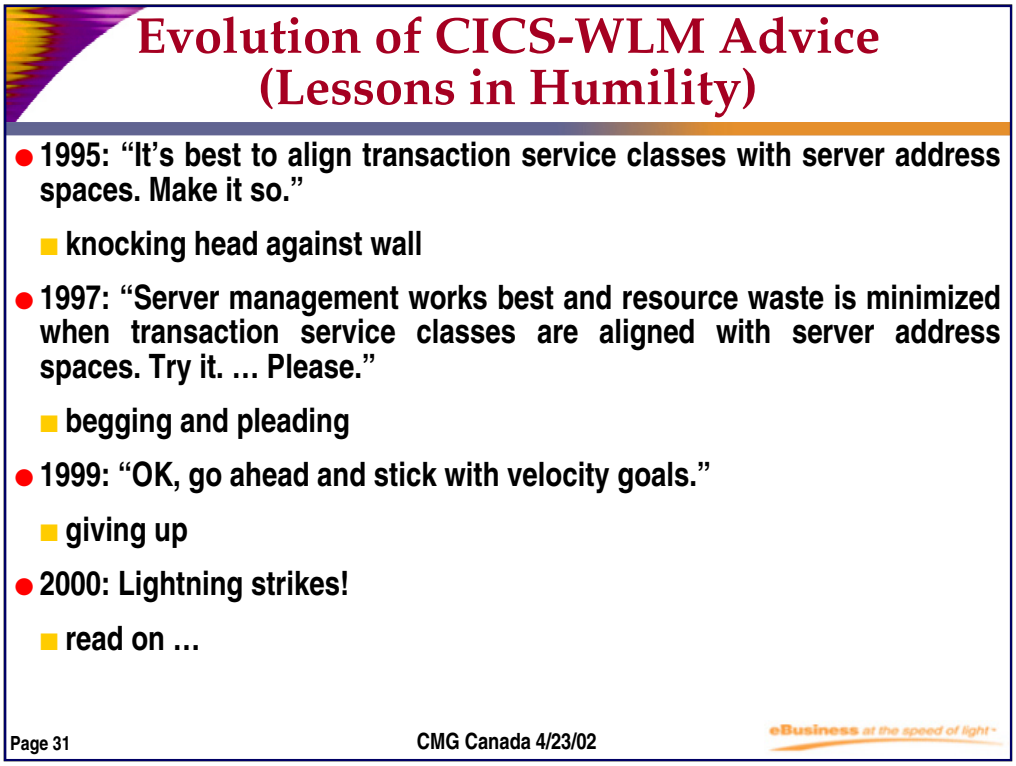
It sounds so easy but it isn't. There are two big problems. The first is securing the freedom to rearrange transactions across regions. To do this requires identifying and eliminating affinities and other interdependencies across transactions in each region. There is also the little matter of identifying the transactions that might be candidates to be moved. A great deal of detailed transaction performance data is needed; it is very difficult to obtain in compatibility mode, and even more difficult to analyze because of the sheer number of objects to consider.

Goal mode changes the picture. All transactions (there may be thousands) must be classified into a small number of service classes. It is far easier to measure and track the performance of a few service classes than of a multitude of transactions.

Suggested Arrangement



Dream On!



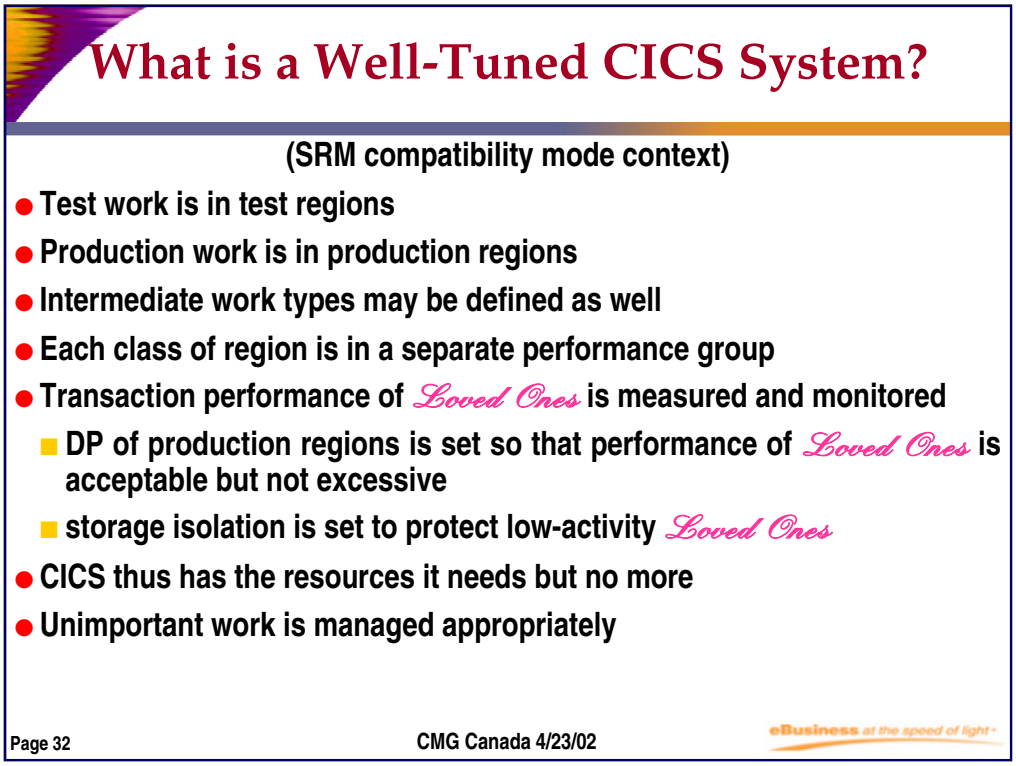
Unfortunately, goal mode does not change the picture enough. The mere availability of service classes does not serve to get the transactions into them, much less rearrange the transactions across regions.

It was easy in 1995 when nobody was in goal mode to suggest that doing it right had a prerequisite of rearranging the CICS transactions. Nobody paid attention. People started migrating to goal mode, but without defining CICS transaction service classes or classifying transactions.

A couple of years later I had created more and more justification and a methodology that could get CICS into the desired configuration with minimal effort. Again, it didn't work.

Time marched on. Still, very few were in goal mode with CICS transaction goals and virtually nobody went to the effort of moving transactions to different regions. The de facto standard was to run the CICS regions with velocity goals and ignore the transactions.

Finally, this year, I had a flash of insight. Read on.

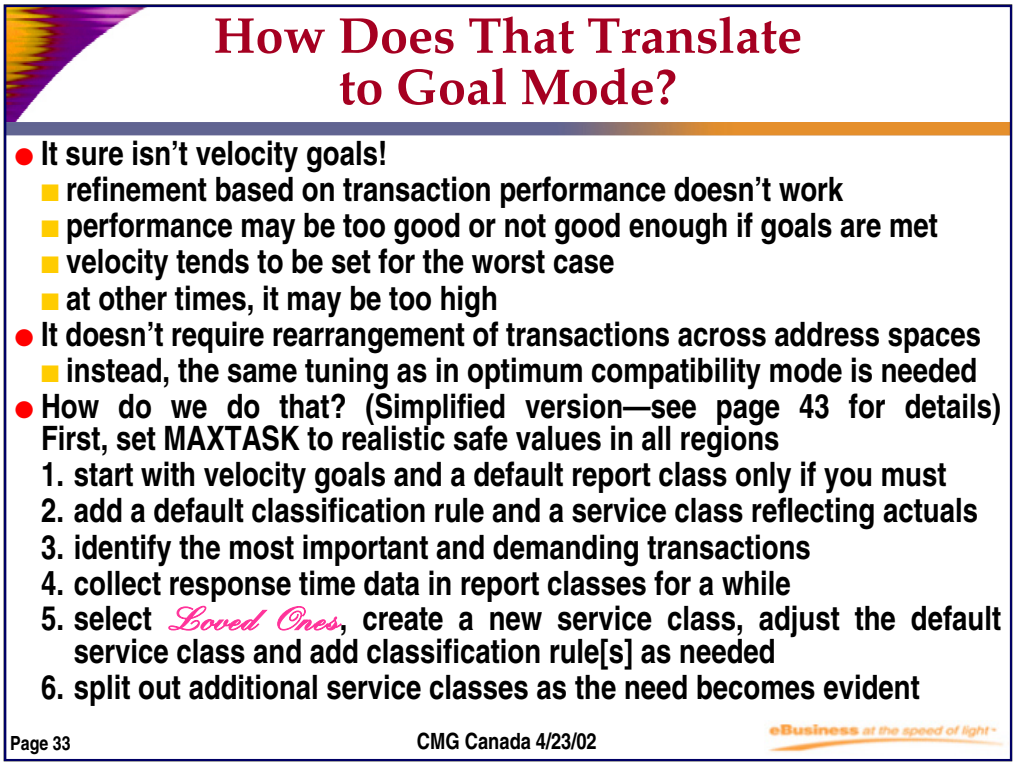


To understand the new approach, it's important to see what is regarded as "good enough" today. If compatibility mode isn't good enough for CICS, it's a problem with few symptoms.

The essence of a well-tuned CICS system in compatibility mode is the appropriate application of resources to the CICS address space. It is guided by the performance of the most critical transactions. The results achieved are:

- each kind of CICS work (production, pilot, test,…) is in a separate performance group
- for production work, the dispatching priority and storage isolation of its performance group are set so that the key transactions achieve their response time targets consistently
- for other classes of CICS work, sufficient resources go to the address spaces so they achieve adequate performance without delaying other workloads

In short, CICS has the resources it needs and no more.

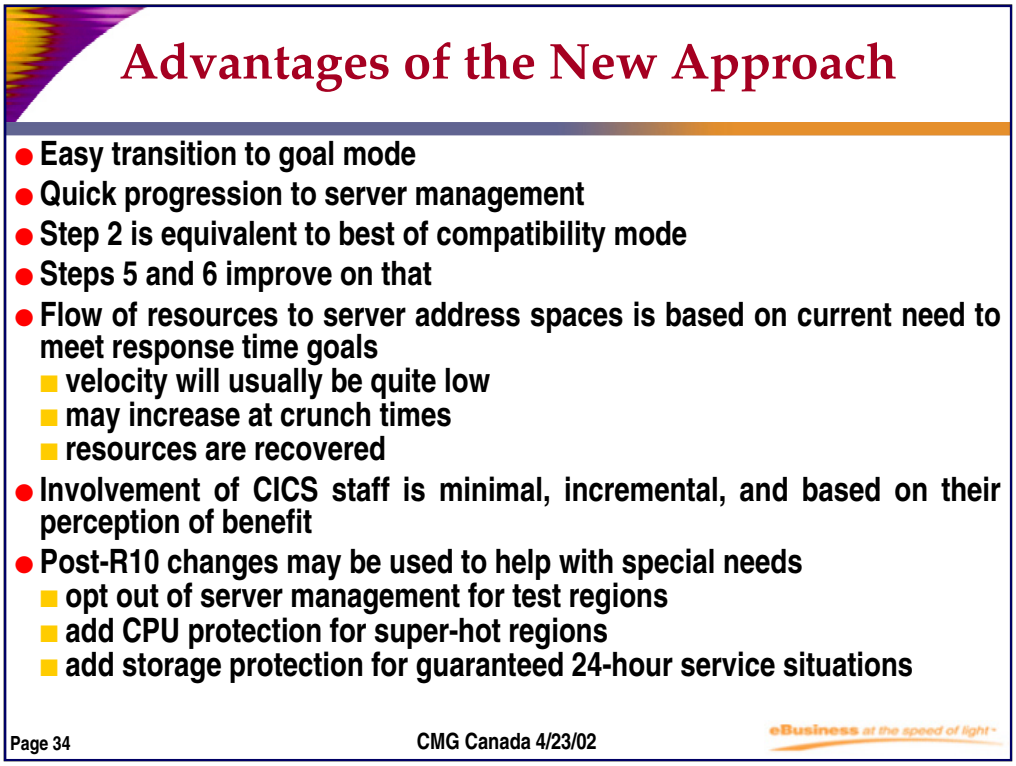


To take CICS from success in compatibility mode to a first success in goal mode, we need to consider what actions in goal mode mirror the best practice in compatibility mode. Let us first recognize that there is no goal mode specification equivalent to a fixed dispatching priority and either fixed or floating storage isolation. The closest we can get is a velocity goal at an appropriate level of importance. However, it's not the same. The priority and protection of a given velocity can vary all over the map depending on workload behavior and contention from other workloads. At some times the DP may be higher than the compat mode fixed priority, and it may be lower at other times.

For initial goal mode success, it's also not necessary to move transactions across regions. They were happy where they were in compat mode; just leave them there until success is solid.

So, how to start? I recommend setting the velocity goals and importance levels of the CICS regions for adequate performance of startup and shutdown. Add the CICS subsystem to the classification rules and specify a default report class to gather response time data. You can then add a default service class with a goal representative of good response times. If the typically experienced good response time is 1.5 seconds, a goal of 1.5 seconds at the 85th percentile will do. Set the importance of the CICS service class to match the business importance of the most important transactions. Most likely, an importance of 2 is a good starting point for a production environment. To avoid excess overhead, make sure the MAXTASK parameter in CICS is only slightly above what is needed.

You have now achieved goal mode with server management, matching and even surpassing the best practice in compatibility mode. Now build on the success by adding report classes and then service classes so that the influence of CICS transactions on server resource management is responsive to performance of transactions within levels of importance. When you add a service class for the loved ones, make sure to adjust the importance of the default service class to reflect the absence of the most important transactions.



This approach can get CICS to run in goal mode as well as it did in compatibility mode using WLM's server management capabilities. The only input required from the CICS staff is information on which address spaces are important and which are not. (In a pinch, you might be able to infer this from performance group assignments.)

Before committing to goal mode with server management, make sure that the CICS MAXTASKS setting for each region is as low as it can be (just above the high water mark of concurrent active transactions) with a slight margin for safety. Excessive MAXTASKS can lead to excessive WLM overhead.

After you have demonstrated a painless transition to goal mode, check with the CICS wizards to see if they want to get on the bandwagon. They might look at adding service classes to increase the influence of Loved Ones and diminish that of unimportant work. They have to supply the transaction name lists and you can do the rest easily.

With the post-V2R10 changes, you can do a lot more when differences are important: remove selected regions from server management, designate regions or transactions as Storage Critical and service classes as CPU Critical. You can also use much more straightforward classification rules when work is segregated by system.

Classifying CICS Address Spaces

```
Subsystem Type . . . . . : STC
Description . . . . . . . . . . All started tasks
  -------Qualifier-----------     -------Class-------  Storage   Manage Regions
  Type      Name       Start     Service    Report     Critical  to Goals of
                       DEFAULTS: DISC____   ________   ___       ___________
1 TN        %MASTER%   ___       SYSTEM__   MASTER__   ___       ___________
  ... ...
1 SY        HOTPROD    ___       STCMED__   GENSTC__   NO_       ___________
2  TNG      HOTCICS    ___       VEL60___   CICSREGS   NO        TRANSACTION
2  TN       CICSPO3    ___       TOPCICS    CICSREGS   YES       REGION
1 TNG       HOTCICS    ___       CICS       CICSREGS   NO        TRANSACTION
1 TN        CICST*     ___       TESTCICS   CICSREGS   NO        REGION
1 TN        %%%%IRLM   ___       SYSSTC__   IRLMS___   YES       ___________
1 TNG       DB2_____   ___       VEL60___   DB2S____   NO_       ___________
```

RED denotes function added in OS/390 V2R10

Here are examples of the new options for classifying address spaces. For CICS and IMS, the "Manage Regions to Goals of" column may be use to choose how resource flow to each address space is managed. The default is "TRANSACTION", the current method. In this mode, as soon as an address space is recognized to be a server to transaction service classes, its velocity goal is ignored and the regions are managed in the interest of the transaction service classes they serve.

If the "REGION" choice is made, server management is not used. If only some AORs of a CICS subsystem are to be managed with velocity goals, the TOR must not be designated as managed to REGION goal. (Classification of transactions will be bypassed.)

The "Storage Critical" column allows a choice of how the working set of an idle address space will be managed. The choice of "NO" preserves the current aggressive page stealing; "YES" says that the address space's frame count will stay close to its established high water mark without stealing during periods of inactivity. The "Storage Critical" designation may also be used for transactions classified in the CICS subsystem. Doing so causes the address spaces in which those transactions run to be managed as "storage critical"—if and only if they are managed to transaction goals. Even though "Storage Critical" may be specified for a single rule affecting as little as a single transaction, all transactions of the service class are so regarded and thereby any region in which the service class is active.

In this example, regions on system HOTPROD get special treatment. CICS regions named in TNG HOTCICS are designated as Storage Critical and are always managed according to the velocity goal of service class VEL60.

HOTCICS regions on other systems receive no special treatment.

Regions whose names begin with the string CICST receive a different kind of special treatment. In this case they always managed to the velocity goal in service class TESTCICS—presumably a low velocity and importance. This is a way to eliminate the overhead of server management and WLM interaction for test regions.

CICS Address Space Service Class

```
Service Class Name . . . . . : TOPCICS
Description  . . . . . . . . . Super-hot CICS
Workload Name  . . . . . . . . CICS       (name or ?)
Base Resource Group  . . . . . ________   (name or ?)
CPU Critical . . . . . . . . . YES        (YES or NO)

---Period---  ---------------------Goal---------------------
#  Duration   Imp.  Description
1              1    Velocity=60%
```


Here is the TOPCICS service class. Not only is it Importance 1 with a high velocity goal, it is also designated as CPU Critical, thus giving it a guaranteed high dispatching priority, above those of other work units at its level of importance. In this case with Importance 1, only started tasks classified as SYSTEM or SYSSTC will have higher priorities.

CICS Classification—First Steps (Report Class Only)

```
 Subsystem Type . . . . : CICS
 Description  . . . . . : IBM-defined subsystem

-------Qualifier-------------        -------Class-------- Storage
   Type     Name     Start           Service     Report    Critical
                           DEFAULTS:________     ALL_CICS
```

CICS Classification— Single Service Class

```
Subsystem Type . . . . : CICS
Description  . . . . . : IBM-defined subsystem

-------Qualifier-------------        -------Class-------- Storage
   Type     Name     Start           Service     Report    Critical
                                DEFAULTS:DFLTCICS    ALL_CICS
```

CICS Classification— More Report Classes

```
 Subsystem Type . . . . : CICS
 Description  . . . . . : IBM-defined subsystem

-------Qualifier-------------        -------Class-------- Storage
   Type     Name     Start            Service     Report    Critical
                                DEFAULTS:DFLTCICS    NOT_HOT
1  SI       CICST*   ___              ________    TEST____  ___
  2  TNG    OPFUNCS  ___              ________    OPFNS___  ___
1  TNG      RHLOVED  ___              ________    HOT_____  ___
1  TNG      LOVED    ___              ________    HOT_____  ___
1  TNG      UNLOVED  ___              ________    NOT_HOT_  ___
1  TNG      OPFUNCS  ___              ________    OPFNS___  ___
```

CICS Classification—Add Loved Ones

```
 Subsystem Type . . . . : CICS
 Description  . . . . . : IBM-defined subsystem

-------Qualifier-------------        -------Class-------- Storage
   Type     Name     Start            Service     Report    Critical
                              DEFAULTS:DFLTCICS    NOT_HOT
1  SI       CICST*   ___              ________    TEST____  ___
1  TNG      RHLOVED  ___              CICSRHOT    HOT_____  ___
1  TNG      LOVED    ___              ________    HOT_____  ___
1  TNG      UNLOVED  ___              ________    NOT_HOT_  ___
1  TNG      OPFUNCS  ___              ________    OPFNS___  ___
```

CICS Classification—
Add *Really* Loved Ones

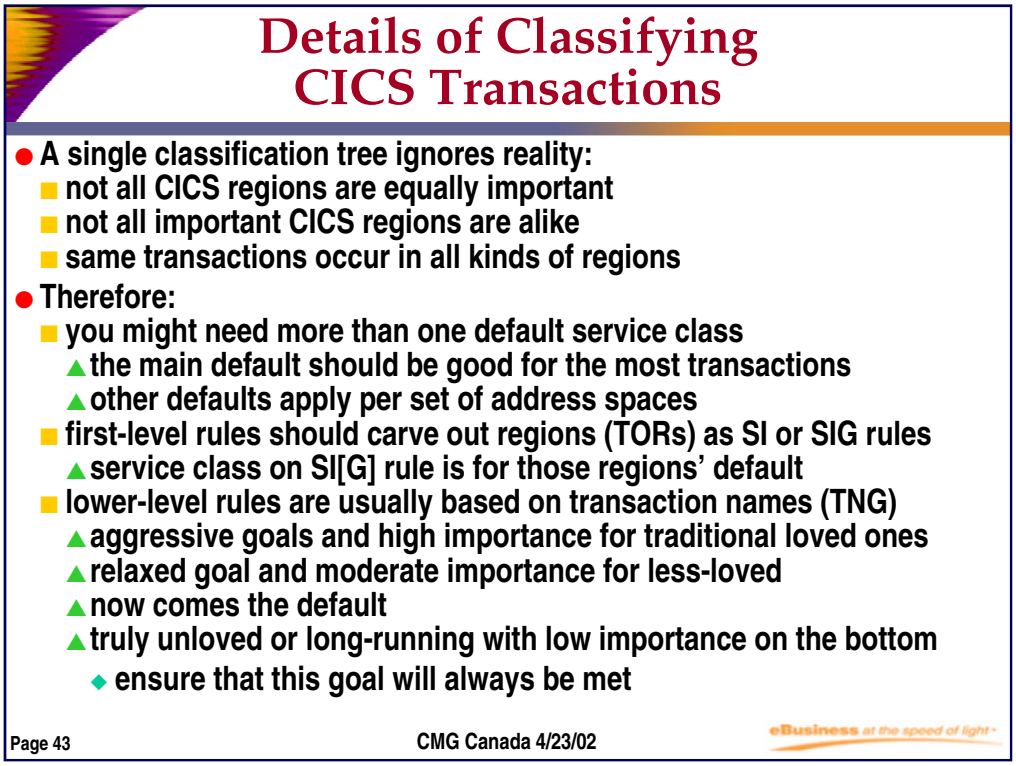
```
 Subsystem Type . . . . : CICS
 Description  . . . . . : IBM-defined subsystem

-------Qualifier-------------        -------Class-------- Storage
   Type     Name      Start            Service     Report    Critical
                              DEFAULTS:DFLTCICS    NOT_HOT
1  SI       CICST*    ___              ________    TEST____    ___
2    TNG    OPFUNCS   ___              ________    OPFNS___    ___
1  TNG      RHLOVED   ___              CICSRHOT    HOT_____    ___
2    UI     TOPDOG    ___              SUPERHOT    HOT_____    YES
1  TNG      LOVED     ___              ________    HOT_____    ___
1  TNG      UNLOVED   ___              ________    NOT_HOT_    ___
1  TNG      OPFUNCS   ___              ________    OPFNS___    ___
```

CICS Classification— More Service Classes

```
 Subsystem Type . . . . : CICS
 Description  . . . . . : IBM-defined subsystem

-------Qualifier-------------        -------Class-------- Storage
   Type     Name     Start            Service     Report    Critical
                              DEFAULTS:DFLTCICS    NOT_HOT
1  SI       CICST*   ___              CICSLOW_    TEST____  ___
2    TNG    OPFUNCS  ___              CICSLOW_    OPFNS___  ___
1  TNG      RHLOVED  ___              CICSRHOT    HOT_____  ___
2    UI     TOPDOG   ___              SUPERHOT    HOT_____  YES
1  TNG      LOVED    ___              CICSHOT_    HOT_____  ___
1  TNG      UNLOVED  ___              CICSLOW_    NOT_HOT_  ___
1  TNG      OPFUNCS  ___              DFLTCICS    OPFNS___  ___
```



In the real world, there are usually many CICS regions, differing by business importance and applications served. A single default transaction service class is rarely applicable across all regions. Fortunately the classification capabilities of goal mode allow you to choose different defaults for each address space or group of address spaces.

The rules may get a bit cluttered but if the transactions of the most critical regions are classified first, performance will be affected very little.

For each significant production region, three or four service classes may be enough:

- one for top performance
- a second for slightly less critical transactions
- the third for the great majority of transactions—the local default
- finally, a service class with a very easy goal and low importance to ensure that unimportant or conversational transactions have very little influence on the management of resource flow to the region(s).

Goal Definition for a CICS Service Class

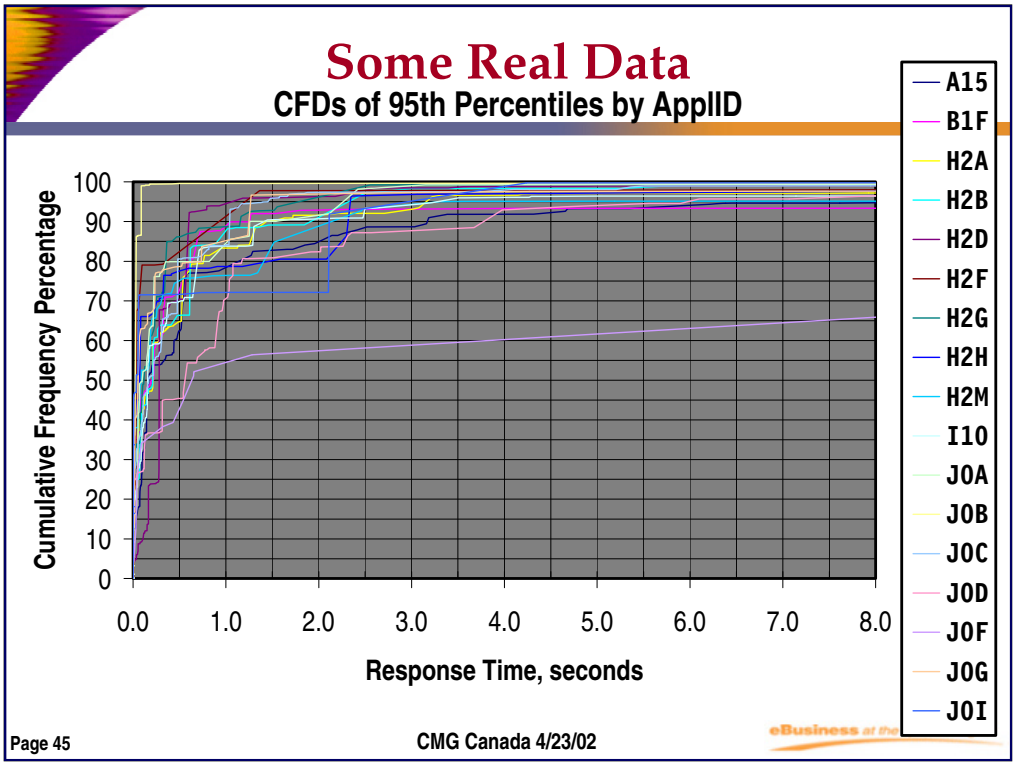
```
Service Class Name . . . . . : TOPTRAN
Description  . . . . . . . . . Time-critical CICS
Workload Name  . . . . . . . . CICS       (name or ?)
Base Resource Group  . . . . . ________   (name or ?)
CPU Critical . . . . . . . . . YES


---Period---  ----------------------Goal---------------------
#  Duration   Imp.  Description
1               1    85% complete within 00:00:01.500
```


Here is a service class definition for a CICS transaction response time service class. Note that it has only one period. Since transaction response time service classes do not own resources, they do not accumulate service and thus cannot move through periods. Note also that it is a percentile goal.

This service class is not in a resource group, and it is is designated as CPU Critical. Every address space in which it runs “inherits” this attribute and level of importance until the service class ceases to be active for approximately 20 minutes.

If CP/SM is used to determine transaction placement based on WLM data, the goal must be an average RT goal, as of OS/390 V2R10.



Here is data from a customer site, showing 17 of about 35 CICS regions on a single system. These are all independent regions (no MRO). What is plotted is the cumulative frequency of response times at the 95th percentile per transaction name. Note that most of the regions would do well if the transaction response time goal is 1 second at the 84th percentile. However, six of the regions cannot achieve this goal and one will do much better. Additional service classes will be needed to allow transactions in these regions to meet their goals and achieve appropriate PIs.

These classes would include:

- 1 second at the 54th percentile, for one region
- 1 second at the 72nd percentile, for two regions
- 1.5 seconds at the 80th percentile, for three regions
- 0.25 seconds at the 95th percentile, for the wild overachiever

We retain the default of 1 second at the 85th percentile at importance 3, as well as the service classes for Loved Ones and The Unloved.

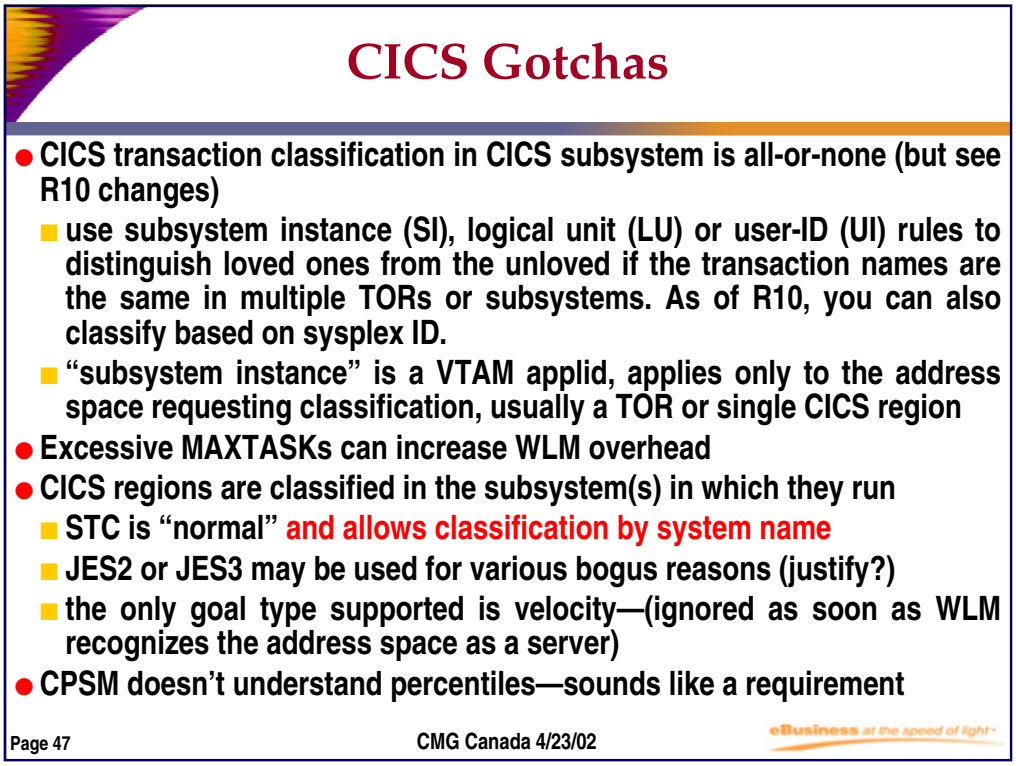
CICS rules in a "Real" Service Definition

```
  Default service class is CICSDFLT (P85S10I3)
Qualifier  Qualifier        Starting       Service
# type       name           position       Class
- ---------- -------------- ---------      --------
1 TNG        STARS                         CICSLOVD (P85S10I2)
1 TNG        WORST                         CICSUGLY (P10H2I5)
1 SIG        FASTEST                       P90S07I3
  2 TNG        EXCEPT1                     P80S40I3
1 SIG        SLOWER                        P85S15I3
1 SIG        SLOWEST                       P80S40I3
1 SIG        WEIRD1                        P75S20I3
1 SIG        WEIRD2                        P55S30I3
```

Pointer to group of server regions named as VTAM APPLIDs

"Default" for this group of regions

Pointer to group of exception transactions



Here are reminders of some typical problems encountered when running CICS in goal mode.

One problem is the all-or-nothing nature of running with transaction response time goals. However, it need not be a problem if the flexibility of the classification scheme is exploited. Suppose that transaction BUZZ is very important on System A and just one of the players on the rest of the systems. Assuming that the VTAM appl-IDs of the TORs on the systems are different (as they should be), a Subsystem Instance rule can pick off System A's CICS first and classify BUZZ as going to service class CICSHOT. BUZZ on other systems can be classified differently or just allowed to default to service class ROUTINE.

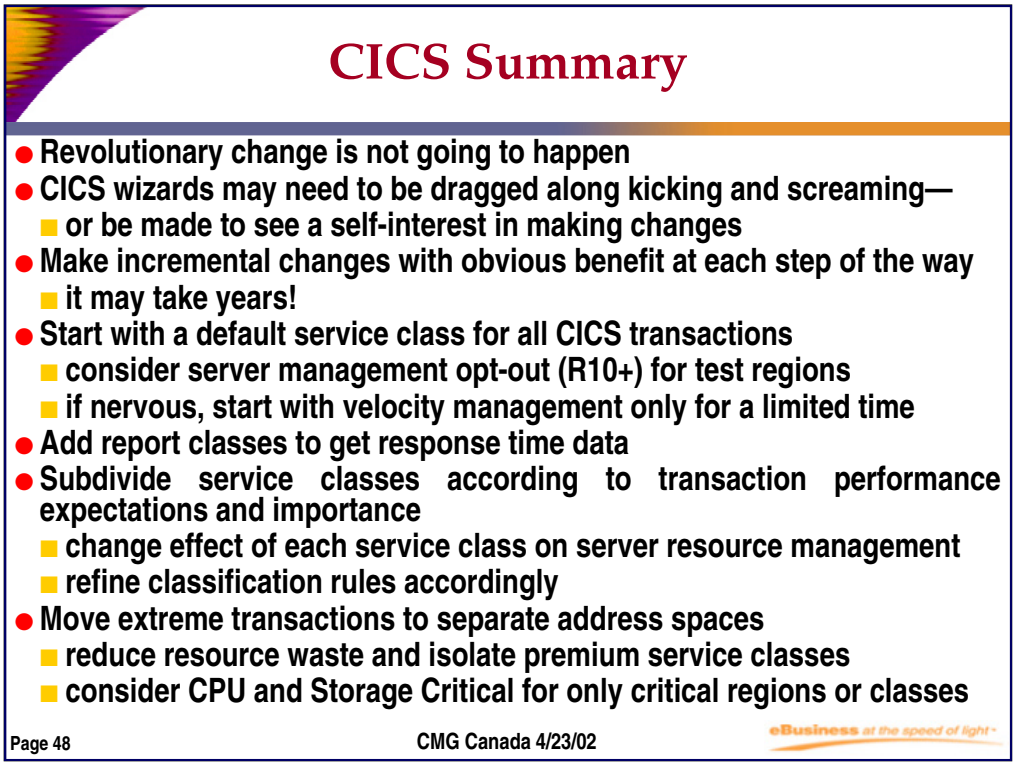
The post-V2R10 changes make it easier and more direct to differentiate work according to the system on which it executes.

The interface between CICS and WLM depends on the construction, updating, and polling of data fields called Performance Blocks (PBs). Once an address space is recognized as a server, a number of PBs equal to MAXTASKS is built. If MAXTASKS is excessive across a large number of CICS regions, WLM overhead can rise unnecessarily. Cut MAXTASKS to a realistic value.

Another source of confusion is the need to classify two different sets of entities two different ways—CICS address spaces in the subsystems in which they run (STC or JESx) and CICS transactions in the CICS subsystem. The permitted goal types are different, and the use of the goals is different. Once the relationship between server address spaces and the service classes they serve is established, the address space becomes a totally dependent entity, managed on behalf of its client service classes. Transaction service classes do not accumulate service or have measured service rates; therefore they cannot be in resource groups or have multiple periods.

A wrinkle appears when the CICSplex Service Manager (CPSM) is being used to route transactions based on goal attainment: CPSM understands only average response times, not percentiles. The need for a user group requirement is obvious. (The alternative is not to use the WLM option of CPSM if percentile response time goals are to be used.)

One last question: can anybody explain why CICS should be run as a batch job?



Once you get over the idea that the CICS staff will drop everything and rush to do your bidding just because you ask, a more practical approach to CICS exploitation of goal mode may be considered.

The usual first step is to avoid the hard questions entirely and set velocity goals for the CICS address spaces—and then walk away. Doing that, however, perpetuates the compatibility mode situation and may even make it worse. At times of constraint, achieving the set velocity may require more dispatching priority and/or storage protection than was necessary in compatibility mode. Better to start with a default CICS service class.

You have now achieved server management without any knowledge of the transaction set. Now the velocity is a consequence of how you are managing CICS, not the means of management.

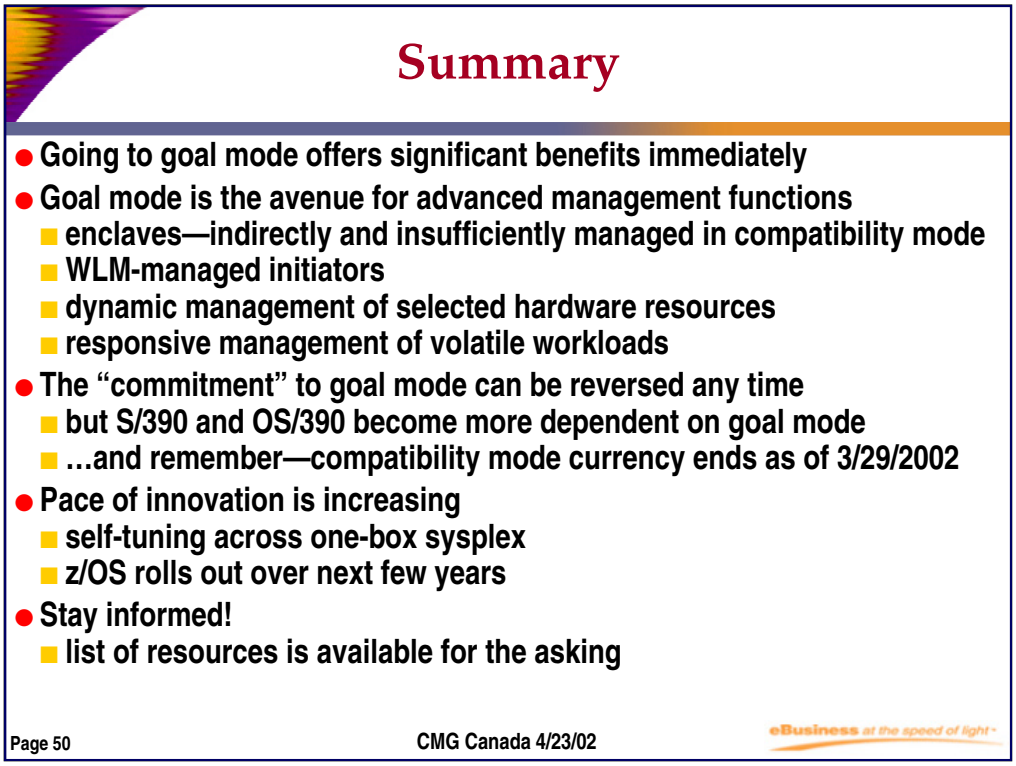
To go on, add report classes for specific lists of transactions. Again, after analyzing data, add service classes so that the more important transactions tend to be the ones that determine server resource distribution.

After it becomes obvious that some transactions deserve special treatment, put the most-loved ones in separate address spaces, and the least-loved in other separate address spaces.

This whole evolution may take several months, but each month you'll be wasting less resources and managing CICS better—and the CICS wizards can become your cooperating allies.

Information Sources

- Read the manual!
  - SA22-7802-02 z/OS V1R2.0 MVS Planning: Workload Management
  - download as a PDF from:
    http://www-1.ibm.com/servers/eserver/zseries/zos/bkserv/r2pdf/
- Read the Redbooks!
  - check the website (*http://www.redbooks.ibm.com/*) frequently
- Go to conferences! (OS/390-z/OS EXPO, ASG, SHARE, GSE, CMG)
- Browse the WLM website:
  http://www.ibm.com/servers/eserver/zseries/zos/wlm/
- Check Cheryl Watson at *http://www.watsonwalker.com*
- Get on the discussion groups:
  - IBM-MAIN (free newsgroup *bit.listserv.ibm-main* or mailing list *IBM-MAIN@bama.ua.edu*) (subscribe at *listserv@bama.ua.edu*)
  - "official" IBM newsgroups at *news.software.ibm.com*
  - MXG-L (free mailing list)
  - … there are others as well … *e.g. search390.com*



Goal mode is enormously superior to compatibility mode in its ability to control directly the performance of key workloads. It also can get the most possible out of the hardware. By managing the dispatching priority and storage protection of address spaces dynamically, goal mode makes premium resources available to work units only when they need those resources. Maximum opportunity for other work units to compete for those resources is preserved.

Goal mode enables the implementation of advanced management functions in OS/390 and will do so to a greater extent in z/OS releases.

With all its benefits, and with the need for rethinking of how work is set up on a system, going to goal mode is risk-free. A simple operator command allows a system to be switched [back] to compatibility mode instantly.

However, the fallback to compatibility mode will not be available indefinitely. Once IBM overcomes all valid technical objections to goal mode, it can, and eventually will, cease to support compatibility mode. This will happen in the third release of z/OS, in the first half of 2002.

That decision may be moot when the operating system, I/O subsystems, and even multiple LPAR support become dependent on goal mode for full function.

For now, you have a choice of goal mode or compatibility mode. That choice will eventually disappear. So you have another choice—enjoy the benefits of, and begin exploiting, goal mode today, or wait….