Redbooks
ibm.com/redbooks

Using IBM z/OS WLM to Measure Mobile and Other Workloads

IBM Client Center

Montpellier

Mobile

z Systems

Redpaper

International Technical Support Organization

Using IBM z/OS WLM to Measure Mobile and Other Workloads

October 2016

REDP-5359-00

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

First Edition (October 2016)

This edition applies to Version 2, Release 1, of IBM z/OS.

This document was created or updated on October 25, 2016.

Contents

Notices

Trademarks

Preface

This IBM® Redpaper™ publication discusses the need to monitor and measure different workloads, especially mobile workloads. It introduces the workload classification capabilities of IBM z™ Systems platforms and helps you to understand how recent enhancements to IBM MVS™ Workload Management (WLM) and other IBM software products can be used to measure the processor cost of mobile workloads.

This paper looks at how mobile-initiated and other transactions in IBM CICS®, IMS™, DB2®, and WebSphere® Application Server can be “tagged and tracked” using WLM. For each of these subsystems, the options for classifying mobile requests and using WLM to measure mobile workloads are reviewed.

A scenario is considered in which a bank is witnessing a significant growth in mobile initiated transactions, and wants to monitor and measure the mobile channels more closely. This paper outlines how the bank can use WLM to do this.

This publication can help you to configure WLM mobile classification rules. It can also help you to interpret Workload Activity reports from IBM RMF™ Post Processor and to report on the CPU consumption of different workloads, including mobile and public cloud workloads.

Authors

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

Nigel Williams is an IT Specialist working at the IBM Client Center in Montpellier, France. Nigel specializes in CICS integration, API enablement, and mainframe security topics. He helps clients to design and test mainframe integration solutions. He is the author of many papers and IBM Redbooks® publications. Previously Nigel worked as a Developer and Tester at the IBM Hursley Software Lab.

Olivier Boehler is an IT Specialist at the IBM Client Center Montpellier, France. He has 8 years of experience in IBM including 4 years spent at the WW Mainframe benchmark center in Montpellier. He holds an engineering degree in computer science from Université des Sciences, Montpellier. Olivier’s area of expertise includes IBM DB2 for z/OS® performance, tuning, and high availability solutions.

Philippe Bruschet is a Certified IT Specialist in France. He has 30 years of experience in the mainframe field. He has worked at IBM for 8 years, and was previously, for more than 20 years, an IBM customer. His areas of expertise include z/OS, WLM, and performance and tuning of z/OS configurations.

Francois Capristo joined IBM in 2003 and started working on Cloud Solutions for provisioning IBM Education systems where he acquired a strong experience in analysis, design, and implementation of cloud infrastructures. Recently he became the leader of the IBM Montpellier Banking Showcase team and is responsible for showcase development, production support, and customer engagements.

Alexis Chretienne is an IT Specialist at the IBM Montpellier Client Center, Montpellier, France. He holds an engineering degree in Mathematics and Computing from the École Nationale Supérieure d'Informatique et de Mathématiques Appliquées de Grenoble (ENSIMAG). He has 3 years of experience and his areas of expertise include CICS Transaction Server (TS), z/OS Connect, and API Connect.

Stéphane Faure is an IT Specialist with the Center for Systems Innovation at the IBM Client Center Montpellier, France. He has 26 years of experience in IT. He holds a degree in Computing from Gustave Eiffel College, Bordeaux. His areas of expertise include IBM z Systems™ operating systems and middleware. He was written extensively about IBM z/VM® and WebSphere Application Server on z/OS.

Richard Gamblin is an IBM Technical Staff Member. As the European Technical Leader for Digital Transformation, Richard works with clients to develop new solutions and capabilities with IBM middleware, mobile, and z Systems technologies. He has worked in a number of Technical Sales roles during his time in IBM, ranging from an Integration and Connectivity Specialist to an IBM WebSphere Architect. Prior to joining IBM, Richard was a researcher at the University of Leeds, from where he obtained a doctorate in the field of bioinformatics.

Fabrice Jarassat is a Certified IT Specialist working on the Banking Showcase team, in Montpellier, France. Before joining IBM in 2000, he worked for a large distribution company where he supported CICS transaction and payment systems. His areas of expertise include CICS TS, Sysplex, CICSPlex/SM, CICS tools, and IBM GDPS®.

Arnaud Mante has worked at IBM for 12 years. After two years spent promoting WebSphere Solutions on z Systems and supporting workshops for European customers, he is now a z Systems administrator. Starting this job in IBM Education, he is now working in the Banking Showcase team at the IBM Montpellier Client Center, Montpellier, France.

Irene Stahl is a Software Engineer with the z/OS Workload Management team in Boeblingen. After receiving her doctorate in Computer Science, she has worked for over 20 years on various z Systems projects, including eight years in z/OS Workload Management.

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

Lydia Parziale and Diane Sherman
International Technical Support Organization, Poughkeepsie Center

Chris Baker and John Burgess
CICS Development, Hursley

Martin Packer
Worldwide Cloud & Systems Performance, IBM

Yann Kindelberger, Executive IT Architect
IBM Montpellier

Now you can become a published author, too!

Here's an opportunity to spotlight your skills, grow your career, and become a published author—all at the same time! Join an ITSO residency project and help write a book in your area of expertise, while honing your experience using leading-edge technologies. Your efforts will help to increase product acceptance and customer satisfaction, as you expand your network of technical contacts and relationships. Residencies run from two to six weeks in length, and you can participate either in person or as a remote resident working from your home base.

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

ibm.com/redbooks/residencies.html

Comments welcome

Your comments are important to us!

We want our papers to be as helpful as possible. Send us your comments about this paper or other IBM Redbooks publications in one of the following ways:

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

  ibm.com/redbooks

- Send your comments in an email to:

  redbooks@us.ibm.com

- Mail your comments to:

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

Stay connected to IBM Redbooks

- Find us on Facebook:

  http://www.facebook.com/IBMRedbooks

- Follow us on Twitter:

  http://twitter.com/ibmredbooks

- Look for us on LinkedIn:

  http://www.linkedin.com/groups?home=&gid=2130806

- Explore new Redbooks publications, residencies, and workshops with the IBM Redbooks weekly newsletter:

  https://www.redbooks.ibm.com/Redbooks.nsf/subscribe?OpenForm

- Stay current on recent Redbooks publications with RSS Feeds:

  http://www.redbooks.ibm.com/rss.html

1

Introduction

This chapter introduces the workload classification capabilities of IBM z Systems and discusses the need to monitor and measure different workloads. It also reviews the special characteristics of mobile and cloud workloads, with a consideration of how to cope with the growing impact of these workloads.

1.1 Workload classification and measurement

The diversity of business services required to power today's enterprises is significant and growing. They range from legacy payroll and billing systems that run periodically, to leading-edge digital services that provide mobile and web users instant access to a full complement of services, at any time of the day or night. This diversity places increasing demands on enterprise IT systems and hosting platforms, which must manage and protect the integrity of these different services.

The design philosophy for IBM z Systems is to deliver a high-performing, resilient, secure, and scalable hosting environment for precisely this type of diverse set of applications and data services. Workloads on z Systems can be segregated at different levels, including firmware, hypervisor, and within the operating system.

- At the lowest level, the IBM Processor Resource/Systems Manager™ (IBM PR/SM™) acts as a type-1, or bare metal hypervisor, and provides the most robust mechanism to partition workloads into logical partitions (LPARs).
- Within each LPAR, hypervisors allow multiple operating system (OS) guests to be run independently of each other, separating workloads at the OS level. Two hypervisors are available on IBM z Systems: z/VM and KVM.
- Multiple operating systems are available on z Systems, including z/OS, Linux, IBM z/VSE® and TPF. Specifically, within z/OS, built-in mechanisms enable workloads to interoperate or remain separate where appropriate.

The focus of this paper is on the exploitation of these built-in mechanisms in z/OS to allow workloads to be classified and measured. The use of "measured" in this paper means the performance characteristics of a workload, such as response time and CPU consumption.

The MVS Workload Management (WLM) component is an integral part of z/OS that provides a solution for managing workload distribution, balancing, measurement, reporting, and prioritization. It is implemented through the cooperation of various subsystems, such as CICS, IMS, or JES, allowing WLM to allocate sufficient system resources—processing capacity or storage—to meet a set of assigned performance goals for a given application.

For most types of work (started tasks, batch work, enclaves, and more), WLM measures the CPU consumption of the work requests. For transaction systems like CICS and IMS, CPU consumption has been traditionally reported to the address space rather than the transaction. However, CICS and IMS have been enhanced recently to continuously report CPU consumption of transactions to WLM. These measurements are then aggregated by WLM to provide enhanced reporting in IBM System Management Facilities (SMF) records and IBM Resource Measurement Facility™ (RMF) reports. The CPU measurements for workloads classified in this way make reporting of CPU consumption for mobile and cloud workloads simpler and more efficient and reliable:

- Simpler because it does not require middleware-specific tooling
- More efficient and low cost because it does not require collecting and evaluation high-volume transaction level accounting data
- More reliable because it is deeply integrated into the operating system.

1.1.1 Why reporting on workloads is important

You cannot manage what you cannot measure, therefore, clearly classifying and measuring types of requests is important. Reporting on types of workloads might be necessary for several reasons:

- Capacity planning

  Accurate planning of future capacity requirements depends on having knowledge today of transactions rates, resource consumption, and trends. In a multi-channel environment, different channels grow or decline at different rates. Accurate predictions can be made only when the current capacity requirements of each channel are known and trend analysis is performed.

- Managing spikes in workload

  Certain workloads are characterized by greater variability and are therefore more difficult to predict (see 1.1.2, "Characteristics of mobile and cloud workloads" on page 3). Workload trend analysis for these channels is therefore important because of the potential need to provision more resources, or enable specific controls, in order to cope with workload peaks.

- Charge back

  In the case where an infrastructure is shared between different entities, lines of business, or different companies, measuring the resources consumed by the workloads of the different entities is necessary. This can be done by segregating workloads into different LPARs. However, it is also possible to classify and monitor the workloads of different entities within the same LPAR to form the basis of a usage-based billing system.

- Pricing

  Software pricing models like IBM Mobile Workload Pricing (MWP), IBM z Systems Workload Pricing for Cloud (zWPC), and IBM z Systems Collocated Application Pricing (zCAP) require the specially priced workloads to be measured and reported.

  MWP provides an enhanced way of reporting z/OS system utilization, which can improve price/performance for subcapacity-eligible programs when running in the same LPAR as mobile workloads processed by select MWP defining programs (see 1.2, "Introduction to Mobile Workload Pricing" on page 4).

  zWPC can improve price/performance for subcapacity-eligible programs when running in the same LPAR as new public cloud workloads processed by select zWPC defining programs (see 1.3, "Introduction to z Systems Workload Pricing for Cloud" on page 7).

  zCAP allows new applications to be deployed in existing LPARs, yet be priced as if running in dedicated LPARs.

1.1.2 Characteristics of mobile and cloud workloads

Mobile devices (smartphones and tablet computers) offer a level of hyper-connectivity and interaction with each other, businesses, and employers, unlike any other time in history. Reports have estimated that, on average, a smartphone owners check their device over 150 times in a day, and that 90% of us keep that same device within arm's reach 24 hours a day.

This shift in the ease with which we can transact means that we are engaging with organizations at an unprecedented rate. Unlike web-based access, where there was a requirement to be bound to a fixed network in an office or home; mobile access means that we can access business services at our convenience, day or night in any location.

Moreover, the simplicity of mobile app and web designs has lowered the barrier to accessing these services: a product can be browsed, researched, and purchased in a matter of a few quick gestures and taps.

As a result of these characteristics, mobile services drive certain behaviors that have had a significant impact on the underlying enterprise IT systems. Evidence has shown that mobile apps influence these results:

- Drive an increase in overall transactions
- Increase off-peak web traffic
- Exacerbate existing workload peaks

In parallel, another new style of computing is having a significant effect on enterprise IT: cloud computing. The adoption of cloud, as a new delivery model for IT infrastructure and services, has seen organizations deliver as-a-service models (infrastructure as a service, platform as a service, and software as a service) through a combination of public and private cloud technologies. One of the advantages of public cloud infrastructure, such as IBM Bluemix®, is that new services can be created and made available without the lead times associated with traditional IT systems. A further advantage comes with the extensible nature of cloud infrastructure, so that unanticipated workload can be accommodated in proportion to the ongoing cost of the service: a pay-per-use model.

One of the characteristics of cloud applications is that, in order to offer meaningful services, they require integration with enterprise applications and data. As a result, this hybrid cloud model harnesses the rapid scale and burst capacity capabilities of cloud to the traditional enterprise IT systems.

To be successful, organizations must have an infrastructure that can handle the demands of emerging digital business, without sacrificing service times and without deteriorating the experience of customers. To achieve this, new mobile and cloud-based workloads must be identified as part of the application deployment process so that they can be properly managed and measured.

Note: This paper focuses on measuring mobile workloads; however, many of the techniques shown here can be equally used to measure other types of workload.

1.2 Introduction to Mobile Workload Pricing

Mobile Workload Pricing for z/OS (MWP) was introduced in June 2014 as an enhancement to Sub-Capacity licensing, and is designed to mitigate against the growing impact of mobile workloads on IBM z/OS software costs. For organizations that exploit Sub-Capacity pricing for z/OS, rather than pay the full license cost of the available system resources, they pay a usage-based variable Monthly License Charge (MLC) for their software. This is calculated based on the rolling 4-hour average (R4HA) utilization of the z/OS LPARs observed within a one-month reporting period.

MWP addresses the impact of mobile workloads by allowing a 60% reduction of mobile-initiated transactions that are recorded at the peak R4HA by the MWP Defining Programs. A number of devices can be considered mobile and, for the purposes of MWP, these are defined as smartphones and tablet computers.

The MWP Defining Programs include these items:

- CICS Transaction Server (V4 and V5)
- CICS Value Unit Edition (VUE) V5
- IBM DB2 for z/OS (V9, V10, V11, and V12)
- DB2 VUE (V9, V10, V11, and V12)
- IBM IMS TM (V11, V12 and V13)
- IBM IMS Database VUE (V12 and V13)
- IBM MQ for z/OS (V6, V7, V8, and V9)
- IBM MQ VUE (V7, V8, and V9)
- IBM WebSphere Application Server for z/OS (V7, V8, and V9)

The impact of MWP on the peak rolling 4-hour average is illustrated in Figure 1-1.

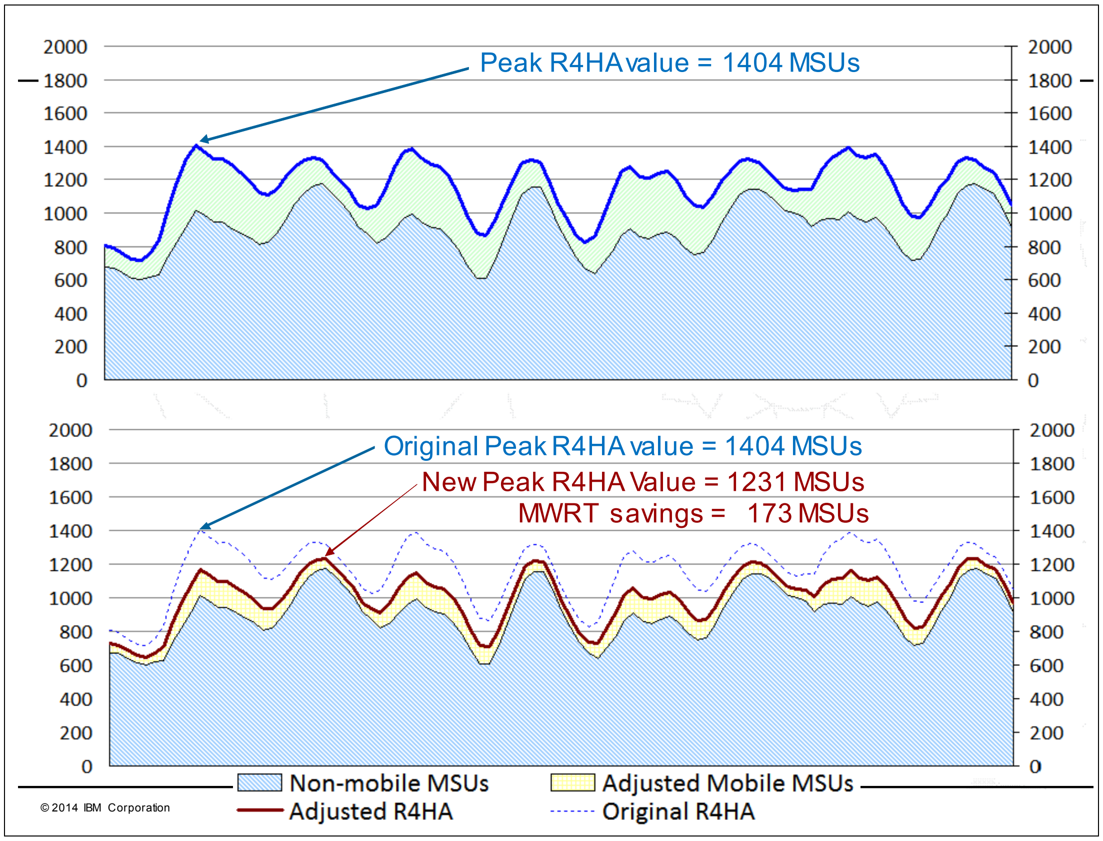


Figure 1-1 Example of Mobile Workload Pricing for z/OS and the impact on the R4HA MSU peak

The upper chart shows the peak R4HA over a period of time, with mobile and non-mobile workloads respectively separated into green and blue shading. The R4HA peak is the first peak, recorded as millions of service units (MSU), 1404 MSUs, and it contains some mobile-initiated transactions. The lower chart shows the impact of the 60% reduction in MSUs as a result of MWP.

The two notable impacts on the peak R4HA are as follows:

- It has reduced from 1404 MSUs to 1231 MSUs.
- It has moved from the first to the second peak in this period.

The way in which MWP works is that the R4HA is assessed on an hour-by-hour basis, per z/OS LPAR, and then the 60% reduction is applied for MWP Defining Programs. This will result in one of three outcomes:

- No change in the overall R4HA peak, because the mobile workloads make no contribution to the overall peak and are therefore absorbed at no additional cost.
- The R4HA peak is reduced because mobile workloads do contribute to the peak.
- The R4HA peak is reduced, and moves to a different peak because, after the MSU reduction, the peak value of consumed MSUs is now located elsewhere in the time period.

Important: Only general purpose processor CPU time is eligible for MWP.

1.2.1 Requirements for Mobile Workload Pricing

To take advantage of MWP, organizations must be able to track and report the general purpose processor (GP) time (CPU Time) for mobile transactions and report those values in a predefined format. Tooling provided by IBM, either the Mobile Workload Reporting Tool (MWRT) or the enhanced Sub-Capacity Reporting Tool (SCRT), will use the reported mobile transaction data to adjust the rolling 4-hour average subcapacity MSUs for subcapacity-eligible programs on a given machine.

In practice, this means that organizations must do the following task:s

- Clearly define the mechanism by which transactions are identified as being initiated by mobile devices (smartphone or tablet computers).
- Provide an auditable mechanism for recording the GP processor time for the relevant MWP Defining Programs.

In terms of identifying mobile-initiated transactions, organizations have adopted a variety of strategies. In some cases, which transactions originate from a mobile device is clear, such as in the case of a mobile application that always communicates with z Systems applications through a specific mobile platform, for example, an IBM MobileFirst™ server. In other cases, the source of the mobile transaction is clear because mobile users are automatically directed to a mobile-only version of a website. In some cases, however, such as where a single website is used for mobile and non-mobile devices or mobile apps reuse an existing Internet gateway, a necessary step is to examine the request in order to identify the device type.

After the transaction origin is identified, this information must be propagated to the z/OS subsystem in order to be correctly audited and measured as a mobile-initiated transaction. Organizations have adopted multiple approaches to achieving this, such as directing mobile workloads to cloned applications listening on different TCP/IP ports, or directing mobile traffic to separate transaction processing regions or even different LPARs.

However, one of the key design principles of MWP has been that it does not require any infrastructure changes in order to be exploited. This paper shows how recent enhancements to WLM and other z/OS software products make it much easier to classify and measure mobile workloads, particularly when transaction processing regions run both mobile and non-mobile workloads. The primary objective of this paper is to provide worked examples of how WLM can be used to measure the CPU consumption of mobile workloads.

1.3 Introduction to z Systems Workload Pricing for Cloud

z Systems Workload Pricing for Cloud (zWPC) was introduced in July 2016 as a further enhancement to Sub-Capacity licensing. It is designed to improve mainframe workload economics, as z Systems increasingly integrate with public cloud applications, which can result in larger workload volumes.

zWPC addresses the impact of public cloud workloads by allowing a 60% reduction of public cloud initiated transactions that are recorded at the peak R4HA by the zWPC Defining Programs. Public cloud workloads are defined as transactions initiated from a named public cloud offering, such as IBM Bluemix, Amazon Web Services, Microsoft Azure, and others. The zWPC Defining Programs are the same as the MWP defining Programs (see 1.2, “Introduction to Mobile Workload Pricing” on page 4), and the impact of zWPC on the R4HA utilization is the same as illustrated in Figure 1-1 on page 5.

1.3.1 Requirements for z Systems Workload Pricing for Cloud

To take advantage of zWPC, organizations must be able track and report the general purpose processor (GP) time (CPU Time) for public cloud transactions and report those values in a predefined format. IBM has recently provided an enhanced version of SCRT (Version 24.10) which uses the reported public cloud transaction data to adjust the R4HA subcapacity MSUs for subcapacity-eligible programs on a given machine.

In practice, this means that organizations must do these actions:

- Clearly define the mechanism by which transactions are identified as being initiated by public cloud applications.
- Provide an auditable mechanism for recording the GP processor time for the relevant zWPC Defining Programs.

The techniques for identifying public cloud initiated transactions are likely to be similar to those used for identifying mobile initiated transactions (see “1.2.1, “Requirements for Mobile Workload Pricing” on page 6). These techniques are explored in detail in this paper.

Using MVS WLM to measure mobile workloads

To measure the CPU consumption of a mobile workload, for example to take advantage of MWP, you need a way to identify mobile-initiated transactions and a process to report on the CPU consumed by the transactions.

Initially, this type of reporting required product-specific tooling from high-volume, transaction-level accounting data. With the recent enhancements made to CICS, IMS and the MVS Workload Management (WLM) component of z/OS, the process of aggregating mobile CPU associated with mobile is much simpler. Together with your favorite performance monitor, such as the z/OS Resource Measurement Facility (RMF), you can easily extract your mobile CPU consumption from low-volume SMF records. And, you can do that for any kind of work, including CICS and IMS transaction work (with exploiting levels of CICS and IMS), which opens up new possibilities for capacity planning and charge-back.

This chapter provides an overview of WLM and RMF, and explains how these have been extended for tagging and tracking mobile workloads.

2.1 What is MVS WLM

One of the strengths of the z/OS operating system is the ability to run multiple workloads at the same time within one system or across multiple systems. This can be done while making the best use of the available resources, maintaining the highest possible throughput, and achieving the best possible system responsiveness, even when the systems are nearly 100% utilized. The function that makes this possible is dynamic workload management, which is implemented in the MVS Workload Management (WLM) component of the z/OS operating system.

With WLM, you define performance goals for your work, and assign a business importance to each goal. You define the goals for your work in business terms, such as the average response time you expect for your transactions; WLM decides how much resource, such as CPU and storage, should be given to the work to meet its goals. To do so, WLM is constantly monitoring actual performance against goals; it takes action if work is missing its goals.

The real-time performance data collected through this monitoring is also made available to reporting and monitoring products like the z/OS Resource Measurement Facility (RMF) for integration into low-volume SMF records, and detailed reports.

2.2 WLM service definition constructs

With WLM, you define performance goals for your work based on business needs and current performance. This explicit definition of workloads and performance goals is called a *service definition*. It is done either through a web-based user interface, which is called the Workload Management task in the z/OS Management Facility (z/OSMF), or an ISPF-based user interface, which is called the WLM ISPF application.

There is one service definition for each sysplex. Figure 2-1 shows the WLM service definition constructs that are relevant for measuring mobile workloads.

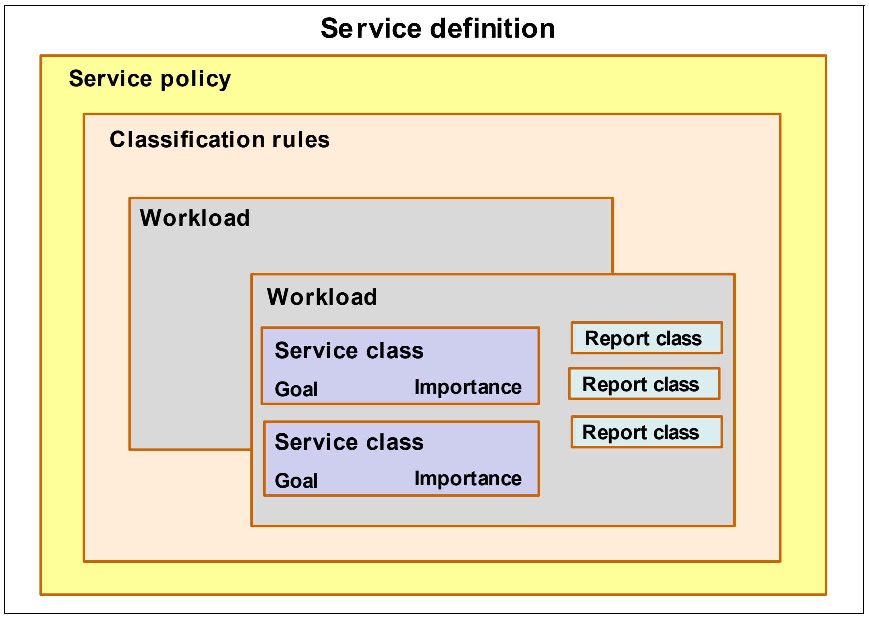


Figure 2-1 WLM service definition constructs

A service definition consists of one or more service policies. A *service policy* is a named set of performance goals for work. It applies to all work in the sysplex. Because processing requirements might change at different times, you can define different performance goals for your work in multiple service policies, and switch between them. For example, you can define different performance goals for your work during day and night shift, and switch between the day shift and night shift service policies at change of shift.

In a service policy, there are multiple workloads. A *workload* is a group of work that is meaningful for reporting and accounting. For example, all the work created by a development group could be a workload, or all the work started by an application, or in a subsystem. Another example of a workload might be one originating from mobile or cloud applications.

Within a workload, you group work with similar performance goals, business importance, and resource requirements into service classes. Service classes are subdivided into one or more periods to allow the goals for work to change when it does not complete in the anticipated time frame. You assign a performance goal to each service class period, and you indicate an importance. The performance goal can be based on response time, or execution velocity. *Importance* is how important it is to your business that the performance goal be achieved. It is used to determine which work should give up resources and which work should receive more when capacity is insufficient for all work to meet its goals.

Classification rules are the rules WLM uses to associate incoming work with service classes. You specify classification rules for a subsystem type, such as JES or DB2, in terms of work qualifiers, such as job name or user ID. Example 2-1 shows a set of WLM classification rules.

Example 2-1 WLM classification rules

```
Modify Rules for the Subsystem Type       Row 1 to 4 of 4
 Command ===> __________________________________________ Scroll ===> CSR

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

 Action codes:   A=After     C=Copy        M=Move     I=Insert rule
                 B=Before    D=Delete row  R=Repeat   IS=Insert Sub-rule
                                                              More ===>
          -------Qualifier--------                -------Class--------
 Action   Type      Name     Start                 Service     Report
                                         DEFAULTS: DISCRETN    BAT00

  ____     1 TN      TV1*     ___                   VEL50       ________
  ____     1 UI      UID1     ___                   VEL30       REPUID1
  ____     1 UI      UID2     ___                   VEL30       REPUID2
```

Example 2-1 shows some classification rules for subsystem type JES. The first rule assigns each job with job name (TN) prefix `TV1*` to service class `VEL50`, while the second rule associates each job running under user ID (UI) `UID1` to service class `VEL30`.

Each subsystem type supports a different set of work qualifiers. See the topic “Defining classification rules for each subsystem” in *z/OS MVS Planning: Workload Management*, SC34-2662-04, and the respective subsystem documentation, for more details.

The attributes of incoming work are compared to these work qualifiers and, if there is a match, the rule is used to assign a service class to the work. WLM uses this service class for management of the work, and reporting of its resource consumption.

Optionally, you can also associate report classes with the work in your classification rules, to analyze the performance of individual workloads running in a service class in more detail, for example, for charge-back. In Example 2-1 on page 11, for instance, the second and third rule both associate work with service class VEL30, but with two different report classes REPUID1 and REPUID2. These allow you to analyze in detail how much of the service consumed by service class VEL30 can be accounted to UID1, and how much to UID2.

2.3 WLM enhancements for measuring mobile workloads

WLM was recently enhanced to more easily classify and measure mobile workloads. To simplify mobile workload reporting, you can specify which work is mobile work that is eligible for Mobile Workload Pricing.

Note: The WLM enhancements are provided with z/OS V2.1 and the WLM APARs OA47042 and OA49728.

Suppose, for example, that you have a classification rule for your banking transactions, and part of the transactions flow in from mobile devices. To differentiate those transactions from “normal” transactions, you can insert a sub-rule as shown in Example 2-2.

Example 2-2   Insert a sub-rule

```
Modify Rules for the Subsystem Type     Row 1 to 12 of 12
 Command ===> ___________________________________________ Scroll ===> CSR

 Subsystem Type . : CICS       Fold qualifier names?   Y  (Y or N)
 Description  . . . CICS rules

 Action codes:   A=After     C=Copy        M=Move     I=Insert rule
                 B=Before    D=Delete row  R=Repeat   IS=Insert Sub-rule
                                                              More ===>
          -------Qualifier--------                 -------Class--------
 Action   Type      Name     Start                  Service     Report
                                          DEFAULTS: CICSDFT     ________
  is__   1 TC       BANKING  ___                    CICSFAST    ________
  ____   1 TC       HR       ___                    CICSSLOW    ________
```

If all mobile banking transactions flowed in through a specific CICS TCP/IP service TCP001, your sub-rule would look like the one shown in Example 2-3.

Example 2-3 Specify a new sub-rule

```
                Modify Rules for the Subsystem Type     Row 1 to 13 of 13
Command ===> ___________________________________________ Scroll ===> CSR

Subsystem Type . : CICS        Fold qualifier names?   Y  (Y or N)
Description  . . . CICS rules

Action codes:   A=After     C=Copy        M=Move     I=Insert rule
                B=Before    D=Delete row  R=Repeat   IS=Insert Sub-rule
                                                             More ===>
         -------Qualifier--------                -------Class--------
Action   Type      Name     Start                  Service     Report
                                        DEFAULTS: CICSDFT     ________
 ____   1 TC       BANKING  ___                   CICSFAST    ________
 ____   2   CT       TCP001  ___                  ________    ________
 ____   1 TC       HR       ___                   CICSSLOW    ________
```

In Example 2-3, a Connection Type (CT) qualifier type is used to classify requests that use a specific TCP/IP port (the TCP/IP port is an attribute of a CICS TCPIPSERVICE). For more information about using Connection Type to classify mobile request see “Using the connection type to classify mobile requests” on page 21.

Important: You do not need to add a service or report class. This means that the service and report class of the parent rule is used, and your existing reporting processes are not affected.

Referring to Example 2-3, when you scroll right on the classification rules panel, you can enter a description for your rules. Scrolling right again leads you to the classification rule attributes panel. Two of them, the Storage Critical and Manage Region attributes, have been available for a long time. They restrict storage donations to other work, and modify transaction response time management for CICS and IMS work.

With the recent enhancements to WLM, a third attribute was added: the Reporting Attribute. With this attribute, you can tag your work (for CICS, IMS, DB2, WebSphere Application Server, or MQ) as mobile work.

Example 2-4 shows the MOBILE Reporting Attribute being set for a subset of banking transactions. In this example, the sub-rule of the first rule tags each transaction with transaction class (TC) BANKING that flows in through TCP/IP service (CT) TCP001 as a mobile transaction.

Example 2-4 The new Reporting Attribute

```
Modify Rules for the Subsystem Type     Row 1 to 13 of 13
 Command ===> ___________________________________________ Scroll ===> CSR

 Subsystem Type . : CICS       Fold qualifier names?   Y  (Y or N)
 Description  . . . CICS rules

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

  ____   1 TC       BANKING  ___        NO        NONE       N/A
  ____   2   CT       TCP001   ___      NO        MOBILE     N/A
  ____   1 TC       HR       ___        NO        NONE       N/A
```

The Reporting Attribute option is supported for all subsystem types that are supplied by IBM. You can specify the following values:

- NONE, for all transactions. This is the default.
- MOBILE, for mobile transactions.
- CATEGORYA, can be used for public cloud transactions.
- CATEGORYB, can be used for public cloud transactions.

The assigned Reporting Attribute is independent from the assigned service and report class. This eliminates the need for introducing new dedicated service and report classes for MWP, and leaves your existing reporting processes unaffected.

Classification with the Reporting Attribute can be based on any work qualifier that is supported for your subsystem types, plus new work qualifiers introduced for CICS (see 3.1.2, “Classifying mobile requests in CICS” on page 21) and IMS (see 3.2.3, “Measuring IMS transaction CPU” on page 26) which are specifically suited for identifying mobile work. As for other classification rules, classification groups can be used to keep the classification rules simple to read, and efficient to check.

WLM aggregates the total and mobile processor consumption[1] for all service and report classes and reports the results in the Workload Activity Collection service IWMRCOLL, one of the WLM APIs, for performance monitors like RMF (see 2.4, “What is the Resource Management Facility (RMF)” on page 15).

With exploiting levels of CICS and IMS, total and mobile processor consumption data[1] is also available for CICS and IMS transaction service and report classes that previously did not report any processor consumption data (see 2.5, “RMF enhancements for measuring mobile workloads” on page 16).

[1] The same processing is done for CATEGORYA and CATEGORYB.

Together with an appropriate performance monitor, such as RMF, this is the first time you get processor consumption data for CICS and IMS transactions from low-volume SMF records. This is invaluable for many reasons, including capacity planning, charge-back, and pricing.

WLM also aggregates the system-wide total and mobile processor consumption data[2], accumulates the values into a rolling 4-hour average, and reports them in the IRARCT data area to performance monitors like RMF.

2.4 What is the Resource Management Facility (RMF)

RMF is an IBM product for performance analysis, capacity planning, and problem determination in a z/OS host environment. Many activities are required to keep a mainframe running smoothly, and to provide the best service on the basis of the available resources and workload requirements. This work is done by systems operators, administrators, programmers, or performance analysts.

RMF can do these tasks:

- Determine that a system is running smoothly.
- Detect system bottlenecks caused by resource contention.
- Evaluate the service an installation provides to different groups of users.
- Identify the delayed workload and the reason for the delay.
- Monitor system failures, system stalls, and failures of selected applications.

RMF provides its benefits through the operation of post-processing and real-time monitoring functions. These functions are based on a set of data gatherers and data services that enable access to all performance-relevant data in a z/OS environment.

RMF has three data gatherers/reporters known as *monitors*:

- Monitor I provides long-term data collection for system workload and system resource utilization. The Monitor I session is continuous. It measures various areas of system activity over a long period of time, and writes the data into SMF record types 70 and 72. The RMF Post Processor is used to create reports from RMF Monitor I data. See 4.7.1, “Measuring the core banking workload” on page 47 for example reports.
- Monitor II provides real-time measurements for viewing the current workloads and solving any existing problems. A Monitor II session can be regarded as a snapshot session collecting data about address space states and resource use.
- Monitor III provides short-term data collection and online reports for continuous monitoring of system status and for solving performance problems, workflow delay monitoring, and goal attainment supervision. See “RMF Monitor III” on page 45 for an example report.

RMF Performance Monitor (RMF PM) is a Java-based monitor which enables you to monitor the performance of your z/OS system from a workstation. RMF PM takes its input data from a data server running on one system in the sysplex. The data server gathers performance data from RMF Monitor III on each MVS image. See “RMF PM” on page 46 for an example RMF PM monitoring view.

[2] The same processing is done for CATEGORYA and CATEGORYB.

2.5 RMF enhancements for measuring mobile workloads

RMF has been enhanced to more easily measure transactional (CICS and IMS) workloads, and to report on mobile transaction CPU consumption.

Note: The RMF enhancements are provided with z/OS V2.1 and RMF APAR OA48466.

In addition to reporting the APPL% field in a Workload Activity report, RMF now reports a new field called Transaction APPL%.

Note: APPL% shows processor utilization based on uniprocessor capacity. This means that the values can exceed 100% in systems with more than one processor.

Transaction APPL% measures CPU time captured for transactions. zIIP and zAAP[3] processor time is also provided for Transaction APPL%. The amount of transaction CPU attributed to mobile transactions is also reported (for an example, see 3.1, "Measuring CICS workloads" on page 20).

Table 2-1 describes the processor types reported in the Transaction APPL% field of an RMF Workload Activity report.

Table 2-1 Transaction APPL%

Processor type	Description
Total CP	Total percentage of general purpose processor time used by transactions.
Total AAP/IIP on CP	Total percentage of general purpose processor time used by transactions eligible to run on specialty processors.
Total AAP/IIP	Total percentage of specialty processor time used by transactions.
Mobile CP	Percentage of general purpose processor time used by transactions classified with reporting attribute MOBILE.
Mobile AAP/IIP on CP	Percentage of general purpose processor time used by transactions classified with reporting attribute MOBILE, eligible to run on specialty processors.
Mobile AAP/IIP	Percentage of specialty processor time used by transactions classified with reporting attribute MOBILE.

For further interpretation of the APPL% and Transaction APPL% measurement fields, see Appendix B, "Analysis of CICS CPU consumption" on page 57.

RMF also reports the long-term average total and mobile consumption and stores it in SMF record type 70, subtype 1, in the CPU control section. SCRT can use this data to apply any subcapacity adjustments for MWP (see 2.6, "What is Sub-Capacity Reporting Tool (SCRT)" on page 17).

[3] z Systems Integrated Information Process (zIIP) and z Systems Application Assist Processor (zAAP)

2.6 What is Sub-Capacity Reporting Tool (SCRT)

The IBM SCRT reports required license capacity for subcapacity-eligible products that run on z/OS. SCRT analyzes a month of data for IBM z Systems environments and produces a subcapacity report. The report indicates the required license capacity (in MSUs) of each subcapacity-eligible product running on z/OS.

Subcapacity products are charged based on the rolling 4-hour average utilization of the LPARs in which the subcapacity products execute. The subcapacity report determines the required license capacity by examining, for each hour in the reporting period:

- The rolling 4-hour average utilization, by LPAR
- The eligible products that were active in each LPAR

Starting with SCRT V23.10.0, SCRT supports MWP. SCRT calculates the rolling 4-hour average of the reported mobile transaction general purpose processor time that is consumed by the MWP defining programs and subtracts 60% of those values from the traditional subcapacity MSUs for all subcapacity-eligible programs running in the same LPAR(s) as the mobile workloads, on an hour-by-hour basis, per LPAR. The program values for the same hour are summed across all of the LPARs in which the program runs to create an adjusted subcapacity value for the program, for the given machine, for each hour. SCRT determines the billable MSU peak for a given program on a machine using the adjusted MSU values.

Processor time for mobile transactions can be collected by using one of these mechanisms:

- Manual mobile transaction reporting through CSV file input:
  - You collect the source data (CPU time) for the mobile transactions that will be used in monthly reporting.
  - You preprocess the mobile transaction data into a predefined format to be loaded into SCRT for each sub-capacity reporting period. The data must consist of general purpose processor CPU seconds for mobile transactions, summarized by hour, by LPAR, for all CPCs that process mobile transactions.
- Starting with SCRT V23.13.0, mobile transaction reporting is possible using WLM mobile workload classification:
  - You classify mobile workloads using WLM classification rules.
  - Workloads that are classified as mobile in the WLM service class do not require any manual aggregation or preprocessing of data.

Note: This Redpaper publication focuses on using WLM to measure mobile workloads.

zWPC requires that you identify public cloud transactions. As with MWP, you can use two SCRT mechanisms to accomplish this:

- Use an INPUTCSV file.
- Starting with SCRT V24.10.0, public cloud transactions can be reported by using WLM.

Note: When you provide zWPC data to SCRT through WLM workload classification, you must use the ASSIGN ZWPC control statement in the SPECIAL DD to instruct SCRT as to which categorization was used for cloud transaction data (CATEGORYA or CATEGORYB).

For more information about SCRT, see *Using the Sub-Capacity Reporting Tool*, SC23-6845:

http://ibm.biz/SCRTguide

3

Subsystem considerations

This chapter looks at specific subsystem considerations for CICS, IMS, DB2, and WebSphere Application Server. For each of these subsystems, the various options for classifying mobile requests and using Workload Management (WLM) to measure mobile workloads are reviewed.

3.1 Measuring CICS workloads

WLM is used to set the service levels that you want your CICS regions and transactions to achieve. It is also a useful method for reporting CICS CPU usage, transaction rates, and response times using Resource Measurement Facility (RMF).

When work enters a CICS system, it is classified into service classes and report classes based on a set of rules associated with its subsystem type.

This section introduces recent enhancements to CICS Transaction Server (TS) 5.3 that make CICS workloads easier to measure. It also discusses options for classifying and measuring CICS transactions that are initiated by mobile devices.

3.1.1 Enhancements to CICS TS V5.3

Together with the WLM enhancements described in 2.3, “WLM enhancements for measuring mobile workloads” on page 12, transactions running in CICS TS V5.3 can now be classified into service and report classes, which directly measure and report their CPU consumption.

CICS is enhanced to continuously measure every transaction’s CPU consumption without the need to activate CICS Performance Class Monitoring. These measurements are aggregated by z/OS WLM to provide enhanced reporting in System Management Facilities (SMF) records and RMF reports, as Figure 3-1 illustrates.

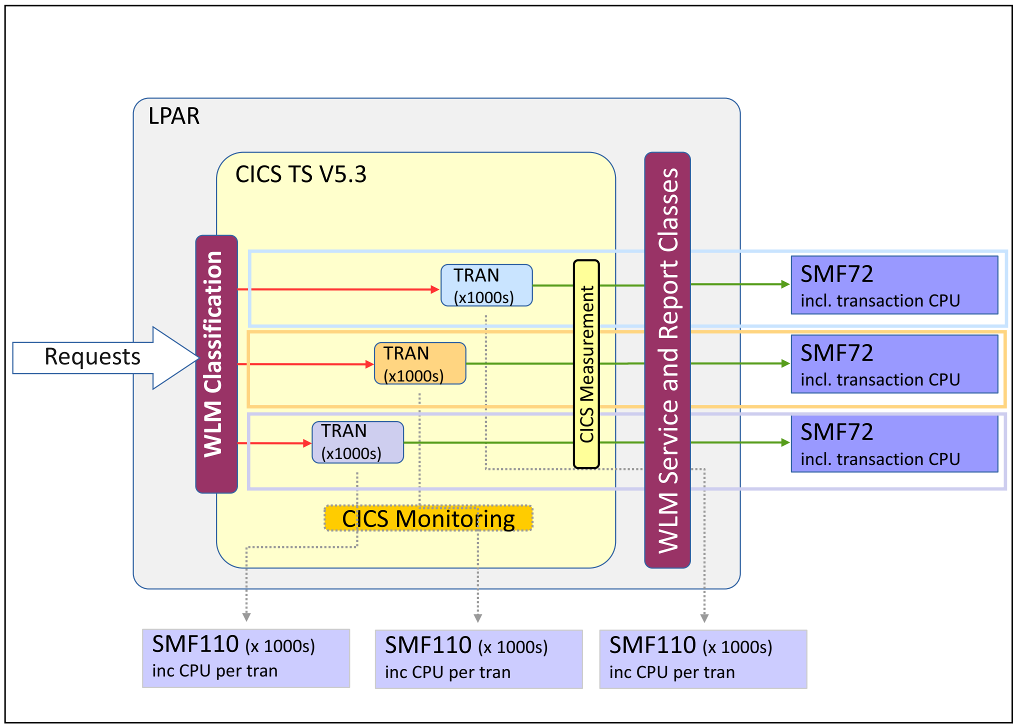

LPAR
CICS TS V5.3
Requests
WLM Classification
TRAN (x1000s)
TRAN (x1000s)
TRAN (x1000s)
CICS Measurement
WLM Service and Report Classes
SMF72 incl. transaction CPU
SMF72 incl. transaction CPU
SMF72 incl. transaction CPU
CICS Monitoring
SMF110 (x 1000s) inc CPU per tran
SMF110 (x 1000s) inc CPU per tran
SMF110 (x 1000s) inc CPU per tran

Figure 3-1 Measuring CICS transaction CPU

Figure 3-1 shows how CICS TS V5.3 now passes the CPU times used by transactions to z/OS WLM. CPU measurements for workloads classified in this way help make collecting the data required for measuring the CPU consumption of CICS workloads simpler and more efficient. This enables CPU time collection for groups of transactions based on WLM classifications.

CICS TS also introduces new WLM qualifier types that can be used to classify CICS requests based on TCP/IP service name and transaction class.

The WLM MOBILE Reporting Attribute can be specified for a CICS classification rule, which then enables WLM to accumulate all the CPU time for this classification and report it directly as mobile CPU time, eligible for Mobile Workload Pricing. The CICS transaction CPU attributed to mobile-initiated transactions is then also displayed in RMF Postprocessor reports.

3.1.2 Classifying mobile requests in CICS

Classification rules categorize work into service classes and, optionally, report classes, based on qualifier types. Table 3-1 describes some of the WLM qualifier types that can be used to classify CICS requests.

Table 3-1 WLM qualifier types to classify CICS requests

WLM qualifier type	Description
CT	Connection type. The name of the TCP/IP service that received the request for this transaction (new in CICS TS V5.3).
LU	The IBM VTAM® LUNAME of the system that issued the request.
SI	Subsystem instance. The CICS APPLID.
TC	Transaction class. The name of the transaction class to which this transaction belongs (new in CICS TS V5.3).
TN	Transaction identifier (trans ID).
UI	User ID. The user ID assigned to the CICS transaction.

Note: For each qualifier, you can specify classification groups by adding a “G” to the type abbreviation. For example, a transaction name group will be TNG.

The qualifier type used to classify a mobile request will depend on the specific environment and the preferred mobile tagging and tracking mechanism. The following examples show how each qualifier in Table 3-1 can be used to classify a request as a mobile request.

Using the connection type to classify mobile requests

A CICS TCPIPSERVICE resource defines which TCP/IP services are to use CICS internal sockets support. The TCPIPSERVICE defines transport requirements like the TCP/IP port that is listened on, whether TLS/SSL is used, and so on.

A TCPIPSERVICE definition is required when using CICS SOAP web services or an IPIC connection from CICS Transaction Gateway.

Figure 3-2 shows a CICS web services scenario where a TCPIPSERVICE is used to classify CICS mobile requests.

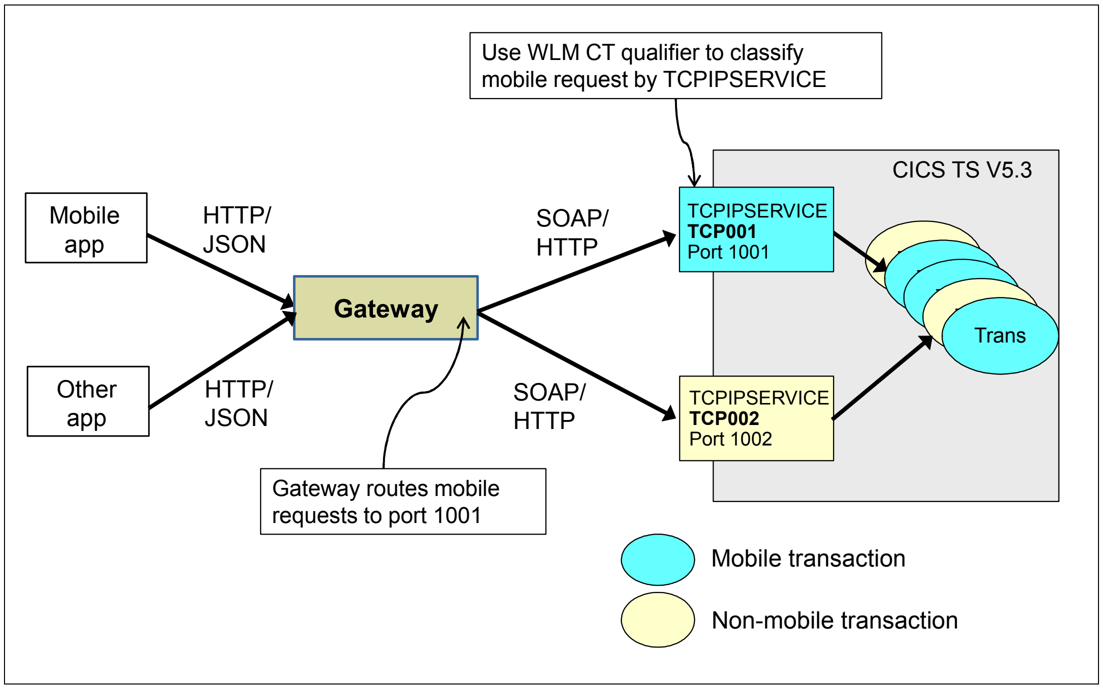

Use WLM CT qualifier to classify mobile request by TCPIPSERVICE
CICS TS V5.3
Mobile app
HTTP/ JSON
SOAP/ HTTP
TCPIPSERVICE TCP001 Port 1001
Gateway
Trans
Other app
HTTP/ JSON
SOAP/ HTTP
TCPIPSERVICE TCP002 Port 1002
Gateway routes mobile requests to port 1001
Mobile transaction
Non-mobile transaction

Figure 3-2 Using TCPIPSERVICE to classify CICS mobile request

In this scenario, a gateway sends mobile initiated requests to a specific port (1001). This port is defined as part of a CICS TCPIPSERVICE resource (TCP001). A WLM classification rule using a CT qualifier type (see Example 2-4 on page 14) is then used to classify the request as a mobile request.

How does the gateway know that the request comes from a mobile device? This can be determined by the network connection, or information in the HTTP request header. One way to establish the client device type is to inspect the HTTP User-Agent header to detect the OS of the device. If the header contains one of the following strings, the request can be tagged as mobile:

```
"iOS" || "iPhone" || "iPad" || "iPod" || "Android" || "BlackBerry" || "BB10" ||
"Windows Phone"
```

Using the LUNAME to classify mobile requests

A terminal LUNAME will unlikely be used to classify mobile requests.

Using the subsystem instance to classify mobile requests

Classifying requests based on the CICS subsystem instance (SI qualifier type) might be interesting when you know that all requests processed by a CICS region are from mobile devices. The value specified for the subsystem instance should be the CICS APPLID. This is a simple way to classify mobile requests based on the CICS region that processes the request.

Using the transaction class to classify mobile requests

Transactions that are defined as belonging to a transaction class are subject to scheduling constraints before they are allowed to execute. If transactions belonging to an active transaction class are already running, any new transactions are queued. You can use the MAXACTIVE attribute to specify the maximum number of transactions that you want to run for a specific transaction class.

To avoid problems with a single CICS service from affecting other services, transaction classes can be used to isolate a group of services. SOAP or JSON-based services can be grouped into a number of transactions classes if different transaction IDs are associated with specific service requests. Each transaction class can then be used to control the desired maximum number of concurrently active service requests.

In CICS TS V5.3, a transaction class name can also be used in a WLM classification rule. A WLM classification rule using a TC qualifier type can be used to classify the request as a mobile request.

For an example of classifying mobile requests using a transaction class (TC qualifier type), see 4.5.4, “Classifying CICS Mobile requests” on page 44.

Using the transaction ID to classify mobile requests

Probably the most common qualifier used to classify CICS requests is the transaction identifier (trans ID). This can be a simple way to classify mobile requests if mobile-initiated transactions run with specific trans IDs.

For an example of classifying mobile requests using a transaction identifier (TN qualifier type), see 4.5.4, “Classifying CICS Mobile requests” on page 44.

Using the user ID to classify mobile requests

Another common qualifier used to classify CICS requests is the user ID. This can also be a way to classify mobile requests if mobile-initiated transactions execute with a specific IBM RACF® user ID.

3.1.3 Measuring CICS transaction CPU

You can use two main ways to measure CICS performance and measure CPU consumption:

- CICS monitoring collects data about the performance of all user and CICS transactions during online processing for later offline analysis. Performance class data is detailed transaction-level information, such as the processor and elapsed time for a transaction. CICS writes at least one performance monitoring record for each transaction.
- MVS WLM provides workload activity reporting by service class, report class, or both.

Until the recent WLM and CICS TS V5.3 enhancements, transaction CPU consumption was not present in the performance data reported by CICS to WLM, so the only way to record transaction CPU was to enable CICS monitoring performance class data. With the enhancements, CICS transaction CPU consumption can be more easily reported by RMF. CICS monitoring remains an important tool for capturing detailed performance data such as I/O and other wait times.

An important factor when using WLM service and report classes for reporting transactions through RMF is that CICS transactions are reported only in the regions in which they were originally initiated, classified, and then ended. This is highlighted in Figure 3-3 on page 24.

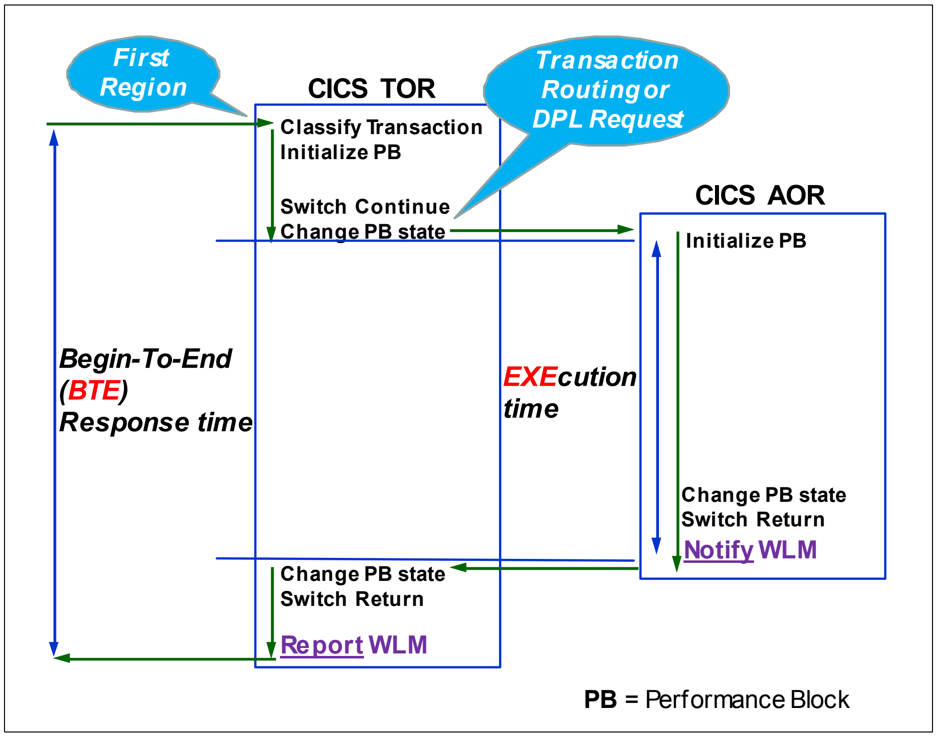


Figure 3-3 How WLM tracks transactions across multiple CICS TS V5.3 regions

Important: CICS transactions are normally classified in *terminal-owning regions* (TORs). Transactions are then normally routed to an *application-owning region* (AOR). In this case, the request is not reclassified in the application-owning region.

CICS regions register as work managers to WLM. When a request is first received by CICS (normally in a TOR), it is classified by WLM and then CICS creates a performance block (PB) to record performance information for the task. If the request is passed to an AOR, WLM information is passed in a Service Report Class token. A new PB is created in the AOR, but the transaction is not reclassified by WLM. When the task completes in the AOR, CICS notifies the task start and end times and CPU consumption to WLM. Similarly when the task completes in the TOR, CICS reports to WLM.

WLM continuously collects data by sampling PBs. It creates a picture of the CICS topology, including the server address spaces and the service classes of the requests. It uses this topology to manage the server address spaces based on the goal achievement of the service classes for the work requests.

For CICS TS 5.3, a service class and report class for a CICS classification rule show the CPU time for the completed transactions (see Example 4-10 on page 48). The CPU time for transactions in execution states in the AOR are accumulated back to the TOR where the transaction is classified.

3.2 Measuring IMS workloads

WLM is used to set the service levels that you want your IMS regions and transactions to achieve. It is also a useful method for reporting IMS CPU usage, transaction rates, and response times when using a performance monitor like RMF.

When work enters an IMS system, it is classified into service classes and report classes based on a set of rules associated with its subsystem type.

This section introduces recent enhancements to IMS V14 that can help you more easily measure IMS workloads. This section also discusses options for classifying and measuring IMS transactions that are initiated by mobile devices.

3.2.1 Enhancements to IMS V14

Together with the WLM enhancements, described in 2.3, "WLM enhancements for measuring mobile workloads" on page 12, transactions running in IMS V14 can now be classified into service and report classes, which directly measure and report their CPU consumption.

Note: The IMS enhancements are provided with IMS V14 and IMS APARS PI46933 and PI51948.

IMS is enhanced to continuously measure every transaction's CPU consumption without the need to collect, analyze, and interpret statistical data in IMS logs. These measurements are aggregated by z/OS WLM to provide enhanced reporting in SMF records and RMF reports. This enhancement is similar to the enhancement introduced in CICS TS V5. 3 (see 3.1.1, "Enhancements to CICS TS V5.3" on page 20).

IMS V14 also introduces new WLM qualifier types that can be used to classify IMS requests based on TCP/IP Service name and the name of the transaction pipe (TPIPE).

The WLM MOBILE Reporting Attribute can be specified for an IMS classification rule, which then enables WLM to accumulate all the CPU time for this classification and report it directly as mobile CPU time eligible for Mobile Workload Pricing. The IMS transaction CPU attributed to mobile-initiated transactions is then also displayed in RMF Postprocessor reports.

3.2.2 Classifying mobile requests in IMS

Classification rules categorize work into service classes and, optionally, report classes, based on qualifier types. Table 3-2 describes some of the WLM qualifier types that can be used to classify IMS requests.

Table 3-2 WLM qualifier types to classify IMS requests

WLM qualifier type	Description
CT	Connection type. The port number of the TCP/IP service that received the request for this transaction, which is new in IMS V14.
CTN	Client transaction name. The name of the transaction pipe (TPIPE), which is new in IMS V14.
LU	The VTAM LUNAME from where the request originated.
SI	Subsystem instance. The IMS subsystem name.
TC	The CLASS keyword on the PGMTYPE=parameter in the APPLCNT macro.
TN	Transaction identifier. The CODE= parameter on the IMS TRANSACT macro.
UI	User ID. The user ID assigned to the IMS transaction.

Note: For each qualifier, you can specify classification groups by adding the letter "G" to the type abbreviation. For example, a transaction name group will be TNG.

The qualifier type used to classify a mobile request will depend on the specific environment and the preferred mobile tagging and tracking mechanism.

3.2.3 Measuring IMS transaction CPU

You can use two main ways to measure IMS performance and measure CPU consumption:

- IMS logs statistical data about transactions in the X'56FA' and X'07' log records. You can use these log records to collect, analyze, and interpret statistical data for general business purposes.
- MVS WLM provides workload activity reporting by service class, report class, or both.

Until the recent WLM and IMS V14 enhancements, transaction CPU consumption was not present in the performance data reported by IMS to WLM, so the only way to record transaction CPU was to analyze IMS log records. With the enhancements, IMS transaction CPU consumption can be more easily reported. WLM aggregates the total and mobile processor consumption for the service and report classes defined for IMS address spaces and transactions. This CPU is then reported in RMF Workload Activity reports in the same way as for CICS.

3.3 Measuring DB2 workloads

WLM is used to set service levels for DB2 address spaces and database access threads that are created to access data on behalf of a requester. A database access thread is created when an SQL request is received from the requester.

Note: When a CICS task executes an SQL request, a large proportion of the CPU consumed by the execution of the request is actually attributed to the CICS region or transaction. Therefore, this section focuses on the measurement of requests from remote clients that use the distributed data facility (DDF).

3.3.1 DB2 support for WLM enclaves

A WLM *enclave* is a construct that represents a transaction or unit of work. Enclaves are a method of managing mainframe transactions for non-traditional workloads. You can think of an enclave as an anchor point for resource accumulation regardless of where the transaction is executing.

With traditional mainframe transactions, mapping the CP resources consumed to the actual transaction is relatively easy. But with non-traditional workloads, for example, web transactions, it is more difficult because the transaction can span several platforms. Enclaves are used to overcome this difficulty by correlating closely to the user's view of the transaction.

An enclave can consist of many pieces spread over many server address spaces. So even though a non-traditional transaction can consist of multiple “pieces” spanning many address spaces, and can share those address spaces with other transactions, an enclave gives you control over the non-traditional workload. And WLM can be used to more effectively manage non-traditional workload through the enclave.

Most DB2 workloads are already classified using the calling address space or transaction service and report class (for example, CICS). However, this is not the case with DB2 DRDA (IBM Distributed Relational Database Architecture™) enclaves. DB2 requests coming into the system through DRDA over a TCP/IP network are dispatched on z/OS within enclave service request blocks (SRBs). Such requests are classified by WLM and the WLM MOBILE Reporting Attribute can be used to classify the requests as mobile workload.

3.3.2 Classifying DB2 DDF requests

When using JDBC type 4 remote connections, distributed applications are connecting to DB2 distributed data facility (DDF). Most of the time, these distributed workloads are classified in one single default WLM service and report class; however, WLM already has the tools for being able to identify and classify distributed threads to either mobile or non-mobile service and report classes using classification attributes (see Figure 3-4).

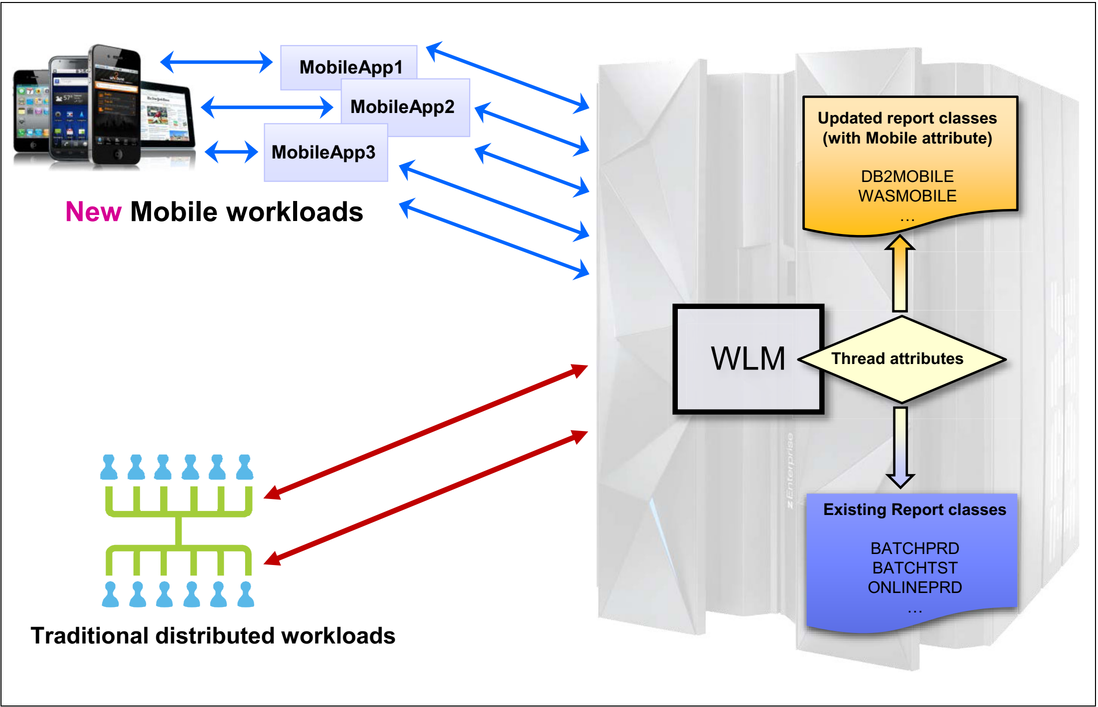


Figure 3-4 Classifying DB2 DDF requests

The objective is to use one or a combination of several qualifier types to be able to identify the mobile-related workload. Example qualifier types are subsystem name, the user ID used to establish the connection, program name, user name, calling IP address, and also the plan name or stored procedure used.

Table 3-3 describes some of the WLM qualifier types that can be used to classify DB2 DDF requests.

Table 3-3 WLM qualifier types to classify DB2 DDF requests

WLM qualifier type	Description
AI	Accounting information. The value of the DB2 accounting string associated with the DDF server thread.
CAI	The client accounting information. This qualifier contains the value of the user-specified accounting string suffix associated with a DDF server thread (new in DB2 V11).
CI	The DB2 correlation ID of the DDF server thread.
CIP	The client IP address (new in DB2 V11).
CN	The DB2 collection name of the first SQL package accessed by the DRDA requester in the unit of work.
CUI	The client user ID (new in DB2 V11).
CWN	The client workstation name (new in DB2 V11).
CTN	The client transaction or application name (new in DB2 V11).
LU	The VTAM LUNAME of the system that issued the SQL request.
NET	The VTAM NETID of the system that issued the SQL request.
PC	Process name. This attribute can be used to classify the application name or the transaction name.
PK	The name of the first DB2 package accessed by the DRDA requester in the unit of work.
PN	The DB2 plan name of the requesting application.
PR	Stored procedure name. This classification only applies if the first SQL statement from the client is a CALL statement.
SI	Subsystem instance. The DB2 server’s z/OS subsystem name.
SPM	Subsystem parameter. This qualifier has a maximum length of 255 bytes. The first 16 bytes contain the client's user ID. The next 18 bytes contain the client’s workstation name. The remaining 221 bytes are reserved.
SSC	Subsystem collection name. When the DB2 subsystem is a member of a DB2 data sharing group, this attribute can be used to classify the data sharing group name.
UI	User ID. The DDF server thread's primary authorization ID, after inbound name translation, which occurs only with SNA DRDA connections.

Note: For many of the qualifiers, you can specify classification groups by adding a “G” to the type abbreviations. For example, a user ID group will be UIG.

The qualifier used to classify a mobile request will depend on the specific environment and the preferred mobile tagging and tracking mechanism. For example, if mobile requests are sent from a particular IP address, then using the CIP qualifier type is appropriate. In some cases, the DB2 correlation ID can be a good option also (see “Using the DB2 correlation ID to classify mobile requests”).

Using the DB2 correlation ID to classify mobile requests

Example 3-1 shows the DB2 correlation ID being used to classify mobile requests.

Example 3-1 Classify mobile workload using DB2 correlation id

```
Subsystem Type . : DDF       Fold qualifier names?   Y  (Y or N)
Description  . . . DRDA Distributed Transactions

Action codes:   A=After     C=Copy        M=Move     I=Insert rule
                B=Before    D=Delete row  R=Repeat   IS=Insert Sub-rule
                                                             More ===>
         -------Qualifier--------               -------Class--------   Reporting
Action   Type      Name     Start                  Service     Report   Attribute
                                        DEFAULTS: DDFLOW      RDDFDEF   NONE
 ____   1 SI        DBP*     ___                  DDFSTD      RDDFDBP   NONE
 ____   2   CI        MOB*     ___                DDFSTD      RDDFDBP   MOBILE
```

In Example 3-1, a combination of subsystem instance (SI) and correlation ID (CI) are used to classify DDF requests to subsystem DBP1. Mobile DDF requests with correlation ID of `MOB*` are assigned the MOBILE reporting attribute

How to set the DB2 correlation ID in the DDF client

For a JDBC type 4 connection, DB2 DDF assigns the client correlation id based on the program name set by the DDF client application. The default program name is **`db2jcc_application`**, which then becomes the correlation ID in DB2 for z/OS. To classify a request based on correlation ID, the client application must set the program name. This can be done in one of two ways, as shown in the next two examples.

Example 3-2 shows a Java snippet from a program that sets the program name MOBILEAPP1 as a JDBC property.

Example 3-2 Setting DB2 program name as JDBC property

```
…
Class.forName("com.ibm.db2.jcc.DB2Driver");
Properties props = new Properties();
props.put("user", "mobuser1");
props.put("password", “xxxxxxxx");
props.put("clientProgramName", "MOBILEAPP1");
Connection conn = DriverManager.getConnection("jdbc:db2://localhost:446/RDBNDBS1",
props);
…
```

Example 3-3 shows a Java snippet from a program that sets the program name MOBILEAPP2 in a DB2 data source.

Example 3-3 Setting DB2 program name in a DB2 data source

```
…
Connection conn = null;
DB2SimpleDataSource ds = new com.ibm.db2.jcc.DB2SimpleDataSource();
ds.setDriverType(4);
ds.setServerName("localhost");
ds.setPortNumber(446);
ds.setDatabaseName("RDBNDBS1");
ds.setUser("mobuser2");
ds.setPassword("xxxxxxxx");
ds.setClientProgramName("MOBILEAPP2");
conn = ds.getConnection();
…
```

3.3.3 Measuring DB2 DDF CPU

Measuring DB2 DDF CPU is possible by using either of the following ways:

- WLM report classes (or service classes)
- SMF101 accounting data (when accounting trace is active in a DB2 environment)

Starting a DB2 accounting trace implies a CPU overhead (typically 2 - 3% overhead depending on the detail level of the trace) and requires the collection of a large volume of data. Therefore, the suggested way is to use WLM to classify and measure DB2 DDF CPU. However, DB2 accounting remains an important tool for capturing detailed performance data such as I/O and other wait times.

Example 3-4 shows an RMF Postprocessor report for the report class RDDFMOB, defined in Example 3-1 on page 29.

Example 3-4 RMF Postprocessor Workload Activity report for report class RDDFMOB

```
REPORT BY: POLICY=POLZOS                     REPORT CLASS=RDDFDBP
                                             DESCRIPTION =DDF Default report class

 -TRANSACTIONS-    TRANS-TIME HHH.MM.SS.TTT    SERVICE TIME    ---APPL %---
 AVG       7.76    ACTUAL                 3    CPU   42.827    CP     29.37
 MPL       7.76    EXECUTION              3    SRB    0.000    AAPCP   0.00
 ENDED    51467    QUEUED                 0    RCT    0.000    IIPCP   0.03
 END/S   857.83    R/S AFFIN              0    IIT    0.000
 £SWAPS       0    INELIGIBLE             0    HST    0.000    AAP      N/A
 EXCTD        0    CONVERSION             0    AAP      N/A    IIP    25.85
 AVG ENC   7.76    STD DEV              319    IIP   25.208
 REM ENC   0.00
 MS ENC    0.00

 TRANSACTION APPL% :   TOTAL :   CP  29.37   AAP/IIP ON CP   0.03   AAP/IIP  25.85
                       MOBILE :  CP  21.79   AAP/IIP ON CP   0.02   AAP/IIP  18.90
```

The report shows that an average of 857 DB2 DDF requests were executed per second, with an average response time of 3 ms. It also shows that 29.37% of a CP was consumed by these transactions, of which 21.79% of a CP is reported as mobile CPU for this report class.

3.4 Measuring WebSphere workloads

WebSphere Application Server for z/OS exploits WLM services to process incoming requests in the following ways:

- WLM queuing services are used to manage the execution priority of address spaces and work requests.
- WLM enclaves are used to manage performance of WebSphere transactions across multiple address spaces.
- WLM Application Environments are used to dynamically manage the number of servant address spaces.

3.4.1 WebSphere support for WLM enclaves

WLM enclave services allow performance management of a transactions across multiple WebSphere address spaces. WebSphere Application Server for z/OS uses an independent enclave type that reports on resource consumption based on a performance goal that is associated to the transaction, unrelated to the performance goals of the address spaces in which the enclave dispatchable units execute.

The WebSphere controller region creates the enclave and associates the transaction to this classified enclave. Then, the transaction is queued, waiting to be served by an available thread in a WebSphere servant region.

WebSphere requests coming into the controller region over various protocols can be classified by WLM and the WLM MOBILE Reporting Attribute can be used to classify the requests as mobile workload.

3.4.2 Classifying WebSphere requests

WebSphere workloads are managed by two types of WLM classification:

- STC type classification

  WebSphere address spaces (controller and servant regions) are classified as STCs in order to manage all of the server activity that does not relate to specific transactions, for example, garbage collector activity and connection pool management.
- CB type classification

  Every request to a WebSphere is dispatched on a UNIX System Services thread and executes under a WLM enclave that is classified as a CB workload transaction.

  The classification of each transaction is managed by the controller region. The controller region acts as a queuing manager that queues work requests to workload management for execution in servant address spaces. Workload management maintains the queues for passing work requests from the controller region to each servant region.

  The controller region listens for work requests and puts them on the Workload Management queue. The Workload Management component of z/OS dispatches the work to the servant region according to the WLM policy specified by the work identifier.

Note: When WebSphere applications create their own threads outside of WebSphere control, these threads are named `unmanaged threads` and because they are not running inside transaction enclaves they are reported in STC activity.

The processor activity of WebSphere transactions is reported to WLM and RMF and so can be measured in RMF Workload Activity reports for service classes and report classes. With the recent WLM enhancements, you can now also tag WebSphere requests with MOBILE Reporting Attribute.

Table 3-4 describes some of the WLM qualifier types that can be used to classify WebSphere requests.

Table 3-4 WLM qualifier types to classify WebSphere requests

WLM qualifier type	Description
CN	The collection name. If the server is clustered, this value is the short name for the cluster. If the server is not clustered, this value is the value specified on the server custom property ClusterTransitionName (which is also the application environment name).
SI	Subsystem instance. This is the short name for the server (displayed at startup in "server_specific_short_name"). This is usually not very useful because you cannot control which server instance runs a transaction within a cluster.
TC	Transaction class. This can be assigned using a workload classification document file (see "Defining a transaction class" on page 32).
UI	User ID. The user ID is assigned to the transaction.

The qualifier used to classify a mobile request will depend on the specific environment and the preferred mobile tagging and tracking mechanism. A TC qualifier type with a workload classification document file offers the greatest flexibility.

Defining a transaction class

A transaction class provides a granular approach for classifying WebSphere transactions. For instance, you can classify requests for different applications inside the same server or requests for the same application depending on which HTTP channel is used.

A workload classification document file is an XML file in which you classify incoming work requests and assign them to a transaction class (TCLASS). The TCLASS value is passed to WLM. WLM uses the TCLASS value to classify the inbound work requests and to assign a service class or a report service class to each request.

The following steps are involved in using transaction class as a WLM qualifier type:

1. The WebSphere administrator creates a workload classification document file with the classification criteria and the names of the transaction classes to pass to WLM.
2. The WLM administrator then creates the CB classification rules that match the workload classification document file and assign the appropriate service and report classes.

These inbound transport types can be defined in a workload classification document file:

- HTTP requests
- IIOP requests
- MDB requests
- SIP requests
- SIB requests
- Optimized Local Adapter Inbound requests

Each type of transport supports its own specific classification criteria. For example, the following classification criteria can be used to classify HTTP requests:

- Virtual host name. The host name in the HTTP header to which the inbound request is being sent.
- Port number. WebSphere Application Server HTTP listener port that receives the request.
- Uniform Resource Identifier (URI); for example, `/myApp/*`.

Classification criteria for multiple transport protocols can be defined in the same workload classification document file.

Note: A transaction class for WebSphere Liberty is enabled using the zosWlm-1.0 feature. Only HTTP requests can be classified.

Using a transaction class to classify mobile requests

Mobile requests will most likely use an HTTP transport and therefore can be classified using a combination of the attributes host, port, and URI.

The host name sent by client application can be a good way to differentiate mobile requests from non-mobile requests. The advantage of host name as a selection criteria is that it allows a different report class to be assigned for requests to the same application accessed over the same port and URI. For example, requests for service `http://mobilehost/app` can be classified differently than requests for service `http://webhost/app` without changing anything in the application or application infrastructure. Requests for service `http://mobilehost/app` can then also be assigned the MOBILE Reporting Attribute.

Example 3-5 shows a workload classification document file that can be used to classify mobile requests.

Example 3-5 Sample workload classification document file

```
Subsystem Type . : DDF        Fold qualifier names?   Y  (Y or N)
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE Classification SYSTEM "Classification.dtd">
<Classification schema_version="1.0">
  <InboundClassification  type="http"
                           schema_version="1.0"
                           default_transaction_class="TCWEBD">
      <http_classification_info transaction_class="TCLWEB"
                                host="webhost"
                                description="Web requests"/>
      <http_classification_info transaction_class="TCLMOB"
                                host="mobilehost"
                                description="Mobile requests"/>
  </InboundClassification>
</Classification>
```

In Example 3-5, requests for service `http://webhost/app` are assigned to transaction class TCLWEB; requests for service `http://mobilehost/app` are assigned to transaction class TCLMOB.

Example 3-6 shows the corresponding WLM classification rules for classifying various types of WebSphere HTTP requests.

Non-mobile requests to the server cluster `AI*`, which are associated with transaction class TCLWEB, are assigned service class CBMED and report class RCBWEB. Mobile requests to the server cluster `AI*`, which are associated with transaction class TCLMOB, are also assigned service class CBMED but a specific report class RCBMOB.

Example 3-6 WebSphere transaction classification rules

```
Subsystem Type . : CB          Fold qualifier names?   Y  (Y or N)
Description  . . . Component Broker

Action codes:   A=After     C=Copy        M=Move     I=Insert rule
                B=Before    D=Delete row  R=Repeat   IS=Insert Sub-rule
                                                             More ===>
         -------Qualifier--------                -------Class--------
Action   Type      Name     Start                 Service     Report
                                        DEFAULTS: CBSTD       RCBDFLT
 ____  1 CN        AI*      ___                   CBSTD       RCBAI
 ____  2   TC        TCLWEB   ___                 CBMED       RCBWEB
 ____  2   TC        TCLMOB   ___                 CBMED       RCBMOB
 ____  1 CN        *        ___                   CBSTD       RCBSTD
```

Note: Example 3-6 defines specific report classes to monitor web and mobile requests. If you do not define report classes for sub-rules, the parent report class is used.

Now that rules for classifying the different types of WebSphere request are defined, the MOBILE Reporting Attribute can be assigned to mobile-initiated transactions, as shown in Example 3-7.

Example 3-7 Assigning the MOBILE Reporting Attribute to WebSphere requests

```
         -------Qualifier--------      Storage   Reporting  Manage Region
Action   Type      Name     Start      Critical  Attribute  Using Goals Of

 ____  1 CN        AI*      ___        N/A       NONE       N/A
 ____  2   TC        TCLWEB   ___      N/A       NONE       N/A
 ____  2   TC        TCLMOB   ___      N/A       MOBILE     N/A
 ____  1 CN        *        ___        N/A       NONE       N/A
```

These WLM classification rules allow you to classify requests for `http://mobilehost/app` service as mobile transactions. And, because the MOBILE Reporting Attribute is specified for these requests, WLM will also include the GP processor usage of these requests in the aggregated system-wide mobile CPU consumption in order to calculate a rolling 4-hour average that can be used for MWP.

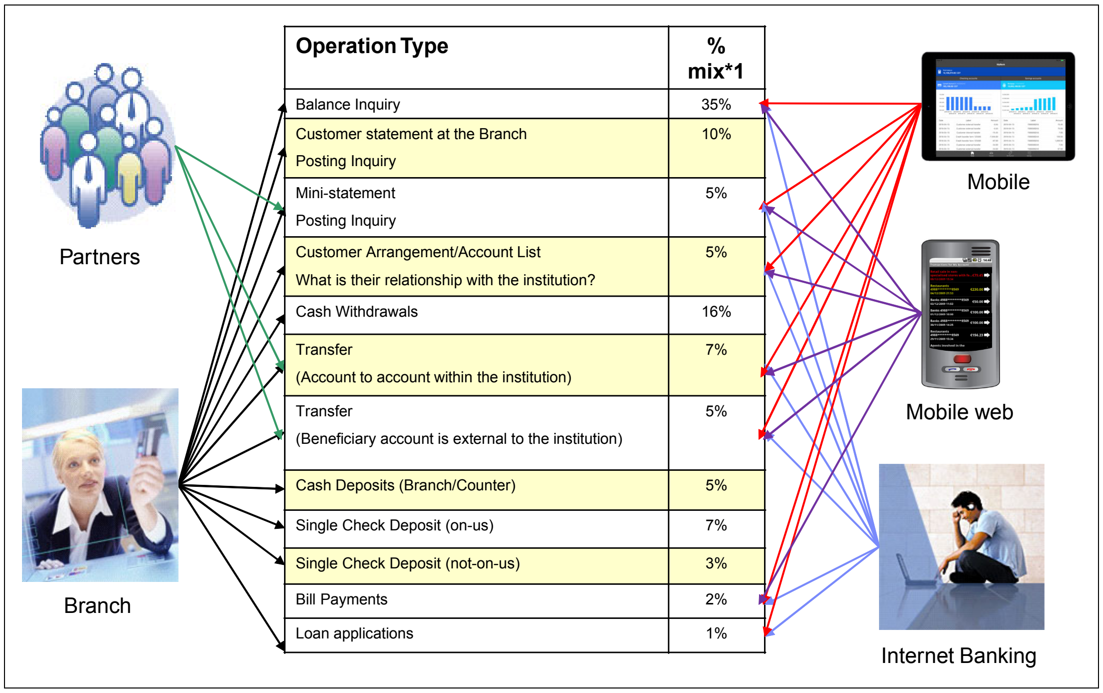
4 Example scenario

Fictional Bank A is a retail bank, offering a custom suite of retail and business banking and credit card services. The bank's modern core banking systems are based on CICS and DB2 applications, developed in COBOL and Java. There are multiple access points to these systems, for example using CICS Transaction Gateway and web services; however, the predominant access method for new mobile applications is through REST services.

Bank A has witnessed a significant growth in mobile initiated transactions, and wants to monitor and measure the mobile channels more closely. This section, which outlines how Bank A uses MVS WLM to do this, focuses on mobile access to the CICS core banking systems.

4.1 Core banking channels

Online banking transactions are processed by the CICS core banking application. Figure 4-1 shows the list of banking operations and the primary channels supported.



Operation Type	% mix*1
Balance Inquiry	35%
Customer statement at the Branch Posting Inquiry	10%
Mini-statement Posting Inquiry	5%
Customer Arrangement/Account List What is their relationship with the institution?	5%
Cash Withdrawals	16%
Transfer (Account to account within the institution)	7%
Transfer (Beneficiary account is external to the institution)	5%
Cash Deposits (Branch/Counter)	5%
Single Check Deposit (on-us)	7%
Single Check Deposit (not-on-us)	3%
Bill Payments	2%
Loan applications	1%

Figure 4-1 Core banking channels

In the bank's test systems, IBM Rational® Performance Tester is used to inject a typical day's online activity with a transaction mix that covers balance inquiries, statement requests, transfers, check deposits, and so on. Figure 4-1 shows the transaction mix and the various channels that are simulated: Branch, Partner, Internet, Mobile web, and Mobile app.

The initial mobile banking services offered by the bank reused the existing Internet banking channel. This channel is referred to here as the *Mobile web* channel. In recent years, however, Bank A placed customer service as its top priority to encourage growth and invested in new mobile apps. These mobile services need to access the core banking systems, a master data management (MDM) system and a business rules engine. This channel is referred to here as the *Mobile* channel.

4.2 Architecture

Multiple ways are available to access the CICS core banking application. Figure 4-2 shows the core banking integration architecture.

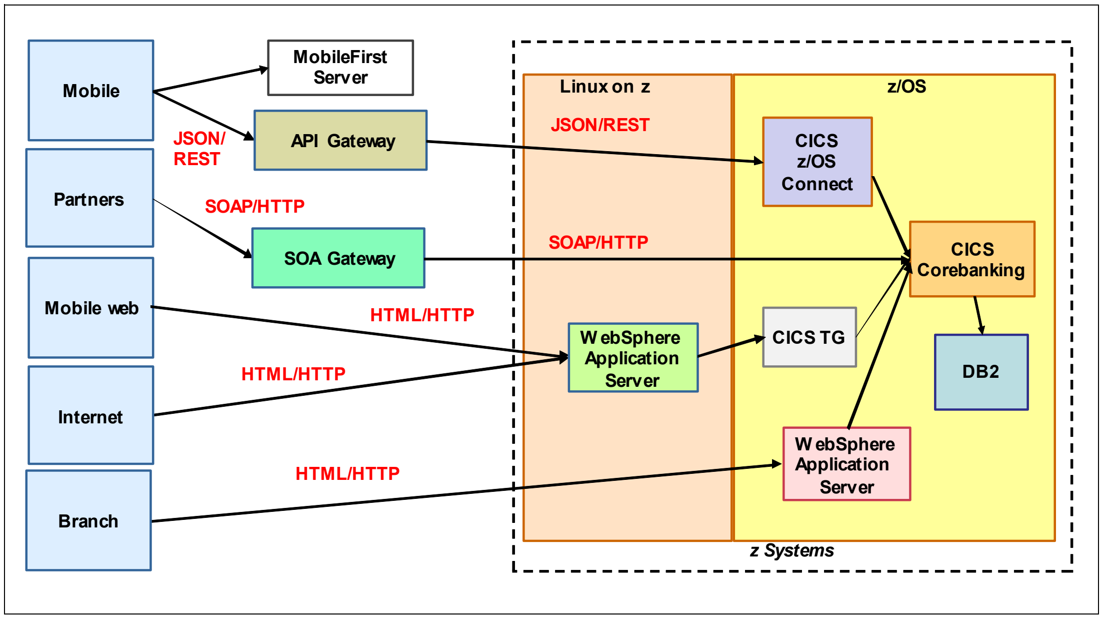


Figure 4-2 Core banking integration architecture

This paper focuses on the two channels that enable requests from mobile devices:

- Mobile web

  Mobile web requests use the existing Internet channel architecture based on WebSphere Application Server and CICS Transaction Gateway.

  For more information about the Mobile web channel see 4.4.1, "Mobile web requests" on page 39.

- Mobile app

  Mobile app requests use REST APIs. z/OS Connect Enterprise Edition (zCEE) provides a REST interface to the CICS core banking application, and an API gateway is used to secure the APIs, manage the API lifecycle, create new API versions, and more.

  For more information about the Mobile channel, see 4.4.2, "Mobile app requests" on page 40.

4.3 CICS configuration

The CICS core banking application is deployed in IBM CICS TS V5.3 CICSPlex®, running within a parallel sysplex in order to take advantage of the z/OS workload management capabilities, including Sysplex Distributor, MVS WLM, and shared access to data, including DB2 and VSAM data.

Figure 4-3 on page 38 shows the CICS configuration.

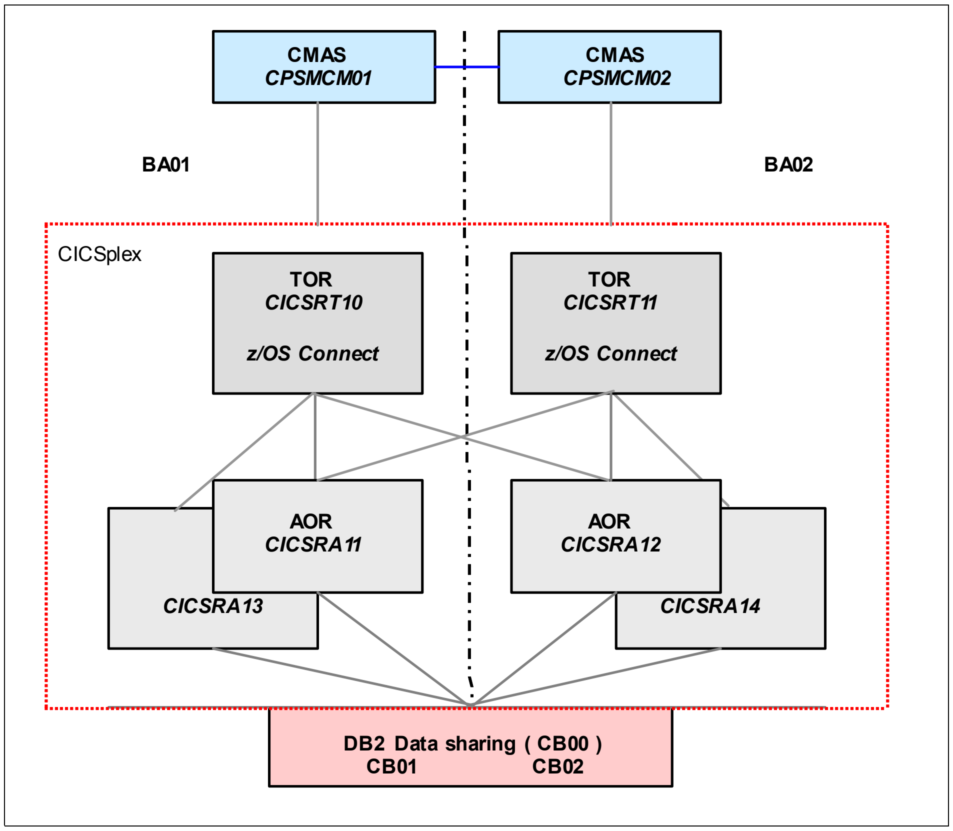


Figure 4-3 CICS configuration

The following list describes the CICS configuration:

- Sysplex Distributor is used for workload management of TCP/IP connections across two LPARs (BA01 and BA02).
- A TOR on each LPAR listens on shared TCP/IP ports:
  - CICSRT10 on BA01
  - CICSRT11 on BA02
- Each CICS TOR receives mobile web requests from CICS Transaction Gateway.
- Each CICS TOR receives mobile app requests (HTTP/JSON) from the API Gateway. These are processed by z/OS Connect running within CICS.
- Program link requests are dynamically routed to cloned AORs using IBM CICSPlex SM:
  - CICSRA11 and CICSRA13 on BA01
  - CICSRA12 and CICSRA14 on BA02
- The CICS core banking programs that run in the AORs share access to the customer and accounts operational database using DB2 data sharing.
- CICSPlex SM provides a single point-of-control and dynamic workload transaction balancing. CICSPlex SM is configured with two CICSPlex SM address spaces (CMAS):
  - CPSMCM01 on BA01
  - CPSMCM02 on BA02

4.4 Mobile request tagging and tracking

The measurement of a mobile workload requires that the mobile work is distinguishable from the non-mobile work in the system. This section describes the method that was used for this paper to *tag and track* mobile requests to the CICS core banking application.

4.4.1 Mobile web requests

Mobile web requests use the same infrastructure as the Internet channel. All web browser requests are processed by a cluster of WebSphere Application servers running on Linux for z Systems platforms. Figure 4-4 shows the architecture of the Internet channel.

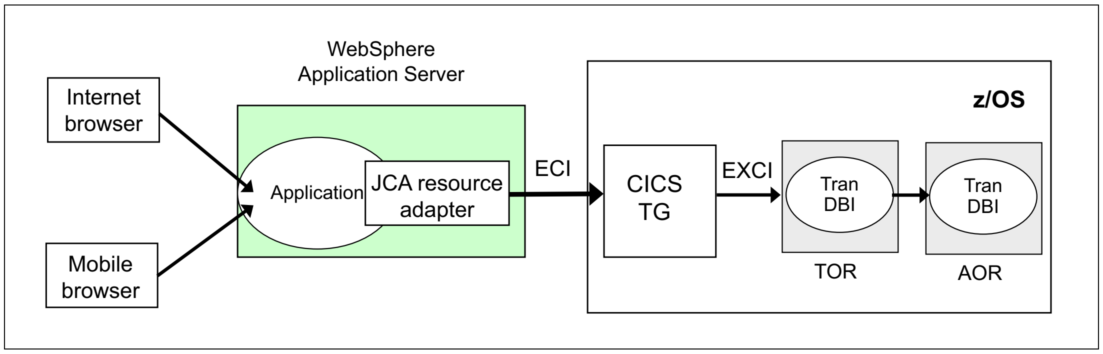


Figure 4-4 Internet channel

Web browser requests arrive in the CICS TORs from a CICS Transaction Gateway (CICS TG) daemon running on z/OS. The same transaction IDs (for example, DBI) are used for Internet browser and mobile browser requests, so the transaction ID cannot be used to differentiate mobile browser requests.

The tag chosen to identify mobile web requests is a user ID. The Java Platform Enterprise Edition (Java EE) application inspects the User-Agent HTTP header in the browser request. If it detects the OS of a mobile device, it sends the External Call Interface (ECI) request to the CICS TG daemon with a specific user ID. This user ID is propagated to the CICS TOR and used to classify the request as a mobile transaction (see 4.5.4, “Classifying CICS Mobile requests” on page 44).

4.4.2 Mobile app requests

Mobile apps make API calls to an API gateway, which then invoke REST APIs exposed in CICS using z/OS Connect. Figure 4-5 shows the architecture of the mobile channel.

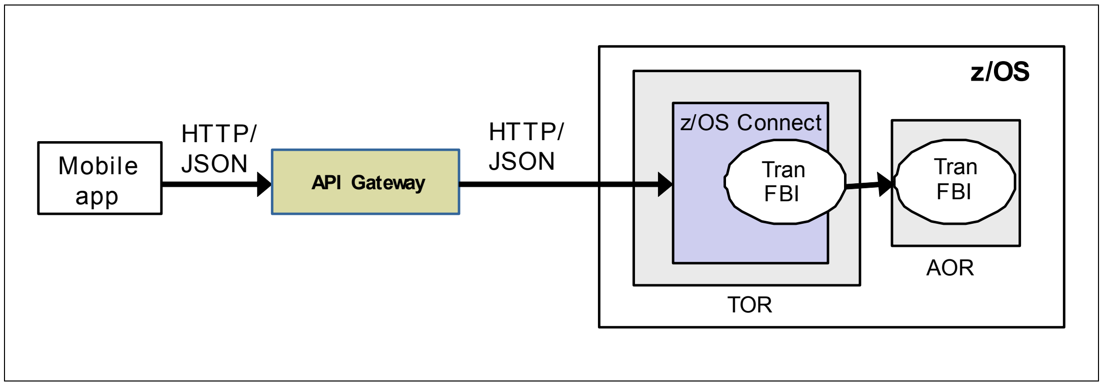


Figure 4-5 Mobile channel

Mobile requests arrive in the CICS TORs and are processed by z/OS Connect. A new set of transaction IDs were created for the mobile channel.

The tag chosen to identify requests from mobile apps is a transaction class. A transaction class gives you a mechanism to limit the number of CICS tasks that can be dispatched in your system. By defining a specific transaction class for mobile app requests, the number of mobile requests that can run at the same time are limited. The transaction class can also be used to classify the request as a mobile transaction (see 4.5, “Classifying mobile requests” on page 40).

For more information about the architecture of the mobile channel, see Appendix A, “Mobile channel architecture” on page 53.

4.5 Classifying mobile requests

This section describes how mobile requests to the CICS core banking application are classified using MVS WLM.

4.5.1 WLM service and report classes

The service classes and report classes that are created to monitor and measure the CICS core banking workload are summarized in the following tables.

Table 4-1 lists a summary of the service classes created for this paper.

Table 4-1 Service classes for CICS core banking workload

Service class	Description
STCMED	Manage CICS STCs based on velocity goal
TRCICSM	Manage CICS transactions based on response time goal

Table 4-2 lists a summary of the report classes created for this paper.

Table 4-2 Report classes for CICS core banking workload

Report class	Description
RSTCTOR	Report on CICS TORs
RSTCAOR	Report on CICS AORs
RCIPART	Report on partner channel
RCIBRAN	Report on branch channel
RCIWEB	Report on Web channel
RCIMOB	Report on mobile channel

Figure 4-6 shows a graphical view of these service and report classes.

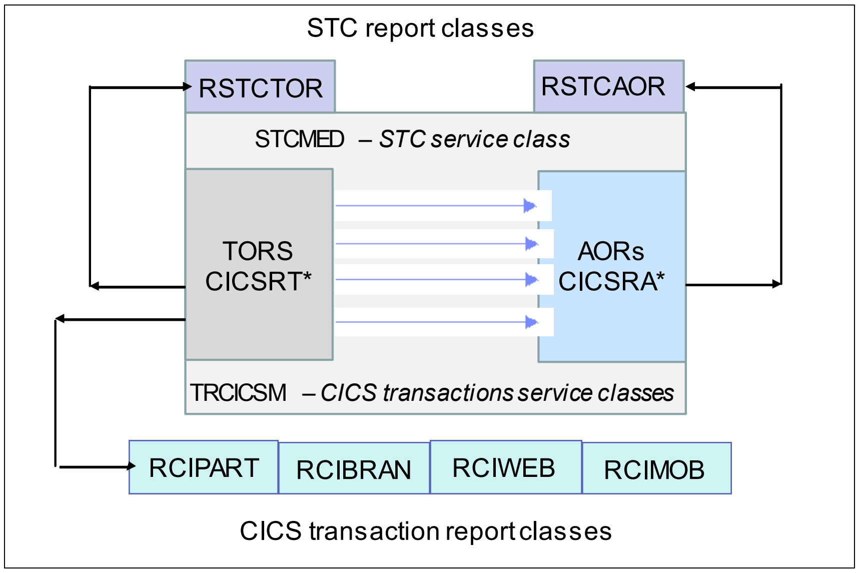

STC report classes
RSTCTOR
RSTCAOR
STCMED – STC service class
TORS
CICSRT*
AORs
CICSRA*
TRCICSM – CICS transactions service classes
RCIPART
RCIBRAN
RCIWEB
RCIMOB
CICS transaction report classes

Figure 4-6 CICS WLM service and report classes

When a CICS region starts, it is classified either as a JES or STC subsystem type (in this case, it is classified in the STC service class STCMED). The CICS regions are managed to the service class until transactions are started. Specified in the CICS region classification rules are that the regions should be *transaction managed* (see Example 4-2 on page 42), so while transactions are running, the regions will to be managed by the service class specified in the associated CICS subsystem classification TRCICSM (see Example 4-3 on page 43) rules rather than the STC classification. More details about the creation of these service classes are provided later.

Two STC report classes for the CICS TORs (RSTCTOR) and AORs (RSTCAOR) are defined. This allows you to measure the CPU consumption of the types of region. Also, the different report classes are associated with the different core banking channels (RCIPART, RCIBRAN, RCIWEB and RCIMOB) in order to be able analyze the performance and CPU consumption of each channel. More details about the creation of these report classes are provided later.

CICS transactions are classified in the TORs and then program links are dynamically routed to the AORs. Transactions are not reclassified in the AORs, so the CICS transactions are reported only in the TORs.

4.5.2 Classifying CICS regions

A single service class STCMED is defined for the CICS STCs (TORs and AORs) and is shown in Example 4-1. Velocity is the goal for this service class.

Example 4-1 Service class for CICS regions

```
      Service Class Name . . . . . : STCMED
Description  . . . . . . . . . STC with Importance 2
Workload Name  . . . . . . . . STC       (name or ?)
Base Resource Group  . . . . . ________  (name or ?)
Cpu Critical . . . . . . . . . NO        (YES or NO)
I/O Priority Group . . . . . . NORMAL    (NORMAL or HIGH)

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

        -- Period --  ------------------- Goal -------------------
Action  £  Duration   Imp.  Description
  __    1  _________   2    Execution velocity of 30
```

This service class is assigned at the address space level in the JES or STC classification rules. See Example 4-2.

Example 4-2 CICS region classification rule

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

Action codes:   A=After     C=Copy       M=Move     I=Insert rule
                B=Before    D=Delete row  R=Repeat   IS=Insert Sub-rule
                                                            More ===>
         -------Qualifier--------                 -------Class--------
Action   Type      Name     Start                  Service     Report       …      Manage Region
                                                                                   Using Goals Of
                                        DEFAULTS: STCSTD      RSTCSTD
 ____  1 …
 ____  1 TN       CICSRT*  ___                    STCMED      RSTCTOR              TRANSACTION
 ____  1 TN       CICSRA*  ___                    STCMED      RSTCAOR              TRANSACTION
 ____  1 …
```

Example 4-2 shows that all address spaces beginning with `CICSR*` have service class STCMED assigned. The TORs are classified using the RSTCTOR report class, and the AORs are classified using the RSTCAOR report class.

Notice that `TRANSACTION` is specified in the `Manage Region Using Goals Of` field. This means that the STC classification rule is used for the CICS address spaces and the CICS subsystem classification rules (see the next sections) for the core banking transactions.

4.5.3 Classifying CICS transactions

Because WLM will manage CICS address spaces using transaction goals, a service class for the CICS transactions must be created next. Example 4-3 shows the TRCICSM service class definition.

Example 4-3 Service class for CICS transactions

```
Service Class Name . . . . . : TRCICSM
Description  . . . . . . . . . Transactions for CICS Importance 2
Workload Name  . . . . . . . . CICS      (name or ?)
Base Resource Group  . . . . . ________  (name or ?)
Cpu Critical . . . . . . . . . NO        (YES or NO)
I/O Priority Group . . . . . . NORMAL    (NORMAL or HIGH)

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

        -- Period --  ------------------- Goal -------------------
Action  £  Duration   Imp.  Description
  __    1  _________   2    Average response time of 00:00:00.100
```

Example 4-3 shows that an average response time of 0.1 seconds is expected for CICS transactions and the goal has an Importance of 2.

CICS subsystem classification rules are used to assign the TRCICSM service class to the CICS core banking transactions and to assign different report classes to each of the core banking channels. Example 4-4 shows the CICS classification rules for the Partner, Branch, and Internet channels.

Example 4-4 Transaction classification rules

```
Subsystem Type . : CICS       Fold qualifier names?   Y  (Y or N)
Description  . . . CICS Environment
Action codes:   A=After     C=Copy        M=Move     I=Insert rule
                B=Before    D=Delete row  R=Repeat   IS=Insert Sub-rule
                                                             More ===>
         -------Qualifier--------                 -------Class--------
Action   Type      Name     Start                  Service     Report
                                        DEFAULTS: TRCISTD     RCICSDEF
 ____  1 …
 ____  1 TN        Y*       ___                    TRCICSM     RCIPART
 ____  1 TN        W*       ___                    TRCICSM     RCIBRAN
 ____  1 TN        D*       ___                    TRCICSM     RCIWEB
 ____  1 …
```

Report class RCIPART is assigned to transactions beginning with the letter “Y” (Partner channel), report class RCIBRAN is assigned to transactions beginning with the letter “W” (Branch channel) and report class RCIWEB is assigned to transactions beginning with the letter “D” (Internet channel).

4.5.4 Classifying CICS Mobile requests

To classify mobile requests, you can now see the updates that were made to the CICS classification rules. For mobile app requests, a new classification rule was added that associates all transactions in transaction class MOBILE with report class RCIMOB. For mobile browser requests, a sub-rule was added that differentiates Internet channel transactions that run with user ID MOBWEB.

Example 4-5 shows the updated CICS classification rules for the new Mobile channel and for mobile web requests.

Example 4-5 Mobile transaction classification rules

```
Subsystem Type . : CICS       Fold qualifier names?   Y  (Y or N)
Description  . . . CICS Environment
Action codes:   A=After     C=Copy        M=Move     I=Insert rule
                B=Before    D=Delete row  R=Repeat   IS=Insert Sub-rule
                                                             More ===>
         -------Qualifier--------                  -------Class--------
Action   Type      Name     Start                   Service     Report
                                          DEFAULTS: TRCISTD     RCICSDEF
 ____  1 TC        MOBILE   ___                     TRCICSM     RCIMOB
 ____  1 TN        Y*       ___                     TRCICSM     RCIPART
 ____  1 TN        W*       ___                     TRCICSM     RCIBRAN
 ____  1 TN        D*       ___                     TRCICSM     RCIWEB
 ____  2   UI        MOBWEB   ___                   TRCICSM     RCIWEB
```

Now that you have defined rules for classifying the types of mobile request, you can assign the MOBILE Reporting Attribute to these transactions, as shown in Example 4-6.

Example 4-6 Assigning the MOBILE reporting attribute

```
-------Qualifier--------      Storage   Reporting  Manage Region
Action   Type      Name     Start      Critical  Attribute  Using Goals Of

 ____  1 TC        MOBILE   ___        NO        MOBILE     N/A
 ____  1 TN        Y*       ___        NO        NONE       N/A
 ____  1 TN        W*       ___        NO        NONE       N/A
 ____  1 TN        D*       ___        NO        NONE       N/A
 ____  2   UI        MOBWEB   ___      NO        MOBILE     N/A
```

These WLM classification rules will allow you to monitor and measure the core banking workload. You will be able to monitor transaction rates and response times for the different channels; for example, you can monitor the mobile channel using the report class RCIMOB. You are also able to measure the CPU consumption for the different channels. See 4.7.1, “Measuring the core banking workload” on page 47 for details of how to capture and analyze mobile CPU consumption.

Because the MOBILE Reporting Attribute for the mobile app and mobile web requests was specified, WLM will also aggregate the system-wide mobile CPU consumption in order to calculate a rolling 4-hour average that can be used for MWP. See 4.7.2, “Measuring the impact of mobile workload on the rolling 4-hour average” on page 51 to view the impact of MWP on the peak rolling 4-hour average.

4.6 Real-time monitoring

Real-time monitoring of the core banking workload is important in order to verify that the workload is running correctly or to identify bottlenecks in the event that problems exist.

A collection of tools can be used to have a real-time view of the core banking workload:

- RMF Monitor III for a real-time view of the performance of the different core banking channels (see “RMF Monitor III”)
- RMF Performance Monitor (RMF PM) for a GUI real-time view of the performance of the MVS LPARs that run the core banking workload and a graphical view of the performance of the core banking channels (see “RMF PM”)
- IBM CICS Explorer® for a real-time view of executing CICS transactions (see “CICS Explorer”)

RMF Monitor III

Example 4-7 shows a snapshot of an RMF Monitor III Sysplex Summary report. This report is used as the entry point for real-time monitoring. The summary report shows all active workloads with their performance values, including the goals for each service class period.

Example 4-7 RMF Monitor III real-time monitoring of core banking channels

```
               ------- Goals versus Actuals --------  Trans --Avg. Resp. Time-
               Exec Vel  --- Response Time ---  Perf  Ended  WAIT EXECUT ACTUAL
Name     T  I  Goal Act  ---Goal--- --Actual--  Indx  Rate   Time   Time   Time

TRCICSM  S  2        63  0.100 AVG  0.014  AVG  0.14  143.5 0.000  0.014  0.014
RCIBRAN  R          N/A                               14.60 0.000  0.029  0.029
RCIGENK  R          N/A                               6.133 0.000  0.000  0.000
RCIMOB   R          N/A                               39.30 0.000  0.015  0.015
RCIPART  R          N/A                               11.60 0.000  0.021  0.021
RCITRNC  R          N/A                               0.400 0.000  0.533  0.533
RCIWEB   R          N/A                               68.80 0.000  0.010  0.010
```

Example 4-7 shows only selected service and report classes. The report of service class TRCICSM (service class for CICS transactions) shows that the core banking transactions are meeting their performance goal (average response time of 100 ms). The report also shows the distribution of workload across the different core banking channels (report classes RCIBRAN, RCIMOB, RCIPART and RCIWEB).

RMF PM

RMF PM is a Java-based monitor that provides a graphical monitoring capability. Figure 4-7 shows an example frame from the performance monitoring view of RMF PM. It shows a real-time graphical view of the transaction rates and response times for the various core banking channels.

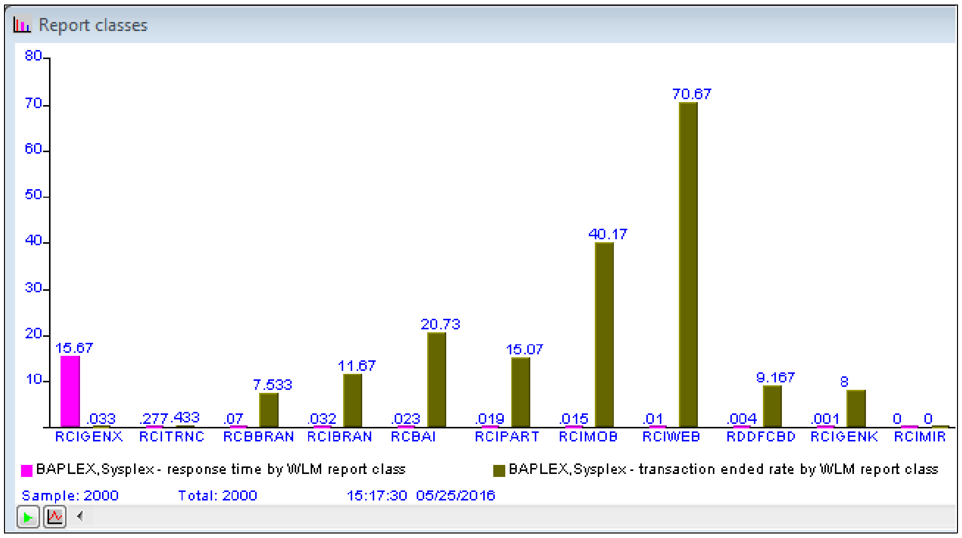
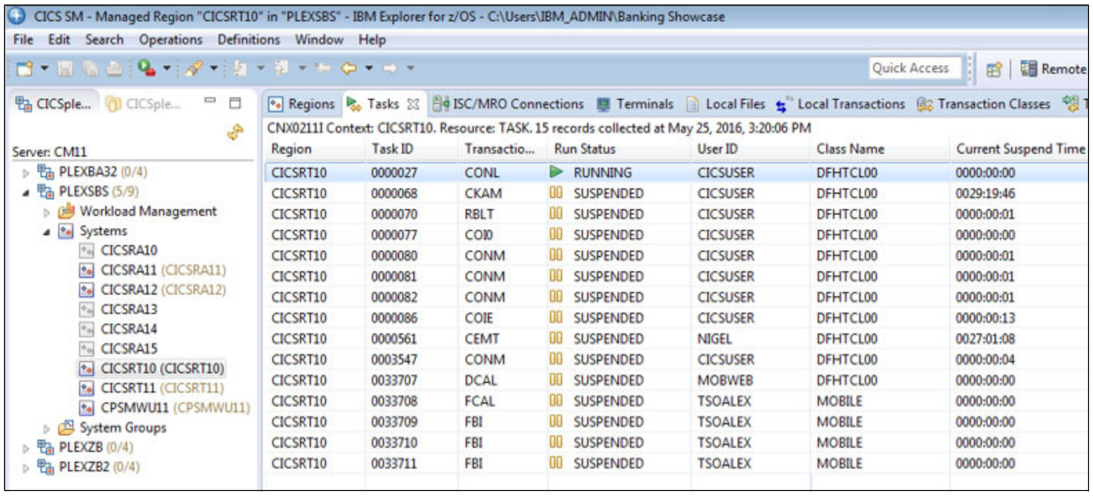


Figure 4-7 RMF PM performance monitoring view

CICS Explorer

CICS Explorer is a system management tool that provides an easy-to-use way to manage one or more CICS regions. It connects to either a CICSPlex SM web user interface (WUI) server or a single CICS region, providing a base set of CICSPlex SM functions to manage CICS regions, perform tasks, and monitor CICS activity.

Figure 4-8 shows an example CICS Explorer view.



CNX0211I Context: CICSRT10. Resource: TASK. 15 records collected at May 25, 2016, 3:20:06 PM

Region	Task ID	Transactio...	Run Status	User ID	Class Name	Current Suspend Time
CICSRT10	0000027	CONL	RUNNING	CICSUSER	DFHTCL00	0000:00:00
CICSRT10	0000068	CKAM	SUSPENDED	CICSUSER	DFHTCL00	0029:19:46
CICSRT10	0000070	RBLT	SUSPENDED	CICSUSER	DFHTCL00	0000:00:01
CICSRT10	0000077	COI0	SUSPENDED	CICSUSER	DFHTCL00	0000:00:00
CICSRT10	0000080	CONM	SUSPENDED	CICSUSER	DFHTCL00	0000:00:01
CICSRT10	0000081	CONM	SUSPENDED	CICSUSER	DFHTCL00	0000:00:01
CICSRT10	0000082	CONM	SUSPENDED	CICSUSER	DFHTCL00	0000:00:01
CICSRT10	0000086	COIE	SUSPENDED	CICSUSER	DFHTCL00	0000:00:13
CICSRT10	0000561	CEMT	SUSPENDED	NIGEL	DFHTCL00	0027:01:08
CICSRT10	0003547	CONM	SUSPENDED	CICSUSER	DFHTCL00	0000:00:04
CICSRT10	0033707	DCAL	SUSPENDED	MOBWEB	DFHTCL00	0000:00:00
CICSRT10	0033708	FCAL	SUSPENDED	TSOALEX	MOBILE	0000:00:00
CICSRT10	0033709	FBI	SUSPENDED	TSOALEX	MOBILE	0000:00:00
CICSRT10	0033710	FBI	SUSPENDED	TSOALEX	MOBILE	0000:00:00
CICSRT10	0033711	FBI	SUSPENDED	TSOALEX	MOBILE	0000:00:00

Figure 4-8 CICS Explorer view

Figure 4-8 shows a snapshot of the CICS tasks that are running in the CICS TOR region CICSRT10. It shows several core banking transactions, for example, transactions DCAL, FCAL, and several instances of transaction FBI. This CICS Explorer view allows you to validate the mobile tagging and tracking strategy. It shows that transaction DCAL is running with user ID MOBWEB (DCAL is a transaction initiated from a mobile browser). It also shows three instances of transactions FBI running within transaction class MOBILE (FBI is a transaction initiated from a mobile app).

4.7 Measuring the mobile workloads

Workload Activity reports from the RMF Post Processor are used here to evaluate the attainment of performance goals and to measure the CPU consumption of the CICS regions and different core banking channels.

This section focuses on the measurements for the WLM report classes that are defined for the core banking workload. For an analysis of the service class measurements, and further interpretation of the APPL% and Transaction APPL% measurement fields, see Appendix B, “Analysis of CICS CPU consumption” on page 57.

4.7.1 Measuring the core banking workload

The example in this section shows Workload Activity reports for a one-minute interval for the CICS AORs, CICS TORs and different core banking channels. It then shows a couple of summary charts that highlight how CPU is attributed between the CICS regions and the core banking channel report classes.

Note: The reports use aggregated data form the two members of Sysplex BAPLEX.

CICS TORs

Example 4-8 shows an extract of an RMF Workload Activity report for report class RSTCTOR (note that certain fields are removed from this example in order to highlight the fields that show the CPU consumption of the TORs).

Example 4-8 RMF Workload Activity report for report class RSTCTOR

```
REPORT BY: POLICY=POLZOS                          REPORT CLASS=RSTCTOR
                                                  DESCRIPTION =CICS TOR

-TRANSACTIONS-  TRANS-TIME HHH.MM.SS.TTT      SERVICE TIME  ---APPL %---
AVG       2.00  ACTUAL                 0      CPU    2.811  CP      3.19
MPL       2.00  EXECUTION              0      SRB    0.031  AAPCP   0.00
ENDED        0  QUEUED                 0      RCT    0.000  IIPCP   0.01
END/S     0.00  R/S AFFIN              0      IIT    0.000
£SWAPS       0  INELIGIBLE             0      HST    0.000  AAP
EXCTD        0  CONVERSION             0      AAP      N/A  IIP     1.10
AVG ENC   0.00  STD DEV            0     IIP   0.928
REM ENC   0.00
MS ENC    0.00

TRANSACTION APPL% :   TOTAL :  CP   0.53   AAP/IIP ON CP   0.00   AAP/IIP   0.37
                      MOBILE : CP   0.00   AAP/IIP ON CP   0.00   AAP/IIP   0.00
```

The report shows that the CICS TORs consumed 3.19% of a CP. The report also shows 0.53% of a CP reported as Transaction APPL%. This is CPU that has not been captured for specific transactions, for example, task dispatch processing. You can consider this as the overhead of the CICS TOR regions (see Appendix B, “Analysis of CICS CPU consumption” on page 57 for more information).

CICS AORs

Example 4-9 shows an extract of an RMF Workload Activity report for report class RSTCAOR (note that certain fields are removed from this example in order to highlight the fields that show the CPU consumption of the AORs).

Example 4-9 RMF Workload Activity report for report class RSTCAOR

```
REPORT BY: POLICY=POLZOS                            REPORT CLASS=RSTCAOR
                                                    DESCRIPTION =CICS AOR

-TRANSACTIONS-  TRANS-TIME HHH.MM.SS.TTT      SERVICE TIME  ---APPL %---
AVG       2.00  ACTUAL                 0      CPU    9.798  CP     16.41
MPL       2.00  EXECUTION              0      SRB    0.057  AAPCP   0.00
ENDED        0  QUEUED                 0      RCT    0.000  IIPCP   0.00
END/S     0.00  R/S AFFIN              0      IIT    0.000
£SWAPS       0  INELIGIBLE             0      HST    0.000  AAP
EXCTD        0  CONVERSION             0      AAP      N/A  IIP     0.01
AVG ENC   0.00 STD DEV               0      IIP    0.008
REM ENC   0.00
MS ENC    0.00

TRANSACTION APPL% :   TOTAL :   CP   1.21   AAP/IIP ON CP   0.00   AAP/IIP   0.01
                      MOBILE :  CP   0.00   AAP/IIP ON CP   0.00   AAP/IIP   0.00
```

The report shows that the CICS TORs consumed 16.41% of a CP. The report also shows 1.21% of a CP reported as Transaction APPL%. This is CPU that has not been captured for specific transactions, for example, task dispatch processing. You can consider this as the overhead of the CICS AOR regions.

Mobile channel

Example 4-10 shows an RMF Workload Activity report for report class RCIMOB.

Example 4-10 RMF Workload Activity report for report class RCIMOB

```
REPORT BY: POLICY=POLZOS                     REPORT CLASS=RCIMOB
                                         DESCRIPTION =New CICS Mobile Channel

-TRANSACTIONS-  TRANS-TIME HHH.MM.SS.TTT
AVG       0.00  ACTUAL                14
MPL       0.00  EXECUTION              0
ENDED     2384  QUEUED                 0
END/S    39.73  R/S AFFIN              0
£SWAPS       0  INELIGIBLE             0
EXCTD        0  CONVERSION             0
AVG ENC   0.00  STD DEV               11
REM ENC   0.00
MS ENC    0.00

TRANSACTION APPL% :   TOTAL :   CP   7.45   AAP/IIP ON CP   0.01   AAP/IIP   0.73
                      MOBILE :  CP   7.45   AAP/IIP ON CP   0.01   AAP/IIP   0.73
```

The report shows that an average of 39.73 mobile transactions were executed per second, with an average response time of 14 ms. It also shows that 7.45% of a CP was consumed by these transactions, and that 100% of this is reported as mobile CPU.

Important: The TRANSACTION APPL% CPU time (7.45%) includes transaction CPU from the TOR and AOR. This is because the transaction CPU from the AOR is accumulated back to the TOR where the transaction is classified.

Partner channel

Example 4-11 shows an RMF Workload Activity report for report class RCIPART.

Example 4-11 RMF Workload Activity report for report class RCIPART

```
REPORT BY: POLICY=POLZOS                         REPORT CLASS=RCIPART
                                                 DESCRIPTION =CICS Partner Channel

-TRANSACTIONS-  TRANS-TIME HHH.MM.SS.TTT
AVG       0.00  ACTUAL                20
MPL       0.00  EXECUTION              0
ENDED      800  QUEUED                 0
END/S    13.33  R/S AFFIN              0
£SWAPS       0  INELIGIBLE             0
EXCTD        0  CONVERSION             0
AVG ENC   0.00  STD DEV               12
REM ENC   0.00
MS ENC    0.00

TRANSACTION APPL% :   TOTAL :   CP   1.59   AAP/IIP ON CP   0.00   AAP/IIP   0.00
                      MOBILE :  CP   0.00   AAP/IIP ON CP   0.00   AAP/IIP   0.00
```

The report shows that an average of 13.33 partner transactions were executed per second, with an average response time of 20ms. It also shows that 1.59% of a CP was consumed by these transactions, and that nothing is reported as mobile CPU for this report class.

Branch channel

Example 4-12 shows an RMF Workload Activity report for report class RCIBRAN.

Example 4-12 RMF Workload Activity report for report class RCIBRAN

```
REPORT BY: POLICY=POLZOS                         REPORT CLASS=RCIBRAN
                                                 DESCRIPTION =CICS AOR/TOR BRANCH

-TRANSACTIONS-  TRANS-TIME HHH.MM.SS.TTT
AVG       0.00  ACTUAL                30
MPL       0.00  EXECUTION              0
ENDED      791  QUEUED                 0
END/S    13.18  R/S AFFIN              0
£SWAPS       0  INELIGIBLE             0
EXCTD        0  CONVERSION             0
AVG ENC   0.00  STD DEV               38
REM ENC   0.00
MS ENC    0.00

TRANSACTION APPL% :   TOTAL :   CP   2.27   AAP/IIP ON CP   0.00   AAP/IIP   0.00
                      MOBILE :  CP   0.00   AAP/IIP ON CP   0.00   AAP/IIP   0.00
```

The report shows that an average of 13.18 branch transactions were executed per second, with an average response time of 30ms. It also shows that 2.27% of a CP was consumed by these transactions, and that nothing is reported as mobile CPU for this report class.

Internet channel

Example 4-13 shows an RMF Workload Activity report for report class RCIWEB.

Example 4-13 RMF Workload Activity report for report class RCIWEB

```
REPORT BY: POLICY=POLZOS                              REPORT CLASS=RCIWEB
                                                      DESCRIPTION =CICS WEB Channel


-TRANSACTIONS-  TRANS-TIME HHH.MM.SS.TTT
AVG       0.00  ACTUAL                 9
MPL       0.00  EXECUTION              0
ENDED     4189  QUEUED                 0
END/S    69.82  R/S AFFIN              0
£SWAPS       0  INELIGIBLE             0
EXCTD        0  CONVERSION             0
AVG ENC   0.00  STD DEV                8
REM ENC   0.00
MS ENC    0.00

TRANSACTION APPL% :    TOTAL :   CP    6.50   AAP/IIP ON CP    0.00   AAP/IIP   0.00
                       MOBILE :  CP    3.07   AAP/IIP ON CP    0.00   AAP/IIP   0.00
```

The report shows that an average of 69.82 Internet transactions were executed per second, with an average response time of 9ms. It also shows that 6.50% of a CP was consumed by these transactions, of which 3.07% of a CP is reported as mobile CPU for this report class.

Summary

Figure 4-9 shows the address space CPU (APPL%) for the CICS TOR and AOR STC report classes across 5 reporting intervals. It also shows the total CPU used by all address spaces in the sysplex.

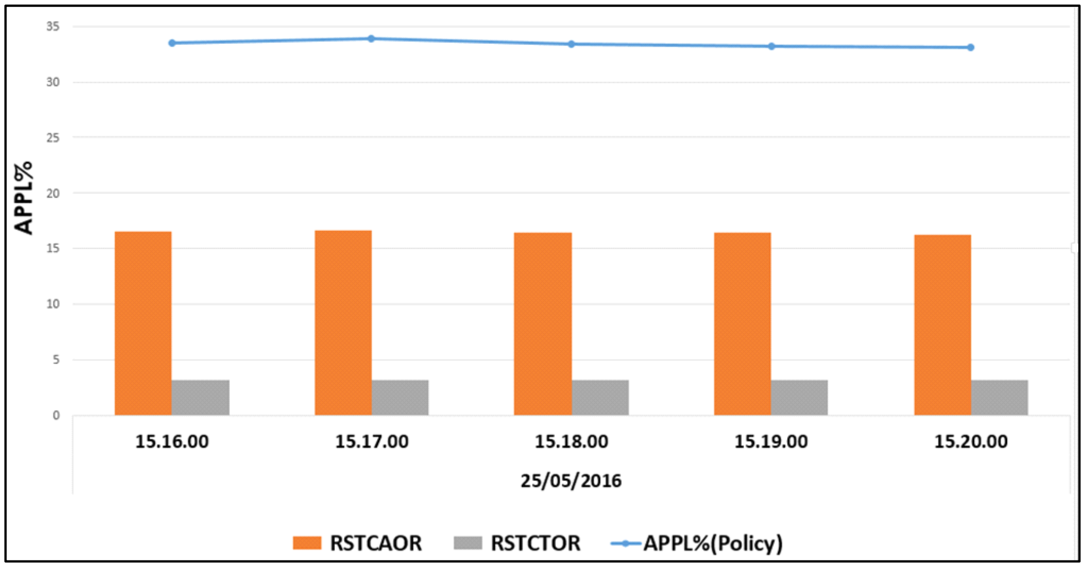


Figure 4-9 Address space CPU (APPL%) for CICS TORs and AORs

Figure 4-9 shows that most of the CICS CPU is consumed in the AORs. This is because the core banking transactions run in the AORs and access the DB2 database in the AORs. It also shows that the CICS region CPU accounts for over half of the total CPU consumed by all address spaces in the sysplex.

Important: Prior to the recent enhancements to CICS and RMF, this was the only way to break down CICS CPU using RMF data. Measuring the CPU consumed by a certain class of transactions without using CICS monitoring data (SMF 110 records) was not possible.

Figure 4-10 shows a CPU consumption comparison across the core banking channels.

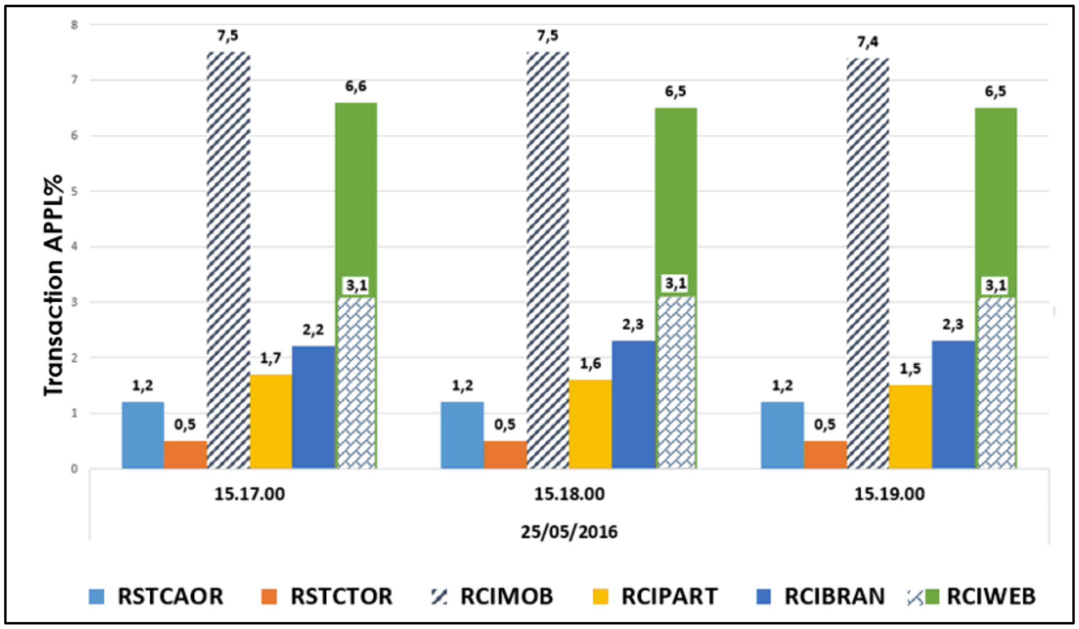


Figure 4-10 Transaction CPU (Transaction APPL%) for core banking channels

Figure 4-10 shows the transaction CPU (transaction APPL%) for the core banking channel report classes across three recording intervals. The checkered bars represent mobile CPU that is eligible for MWP. The figure also shows the transaction APPL% values for the CICS TOR (report class RSTCTOR) and AORs (report class RSTCAOR).

4.7.2 Measuring the impact of mobile workload on the rolling 4-hour average

WLM tracks processor consumption separately for each value of the MOBILE reporting attribute, and reports consumption at the service and report class level. WLM also aggregates the system-wide mobile consumption separately, rolls the values into a rolling 4-hour average, and makes this value available in MSUs (Millions of Service Units) to monitors like RMF.

As an illustration, Figure 4-11 shows the R4HA and adjusted R4HA for a 6-hour period for the two LPARs running the core banking workload, BA01 and BA02. The values shown in Figure 4-11 are calculated automatically by WLM.

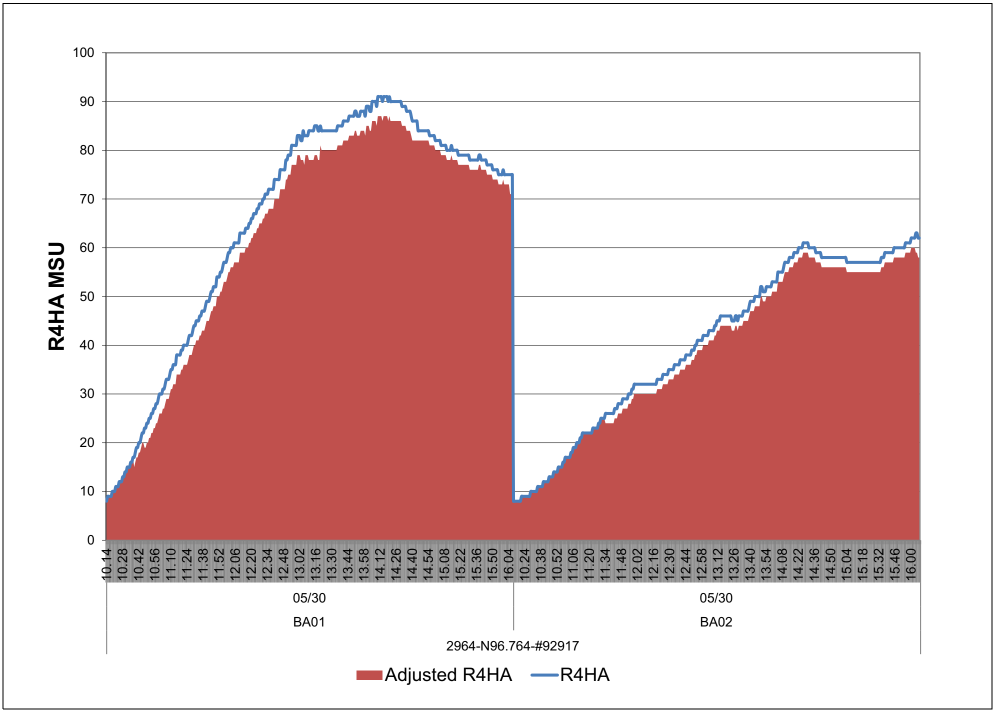


Figure 4-11 Impact of mobile workload on the rolling 4-hour average

The blue line represents the R4HA, and the red area is the adjusted R4HA, obtained by subtracting 60% of the mobile MSU R4HA. Consider the following information for this interval:

- For BA01, the R4HA peak is 91 MSUs at 14h12 and the adjusted R4HA peak is 87 MSUs (also at 14h12).
- For BA02, the R4HA peak is 63 MSUs at 16h00 and the adjusted R4HA peak is 60 MSUs (also at 16h00).

Note: After calculating the adjusted R4HA peak for an interval of one month, the monthly R4HA peak might possibly move to a different interval. This is most likely to occur when there is a very variable mobile workload across the month.

4.8 Summary

The recent enhancements to WLM outlined in this paper help to more easily measure mobile workloads. Measuring different types of workloads is often important for the purpose of capacity planning, managing workload spikes, charge back and pricing.

In addition to the WLM enhancements, CICS and IMS, the leading transaction processing systems from IBM, have been enhanced to help more easily classify mobile transactions with WLM, and report and measure mobile transaction CPU usage.

A

Mobile channel architecture

This appendix provides more information about the new mobile channel and how mobile app requests are processed by the CICS core banking application.

Mobile channel

Digital channels have become the primary way to initiate consumer and business engagements, providing both accessibility to a global audience and a cost-effective means to conduct business. In recent years, we have seen mobile devices become the preferred way in which consumers, business partners, and employees interact with companies and each other.

Take the example of a banking mobile app. It commonly offers access to account information, the ability to make payments, and perhaps the user can deposit a check using the device's camera. All of these activities require information directly from the bank.

To create powerful and useful mobile apps, the use of application programming interfaces (APIs) becomes essential. This is why Bank A is developing mobile apps that make use of a set of APIs created from the bank's mainframe services, including the core banking application.

One challenge is in creating APIs based on z/OS assets that were implemented long before Representational State Transfer (REST) and JavaScript Object Notation (JSON) came into being. This challenge is resolved using z/OS Connect Enterprise Edition (see "z/OS Connect Enterprise Edition" on page 55).

A second challenge is to control access to the APIs, socialize the APIs and monitor their usage. This challenge is resolved using IBM API Connect™ (see "IBM API Connect" on page 55).

Core banking APIs

Table A-1 shows some of the APIs exposed by the core banking system.

Table A-1 Core banking APIs

Operation	HTTP verb	URI	JSON message
Balance Inquiry	GET	/accounts/{accountID}	
Internal transfer	POST	/accounts/{accountID}/internalTransfer	{"toAccount": 8002, "amount": 1000, "currency": "USD"}
Posting Inquiry	GET	/accounts/{accountID}/transactions?count=n	
Customer Account List	GET	/customers/{customerID}	
Bill Payment	POST	/accounts/{accountID}/billPayment	{"idBill": 123456, "amount": 100, "currency": "USD"}

Table A-1 describes the following APIs:

- Balance Inquiry is enabled using an HTTP GET verb and the URI `/accounts/{accountID}` without a JSON request message because of the account identifier is a path parameter.
- An internal transfer between two accounts of the bank is enabled using an HTTP POST verb and the URI `/accounts/{accountID}/internalTransfer`. The URI identifies the account to be debited and the account to be credited is part of the JSON request message, along with the amount and currency.

- A posting inquiry returns the last set of transactions for an account and is enabled using an HTTP GET verb and the URI `/accounts/{accountID}/transactions?count=`*n* where *n* represents the number of transactions.
- A customer account list returns a list of accounts held by a customer and is enabled using an HTTP GET verb and the URI `/customers/{customerID}`.
- A bill payment is enabled using an HTTP POST verb and the URI `/accounts/{accountID}/billPayment`. The URI identifies the account to be debited and the amount of the bill is part of the JSON request message, along with the bill payment id and currency.

z/OS Connect Enterprise Edition

z/OS Connect Enterprise Edition (z/OS Connect EE) enables API developers to construct REST APIs from existing z/OS assets such as CICS and IMS applications. The APIs are constructed and packaged with the Eclipse-based API Editor that is provided with z/OS Connect EE, and then deployed to the z/OS Connect runtime.

JSON schema definitions have been generated from the COBOL copybooks of the core banking programs using the z/OS Connect provided tool BAQLS2JS (language structure to JSON). The API editor is then used to visually map the RESTful requests (see Table A-1 on page 54) to the JSON schema. This means that only the copybook fields that need to be exposed to the API client are defined in the REST/JSON interface. For example, in the case of internal transfer, only account IDs, amount, and currency are exposed.

The generated API package includes the SWAGGER definition file which is imported into API Connect (see “IBM API Connect”).

In the case of Bank A, z/OS Connect was configured to run in a Liberty server in CICS. This configuration enables local, high-performance connectivity to CICS programs. It runs on zIIP engines because of the use of Java and Liberty server. The Java Virtual Machine (JVM) is enabled in CICS which allows z/OS Connect to benefit from the workload management, scalability and performance of CICS.

IBM API Connect

Bank A’s mobile apps make API calls to an API gateway, which is a component of IBM API Connect. IBM API Connect provides full API management to create, run, manage, and secure APIs and microservices:

- Create: Create high-quality, scalable and secure APIs for application servers, databases, enterprise service buses (ESBs), and mainframe services.
- Run: Take advantage of integrated tooling to build, debug and deploy APIs and microservices using Node.js or Java.
- Manage: Create and manage portals that allow developers to quickly discover and consume APIs and securely access enterprise data, and monitor APIs to improve performance.
- Secure: Secure and govern APIs and microservices, and enforce policies to secure back-end information and compliance with regulatory mandates.

Bank A deployed the APIs that it created by using API Connect to an IBM DataPower® Gateway. DataPower provides threat protection, security enforcement, service level agreement enforcement, rate limiting, intelligent load balancing, and response caching for APIs and microservices.

In addition to controlling access and securing the APIs, API Connect also has a role in mapping the JSON payloads, for example, the fields exposed in the internal transfer API are renamed to those generated from the COBOL copybook. This avoids having to expose mainframe naming models to API clients.

Classifying the mobile app requests

The tag chosen to identify mobile web requests is a transaction class. A transaction class also gives you a mechanism to limit the number of CICS tasks that can be dispatched in your system. By defining a specific transaction class for mobile app requests, you can limit the number of mobile requests that can run at the same time.

B

Analysis of CICS CPU consumption

This appendix explains the difference between CPU attributed to the CICS address space (APPL%) and CPU attributed to transactions (transaction APPL%). This is done by analyzing the RMF Workload Activity reports for service classes that are used to manage the CICS regions and CICS transactions.

CICS regions

Example B-1 shows an extract of an RMF Workload Activity report for service class STCMED. This is the service class that we defined in Example 4-1 on page 42 or the CICS STCs (TORs and AORs). Certain fields are removed in this example in order to highlight the fields that show the CPU consumption of the CICS regions.

Example B-1 RMF Workload Activity report for service class STCMED

```
REPORT BY: POLICY=POLZOS     WORKLOAD=STC        SERVICE CLASS=STCMED
                                                 CRITICAL     =NONE
                                                 DESCRIPTION  =STC Medium Priority

-TRANSACTIONS-  TRANS-TIME HHH.MM.SS.TTT      SERVICE TIME  ---APPL %---
AVG      42.00  ACTUAL                 0      CPU   18.868  CP     22.46
MPL      42.00  EXECUTION              0      SRB    0.180  AAPCP   0.00
ENDED        0  QUEUED                 0      RCT    0.000  IIPCP   1.00
END/S     0.00  R/S AFFIN              0      IIT    0.000
£SWAPS       0  INELIGIBLE             0      HST    0.000  AAP      N/A
EXCTD        0  CONVERSION             0      AAP      N/A  IIP     6.66
AVG ENC   0.00  STD DEV              0      IIP    5.570
REM ENC   0.00
MS ENC    0.00

TRANSACTION APPL% :   TOTAL :   CP   4.61   AAP/IIP ON CP   0.99   AAP/IIP   5.93
                      MOBILE :  CP   0.00   AAP/IIP ON CP   0.00   AAP/IIP   0.00
```

The report shows that the CICS regions consumed 22.46% of a CP. The report also shows 4.61% of a CP reported as Transaction APPL%. This is CPU that has not been captured for specific transactions, for example, task dispatch processing. Consider this as the overhead of the CICS regions.

Note: Transaction APPL% reported in a STC service class for a CICS region is CPU time uncaptured by CICS.

The total and mobile processor consumption WLM aggregates for the CICS region service classes excludes transaction service, which is separately accumulated in the CICS transaction service classes and must not be counted twice. So the values aggregated and reported for the CICS region service classes is region-only service, that is, the region overhead.

CICS Transactions

Example B-2 shows an extract of an RMF Workload Activity report for service class TRCICSM. This is the service class that is defined in Example 4-3 on page 43 for the CICS transactions. Certain fields are removed from this example in order to highlight the fields that show the CPU consumption of the CICS transactions.

Example B-2 RMF Workload Activity report for service class TRCICSM

```
REPORT BY: POLICY=POLZOS     WORKLOAD=CICS       SERVICE CLASS=TRCICSM
                                                 CRITICAL     =NONE
                                                 DESCRIPTION  =CICS Transactions

-TRANSACTIONS-  TRANS-TIME HHH.MM.SS.TTT
AVG       0.00  ACTUAL                14
MPL       0.00  EXECUTION              0
ENDED     8609  QUEUED                 0
END/S   143.48  R/S AFFIN              0
£SWAPS       0  INELIGIBLE             0
EXCTD        0  CONVERSION             0
AVG ENC   0.00  STD DEV               31
REM ENC   0.00
MS ENC    0.00

TRANSACTION APPL% :   TOTAL :   CP  17.87   AAP/IIP ON CP   0.01   AAP/IIP   0.73
                      MOBILE :  CP  10.53   AAP/IIP ON CP   0.01   AAP/IIP   0.73
```

The report shows that an average of 143.48 transactions were executed per second, with an average response time of 14 ms. It also shows that 17.87% of a CP was consumed by these transactions, of which 10.53% of a CP is reported as mobile CPU.

Summary

Figure B-1 shows percentages of the CICS transaction CPU.

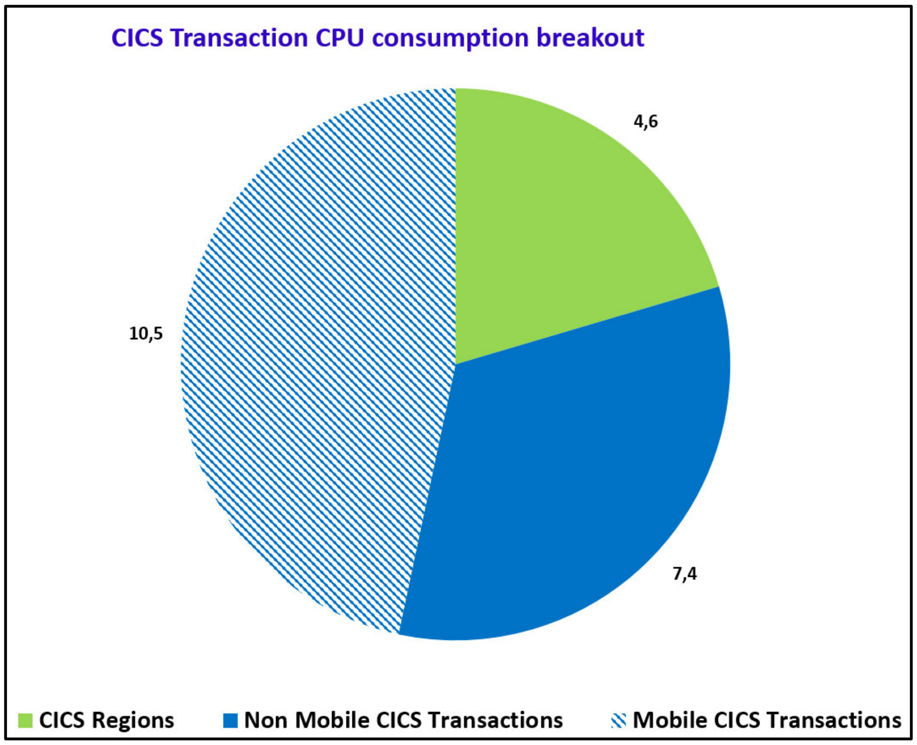


Figure B-1 CICS Transaction CPU

Figure B-1 shows the following information:

- 10.5% of a CP is attributed to mobile CICS transactions.
- 7.4% of a CP is attributed to non-mobile transactions.
- 4.6% of a CP is attributed to CICS overhead.

The sum of these values, 22.5% of a CP, represents the same CPU consumption as reported in APPL% for the CICS regions (22.46) in Example B-1 on page 58.

Note: CICS Transaction APPL% + CICS STC Transaction APPL% = CICS STC APPL% (17.9 + 4.6 = 22.5).

This highlights that Transaction APPL% provides a detailed breakdown of CICS transaction CPU between mobile-initiated transactions, non-mobile transactions and CICS overhead. When using different report classes (see 4.7.1, “Measuring the core banking workload” on page 47) it can also provide a breakdown of CPU between different sets of CICS transaction.

C

IBM Banking Showcase

The mobile workload monitoring and measurement scenarios and examples documented in this publication are based on the IBM Banking Showcase. The showcase is a simulation of a real bank, developed by a team of banking and IT infrastructure specialists working in the IBM Montpellier Client Center in France.

The banking systems that are used in the showcase run a mixed workload of real-world financial transactions, including cash withdrawals, deposits, mortgages, and payment transactions.

The main objectives of the showcase are these:

- To help banks architect their future IT systems by demonstrating innovative banking solutions that support their digital transformation challenges by leveraging a rich heritage on the z Systems platform.
- To demonstrate that infrastructure matters. By integrating the latest IBM hardware and software products and also IBM Business Partner solutions, the showcase highlights z Systems infrastructure value and uniqueness.

The showcase uses a realistic mix of workload and runs at operational volumes that are representative of a typical European bank:

- Six million clients and twelve million accounts
- A transaction rate rising to a peak of 500 transactions per second

The COBOL core banking application runs on CICS and stores customer and account records in DB2 for z/OS. As the showcase evolved, an approach based on service-oriented architecture (SOA) was implemented for reusing the CICS core banking system across multiple channels (web, ATM, mobile, and others).

Appendix B, “Analysis of CICS CPU consumption” on page 57 describes how a new mobile channel is enabled in the showcase by using APIs based on the core banking systems. Mobile apps also use APIs based on a master data management (MDM) system and a business rules engine. Figure C-1 shows several screen captures of the mobile app.

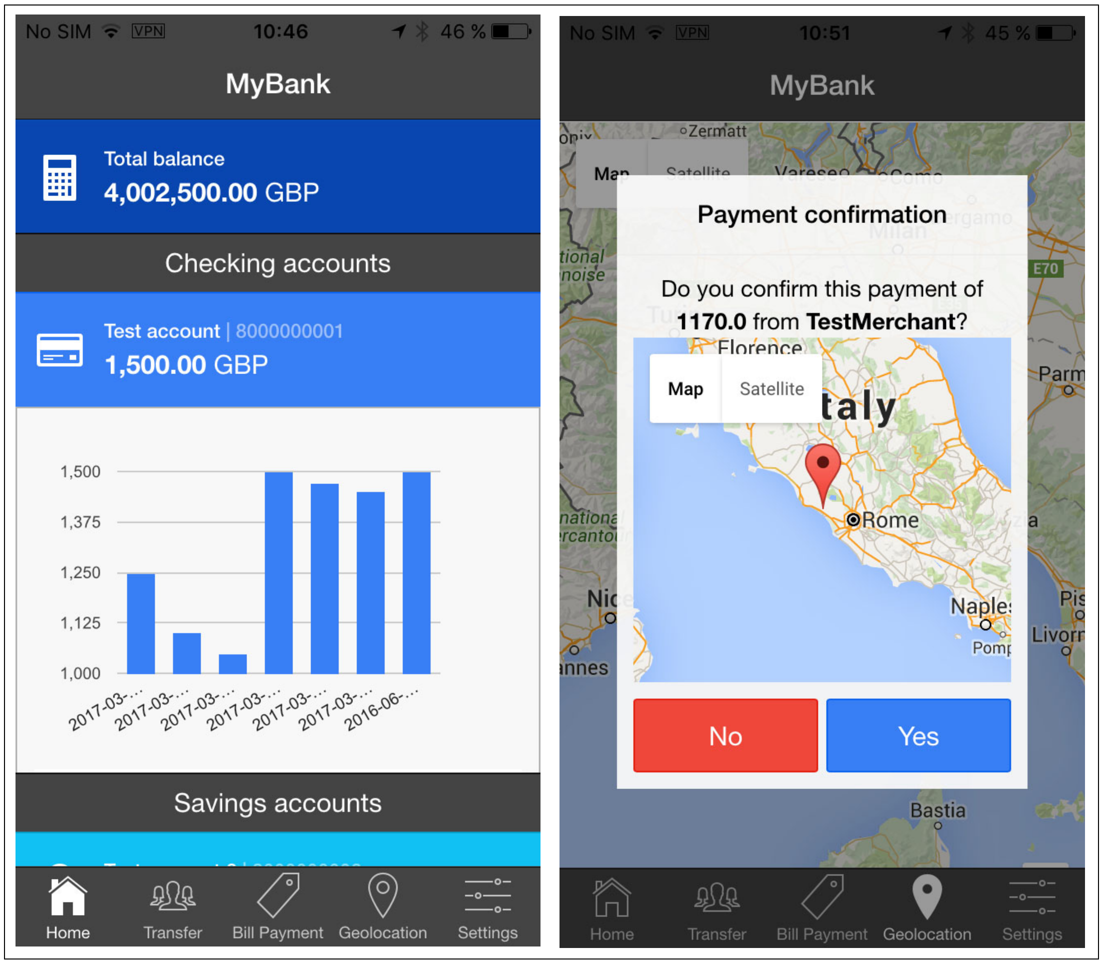


Figure C-1 Mobile app screen captures

Recently new scenarios based on the IBM Analytics portfolio (IBM DB2 Analytics Accelerator, SPSS®, Operational Decision Manager, and others) and products from independent software vendors (Apple, Zementis, Keyword, and others) have been integrated.

Figure C-2 shows the banking IT architecture, including the various channels and major software components. It is a logical representation or map of the business components or building blocks of the bank.

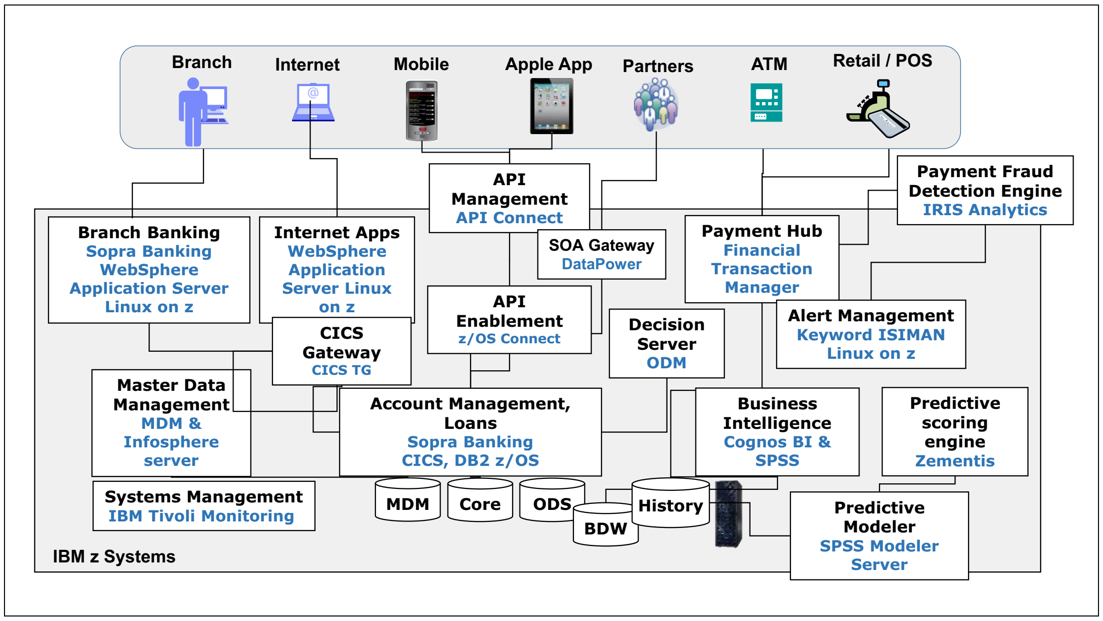

Branch
Internet
Mobile
Apple App
Partners
ATM
Retail / POS
API Management API Connect
Payment Fraud Detection Engine IRIS Analytics
Branch Banking Sopra Banking WebSphere Application Server Linux on z
Internet Apps WebSphere Application Server Linux on z
SOA Gateway DataPower
Payment Hub Financial Transaction Manager
API Enablement z/OS Connect
CICS Gateway CICS TG
Decision Server ODM
Alert Management Keyword ISIMAN Linux on z
Master Data Management MDM & Infosphere server
Account Management, Loans Sopra Banking CICS, DB2 z/OS
Business Intelligence Cognos BI & SPSS
Predictive scoring engine Zementis
Systems Management IBM Tivoli Monitoring
MDM
Core
ODS
BDW
History
Predictive Modeler SPSS Modeler Server
IBM z Systems

Figure C-2 Banking showcase architecture

The evolution of the showcase continues with the introduction of the following scenarios:

- Banking Digitalization:
  - Integration through APIs: Provide systems of engagements (mobile, digital, social apps) controlled access to data and services in systems of record.
  - Hybrid IT: Integrate traditional IT and private cloud environments with public cloud environments.
- Payments:
  - Payment Convergence: Rationalize multiple payment engines.
  - Payment APIs: Provide easier access to payment processing systems to registered third-party providers (TPPs).
  - Predictive Insight: Analyze payments data in order to improve fraud detection and to propose new services to clients.
  - Instant Payments: Implement instant (real-time) payments.
- Risk, Compliance, Security and Resilience:
  - IT Governance: How to handle the Payment Card Industry Data Security Standard (PCI-DSS).
  - Addressing new cyber security challenges: Prevent advanced threats and protect critical assets.
  - Choice for Optimization: IT resiliency with the GDPS active-active solution.
- Analytics and Cognitive:
  - Data Centric Design: Deliver a 360° customer insight view.
  - Analytics Acceleration: Enable advanced analytics to deliver best value from customer insights.

- Forthcoming innovations in banking:
  - Blockchain: Chart the progress of distributed ledger technology.
  - Watson: Embrace the next cognitive engagement.
- Innovation with open technologies and ecosystems:
  - Stack optimization: For Branch and Internet banking with independent software vendors including Sopra Banking, Zementis, and Keyword.
  - Multiple speed IT: Using DevOps to enable continuous build and test of banking solutions.

For more information and contacts for the IBM Banking Showcase, visit the IBM Client Center Montpellier web page:

http://www.ibm.com/ibm/clientcenter/montpellier/

Related publications

The publications listed in this section are considered particularly suitable for a more detailed discussion of the topics covered in this paper.

IBM Redbooks

The following IBM Redbooks publications provide additional information about the topic in this document. Note that some publications referenced in this list might be available in softcopy only.

- *System Programmer's Guide to Workload Manager*, SG24-6472
- *IBM CICS Performance Series: CICS TS V5.3 Benchmark on IBM z13,* REDP-5320

You can search for, view, download or order these documents and other Redbooks, Redpapers, Web Docs, draft and additional materials, at the following website:

ibm.com/redbooks

Other publications

These publications are also relevant as further information sources:

- *z/OS MVS Planning: Workload Management*, SC34-2662-04
- *Using the Sub-Capacity Reporting Tool*, SC23-6845
- WLM and RMF Reporting Enhancements for Mobile Workload Pricing, z/OS Hot Topics Newsletter (August 2016 - Issue 30)

Help from IBM

IBM Support and downloads

ibm.com/support

IBM Global Services

ibm.com/services

REDP-5359-00

ISBN 0738455504