Redbooks
ibm.com/redbooks

How Walmart Became a Cloud Services Provider with IBM CICS

Jennifer Foley

Randy Frerking

Mark Hollands

Rich Jackson

Kellie Mathis

Phil Wakelin

Matthew Webster

Cloud

z Systems

IBM

Redbooks

International Technical Support Organization

How Walmart Became a Cloud Services Provider with IBM CICS

April 2016

SG24-8347-00

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

First Edition (April 2016)

This edition applies to Version 5, Release 2 of IBM CICS Transaction Server for z/OS (product number 5655-Y04).

Contents

Notices

Trademarks

Preface

The world has changed. With the new cloud options, enterprises no longer must rely on only their IT organization to meet their computing needs. Business units now have options that were not available just a few years ago. They can get some of their needs met by traditional IT processes, and reach out to a cloud provider to meet other needs. The concern is that if you, working in a traditional IT organization, do not meet these needs, someone else will.

This IBM® Redbooks® publication helps you to understand the benefits of becoming your own cloud service provider. It describes a simple approach that allows you to be successful. The main focus of the book is lessons learned from the implementation by an IBM client, Walmart Stores, Inc.®, that achieved impressive results in their efforts to become their own cloud service provider to their developer community. In this way, Walmart successfully made z Systems a relevant part of their Hybrid Cloud strategy.

Walmart embarked on this journey to help their application developers achieve results that were previously time-consuming and difficult to deliver. In the process, they realized that they had everything that they needed to become a services provider to their developer community. This book describes the choices that Walmart made, and explains the steps they took to be successful.

The goal of the book is not to imply that the only way to achieve success is by following Walmart's process exactly. Rather, this book allows you to use the same basic constructs, but choose implementation details that fit your environment so that you can achieve success on your own terms. With IBM CICS® Transaction Server (TS) for z/OS®, you also have the resources for a successful transition to becoming your own cloud service provider.

IBM Design Thinking[1] is a methodology that is used by designers to solve complex problems by focusing on individual user roles. This book is organized from the viewpoint of these roles in the IT organization. It provides guidance in the following areas:

- What does the line of business expect from a cloud service?
- What topology and high availability characteristics does the system programmer need?
- What unique facilities does IBM CICS provide to the service developer?
- How does a developer discover and consume services in an application?
- How does the operations team manage the service in production?

One of the services that Walmart built and how the decisions made by each job role affected the overall outcome of the service are used as an example throughout this book. It shares the experience of the team that created this and other business critical cloud services that are all hosted in CICS. Comments from Walmart IT leaders that were captured during the authoring process are presented to emphasize why the company adopted cloud and how cloud has helped Walmart to achieve success.

Developers understand the risk protection that IT groups provide. They also understand that waiting to move applications to production, or for a service to be provisioned, compromises the agile environment required by today's businesses.

This book is intended for enterprise service providers looking to enable their developers to increase the speed at which functionality is delivered to the business. For more information about creating IBM z/OS cloud services, see *Creating IBM z/OS Cloud Services*, SG24-8324.

[1] The Making of IBM Design Thinking: http://www.ibm.com/design/social.shtml

Authors

This book was produced by a team of specialists from IBM and Walmart Technology, Walmart Stores Inc.

Figure 1 From left to right: Matthew Webster, Jennifer Foley, Rich Jackson, Phil Wakelin, Randy Frerking, Kellie Mathis, and Mark Hollands

Jennifer Foley is an integration architect in the Worldwide IBM Client Center for Systems Innovation. In this role, she works with clients and business partners to explore new ways to use IBM technology and offerings. She holds the IBM and The Open Group certifications for Master IT Specialist and Master Architect. Jennifer is the leader for the Internet of Things and Mobile Centers of Excellence in North America. She has contributed to two previous IBM Redbooks publications and blogs for IBM MobileFirst™.

Randy Frerking is an Enterprise Technical Expert in Infrastructure and Platform Services at Walmart Technology, Walmart Stores Inc. He is responsible for designing and developing foundational enterprise and cloud services to satisfy various IT and business needs. He provides training and socialization of technological concepts, influences direction, and promotes innovation throughout Walmart's IT organization. His primary focus is on IBM z Systems™ engineering and software development. He has 36 years of IBM CICS and mainframe middleware experience with 8 years at Walmart Technology.

Mark Hollands is a CICS Software Engineer working on the Customer in Production (CiP) team, which uses CICS to improve the customer experience. During his time in CICS CiP, Mark has specialized in Java technologies, in particular using the IBM WebSphere® Liberty Profile hosted in CICS in a production environment. Mark's previous experience includes working in the CICS Tools Development team and developing the CICS Configuration Manager and CICS Deployment Assistant plug-ins for IBM CICS Explorer®.

Rich Jackson is a Technical Expert in Infrastructure and Platform Services at Walmart Technology, Walmart Stores Inc. He is responsible for designing and developing foundational services to satisfy various IT and business needs. He provides training and socialization of technological concepts, influences direction, and promotes innovation throughout Walmart's IT organization. His primary focus is on IBM z Systems™, but he has participated in grid computing planning and design as well. He has been with Walmart for over 11 years with a background in storage and z/OS management.

Kellie Mathis is a Director in IBM Systems and has been with IBM for 31 years. Kellie works with clients to understand and develop the business justification and use cases for using IBM z Systems in modern ways. She helps clients overcome old paradigms of mainframes within their organizations and develop a modern plan to use the power of z/OS systems.

Phil Wakelin works for CICS development at IBM UK in Hursley, and is a member of the CICS strategy and design team. He has worked with many CICS technologies for the last 25 years, and now helps CICS customers adopt Java-based technology. He is the author of many white papers, IBM SupportPacs, and IBM Redbooks publications in the areas of CICS integration and Java support.

Matthew Webster is an IBM Senior Technical Staff Member on the CICS Transaction Server for z/OS team, based in Hursley, UK. He holds a BSc in Physics with Computer Science from Southampton University, and has over 25 years of experience as a software engineer at IBM leading a wide range of projects, from mainframe to open source. His areas of expertise include CICS TS, Java, agile software development, and cloud technologies. Matthew is a regular blogger, podcaster, and presenter, and has published articles and papers on CICS TS and software development.

The project that produced this publication was managed by **Marcela Adan**, IBM Redbooks Project Leader - International Technical Support Organization.

The IBM team would like to thank Walmart Technology for sharing their experience of developing cloud services and allowing Randy and Rich to commit so much time to the book, including travelling to the IBM Hursley development laboratory.

Thanks also to **Andy Bates**, CICS TS Offering Manager, IBM UK, for his contributions to this project.

Now you can become a published author, too!

Comments welcome

Your comments are important to us!

We want our books to be as helpful as possible. Send us your comments about this book 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

Reasons to become a cloud service provider

This chapter describes the original obstacle that was faced by the Walmart development team and how the engineering team met this challenge by providing a service that follows the cloud computing model. The authors describe the characteristics of this model and how both IBM z/OS and IBM CICS offer the capabilities to make these characteristics available.

In this chapter and throughout this book, the authors also consider the challenges that the cloud presents to traditional IT roles and how to address them.

This chapter includes the following topics:

1.1 The development obstacle

A team of developers working on an application for a non z Systems platform needed a distributed caching service. Their existing solution had performance and availability issues, did not scale, and was unreliable. They were facing a rapidly approaching deadline for their application to go into production. A solution to this problem was proposed by the CICS platform engineers, but this solution was met with skepticism because of perceived complexity and lack of agility of the mainframe platform

Walmart embarked on the journey to find a solution to this challenge with an eye toward cost neutrality, where there was no initial investment and the hope was that someday the project would be self-funding

1.2 The developer is the consumer

Have you noticed the attention that is being given to cloud service providers? The reason for this attention is that service providers are following a model where the developer is the customer and providers are catering to their needs. Expressed differently:

> *"Cloud is a consumer-driven market."*
>
> Rich Jackson, Enterprise Technical Expert at Walmart Technology, Walmart Stores Inc.

The reason that external cloud service providers are being heard is that they changed their behavior and focus to those of the consumer. In the case of Walmart, the consumer is the application developer. The user of a service ultimately decides the fate of the service. Even if a service is low-cost and reliable, developers will pursue services elsewhere if the service is not easy to consume and free.

Just as developers seek service providers that satisfy their needs, business units do the same. Ultimately, businesses do not need IT: They need capabilities. These capabilities include functions, such as servicing a customer, collecting money, and moving inventory. IT happens to be an effective way of providing these capabilities, but it is still only a means to an end. Business demands are dynamic and can change rapidly. IT must accommodate these changes as needed. Providing developers accessible, easy to use services allows them to be responsive to business needs.

Microservices architecture is becoming the standard approach for developing, deploying, and managing applications and application components. Small teams work independently and use their technology of choice to create services that are accessed by easily consumed application programming interfaces (APIs). CICS provides the multi-language run time for such teams on a robust, secure, highly scalable z Systems platform. The resulting microservices are stand-alone pieces of code, each with a finite function that can be accessed by anyone on the team and independently deployed. The APIs can call other services or be used alone. It is all about the service that is faster, cheaper, and more flexible than traditional IT provided.

Timely delivery of a business capability is the ultimate objective of any development activity. Developers understand the need for the risk protection that traditional IT groups provide. It is inefficient to wait to move applications to production, wait for a service to be provisioned, or to have so much governance that they cannot be agile. Walmart found that by focusing on and addressing the needs of the developer as the consumer, they increased the speed at which functionality is delivered to the business.

1.3 How z/OS and CICS are relevant to cloud

"The future of z/OS is taking something that is cloud-like and making it cloud."

Randy Frerking, Enterprise Technical Expert at Walmart Technology, Walmart Stores Inc.

The National Institute of Standards and Technology (NIST[1]) defines the five essential characteristics of cloud computing as shown in Figure 1-1. These characteristics can be satisfied by CICS Transaction Server (TS) on z/OS. You could argue that IBM Multiple Virtual Storage (IBM MVS™) was the original cloud system, as the functions in the operating system, together with the middleware and database, provided most of the characteristics of today's cloud systems. With CICS TS on z/OS, adopting cloud can deliver maximum business value from small changes to applications and processes.

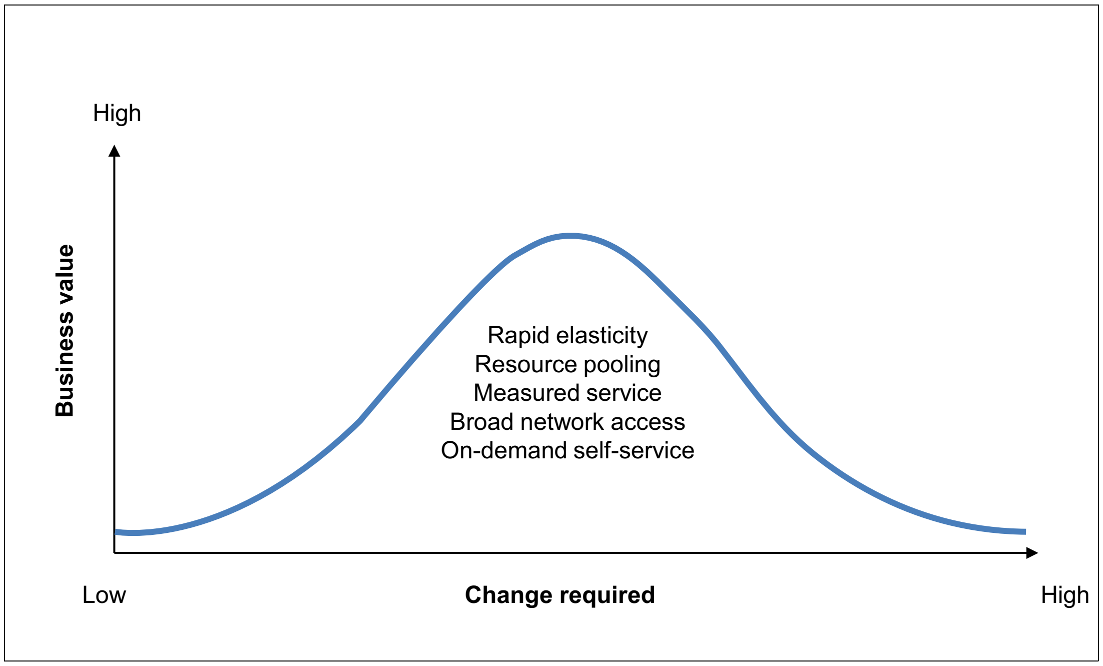


Figure 1-1 Adopting cloud delivers maximum business value from small changes

Web services (or in general terms, application services) provide broad network access to functions that are hosted in CICS. Support for SOAP web services has been available since CICS TS V3.1, with JSON web services support arriving in CICS TS V4.2. Use of IBM WebSphere Application Server Liberty profile for z/OS in CICS TS V5.1 further strengthened service enablement, providing servlet, RESTful, and SOAP web service options for Java applications.

[1] For more information, see this publication:
http://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-145.pdf

The approach that was taken by Walmart is based on ReSTful services and is described in Chapter 3, “The service provider” on page 13. Chapter 4, “The CICS systems programmer” on page 27 describes how CICS services are used.

The economics of cloud depends on resource pooling with multiple tenants sharing underlying infrastructure. The best practice for most CICS customers is to deploy a cluster of servers or cloned regions, and this approach is used by Walmart. The configuration that was chosen is described in Chapter 4, “The CICS systems programmer” on page 27.

Rapid elasticity on the z/OS platform is achieved by starting more servers and ensuring that those servers that are already running have the resources (for example, the CPU and storage) that they need when workloads increase. In addition to z/OS workload management (WLM), clients can take advantage of the 64-bit exploitation of CICS and the ability to run thread-safe applications. For more information about the approach that was taken by Walmart, see Chapter 5, “The z/OS systems programmer” on page 45.

A critical difference between service-oriented architecture (SOA) and cloud is the need to provide a measured service. The CICS monitoring facility (CMF) allows you to measure the resource consumption of every service request. For more information about how Walmart uses strict naming conventions with CMF to uniquely identify each service consumer, see Chapter 4, “The CICS systems programmer” on page 27.

As you know, the direction of a business is ultimately determined by its customers. The ability to quickly adapt business services to the needs of the customers requires on-demand self-service access to IT assets for Walmart developers. For more information about the user experience with a provisioning portal, see Chapter 2, “The service consumer” on page 7. For more information about how the developer experience fits into the broader perspective of DevOps, see Chapter 7, “DevOps perspective” on page 67. Walmart takes advantage of the web server that is includes with CICS to host this portal and uses resource definition online (RDO) and other facilities to dynamically create the resources required by each service instance.

1.3.1 The journey from development obstacle to cloud services

Walmart did not set out to explicitly use z/OS and CICS to deliver cloud services. The initial goal was to address the obstacle that the application development team was experiencing (see 1.1, “The development obstacle” on page 2).

While working on a solution, it became apparent that not only did CICS on z/OS have the capabilities that are needed to solve the issue, it also provided the means to deliver the capability *as a service*. If done correctly, this service could then be called by anyone who needs the same functionality and it could be called from any platform.

Benefits of a cloud solution

The easy way to address the development obstacle is to create a single enterprise service. However, it was at this point that the actions became intentional. By following the NIST standards for cloud services (which included the creation of a self-service provisioning mechanism and metering), a cloud service was born. After the model was developed, Walmart was able to reuse it. Many other z/OS cloud services were created later.

Part of the benefit of a cloud service is that it frees the consumer from having to know or be concerned about the operating environment hosting the service. The consumer can call the service as an HTTP endpoint, without concern for the underlying infrastructure. In fact, the original z/OS cloud service that was developed was created for consumption by a non z Systems team. The application development team needed a resolution to the development obstacle that was fast, inexpensive, and flexible. They got these features and more.

The first z/OS cloud service that Walmart created contained base functions and opened the door to a new way of satisfying the application developers. The new thinking became the idea of getting a service available quickly to support agility, and then to go back and make enhancements. These principles are the principles of the agile development methodology.

This thinking allows the ability to master a step, provide updates, and then master the next step. One of the key issues is to provide governance, which at first might be manual, but that can continue to evolve and become automated. Early feedback from the service consumer provided input for prioritization of new function and assurance that the service met the business need at hand.

It is important to understand that none of the consumers of the services that Walmart created sought out a CICS on z/OS solution. And inversely, the z/OS and CICS teams did not seek to be an enterprise cloud service provider. The goal was to meet the needs of the application developer. The outcome is that z/OS and CICS are now a private cloud service provider as part of a Walmart hybrid cloud strategy.

Evolving the solution

Walmart started its first service in their production sysplex environment and provided only a small amount of isolation by creating CICS servers (regions) that were specifically for the z/OS cloud service.

After it became apparent that the use of CICS cloud services would grow, creating a greenfield implementation that was reusable allowed the freedom to move away from dependencies on traditional systems. These traditional environments are commonly not ideal to host services because of high levels of dependencies among processes, the dynamics of workloads, and restrictive change management processes.

The difference is how the services are provided. Traditional environments typically follow a long change cycle, might deploy with a strategy that every system must be the same, and lose the ability to be agile. The cloud services sysplex is run with a thought of fit for purpose. For example, if one system can benefit from a newer code version, it is pushed to only that system to deliver the needed function. Languages and deployment environments are chosen based on which one meets the needs of the application.

Cost neutrality

Walmart embarked on this journey with an eye toward cost neutrality. One of the first services that was built performed much better than did the older application. It also returned savings in the form of resources saved; that is, a lower amount of capacity and memory were used. These savings were then used to create the next service. In this way, the cloud services environment was self-funding. The success demonstrated with this model led to the investment in the greenfield sysplexes.

1.4 Challenging the roles of IT

"I have always been driven to buck the system, to innovate, to take things beyond where they've been." [a]

Sam Walton, Founder, Walmart Stores, Inc.

a. Sam Walton: Made In America by Sam Walton and John Huey

With disruptive technologies continuing to emerge, traditional IT roles are no longer sufficient. Today's system programmer spends more time in the system administrator functions and less time with their application programming counterparts.

The CICS system programmer can tend to have a silo view and forget the effect on the operational environment. The operations team might forget that they are not just keeping things running, but they are servicing their client. In this way, all of the IT functions depend on the needs and successes of each other.

Walmart learned that by increasing the communication and knowledge across traditional roles, a more efficient and more functional system is created. The more feedback in the cycle, the faster the changes are delivered, with the continued focus on the needs of the service consumer.

Chapter 2, "The service consumer" on page 7, Chapter 3, "The service provider" on page 13, Chapter 4, "The CICS systems programmer" on page 27, and Chapter 5, "The z/OS systems programmer" on page 45 focus on the concerns and actions that a specific IT role should consider. An increased focus on collaboration between the roles enhances the ability of the service provider to meet the needs of the service consumer.

1.5 Summary

This chapter described what motivated Walmart to become a cloud services provider; removing obstacles to developer productivity, while keeping a keen eye on cost.

By following the five essential characteristics of cloud, Walmart maximized business value by exploiting a number of key capabilities of CICS TS on z/OS. The remaining chapters of this book help you achieve the same success.

2

The service consumer

The service consumer is the customer of the IT department in an organization. It is the responsibility of the IT department to determine the needs of the service consumer through continuous dialog.

The service consumer is primarily an application developer who is looking for a way to efficiently accomplish a repeatable task. This task is part of a larger process and commonly achieved through microservices in modern development approaches.

The background of the service consumers and their skills vary based on their application development experience. Regardless of background, the service consumers are focused on their specific area of work without concern or knowledge of the underlying intricacies of the systems that are hosting the services that they want to use.

The service consumers are expected to deliver code updates in short iterative cycles while also working on code defects. The quicker development cycle is driving the need for services that can be delivered quickly to the consumers. At the same time, consumers should not need to understand or be responsible for the underlying system components.

In this chapter, the authors share Walmart’s journey working with the consumers, the typical needs of the service consumers, and the criteria for success that Walmart discovered during the development of their initial service offering.

This chapter includes the following topics:

2.1 Consumer requirements

It is important to understand the requirements of the consumers before a solution is selected for a specific need. After the requirements, such as service level agreements (SLA), I/O rates, CPU rates, and accessibility needs are gathered, an informed decision can be made about the best way to satisfy those requirements.

This ideal approach is not always reflective of reality, though. Traditional processes in enterprise IT departments are commonly not conducive to the needs of the application developers.

The ability to deliver new capabilities quickly is as important as the ability to address technical specifications. Decisions based on speed might sometimes be made in lieu of properly addressing other requirements. This situation presents a challenge to infrastructure and platform engineers in many cases, but it also represents an opportunity. This scenario provided the platform engineers in Walmart an opportunity to truly address all of the needs of a consumer.

2.2 Walmart caching requirement

A team of Walmart developers required a distributed caching solution to support new features that were being incorporated into their application. This distributed non-z/OS application required storing and retrieving session information from any node in the server farm across which the application was hosted.

The functional requirements for the new capability were well-defined. The caching solution needed to employ ubiquitous communications protocols to ensure accessibility from a broad array of platforms and programming languages. The information that was being stored and retrieved required a unique key to identify the information, which might be in any number of formats and would be of variable length up to 3.2 MB. The solution also needed to support automatic expiration of cached objects at user-defined durations.

The non-functional requirements were not as clearly understood in the early stages of development. This issue led to the selection of a caching solution that did not meet the demands of the application. Problems with performance and availability became pervasive with the chosen product. An issue like this issue is never welcomed, but it did lead to increased communication between the developers and platform engineers and allowed the engineers to gain a better understanding of the needs of the application.

The increased communication also led to a better understanding by the platform engineers of the needs of the consumer. Awareness of functional and non-functional requirements is critical when providing a particular capability to solve a business problem. Ensuring that a capability is quickly acquirable, highly accessible, and flexible is as critical when providing for consumers' needs. This concept drives the need to deliver capabilities as services, and provides the basis for the cloud computing service delivery model.

This publication follows the creation of a caching service by the Walmart z/OS platform engineers based on the needs of the developers. The means by which the functional and non-functional requirements of the service were satisfied are described in Chapter 3, "The service provider" on page 13, Chapter 4, "The CICS systems programmer" on page 27, and Chapter 5, "The z/OS systems programmer" on page 45. The remainder of this chapter focuses primarily on needs of the consumer and how Walmart addressed these aspects.

2.3 Self-service

As the design for a caching solution was being solidified, the Walmart engineers also remained focused on satisfying the consumers. Based on a recognized need within their environment (along with an awareness of trends in the IT industry), the engineers decided to ensure that their solution did satisfy technical requirements and would be easily acquired by the developers.

The engineers iterated through numerous builds of caching service instances to ensure that the process was fully documented and understood. Then, they automated the build process. With the automated process in place, consumers could contact the engineering team and acquire a service instance in a matter of minutes. To further improve consumer responsiveness, the engineers developed an online web portal with which consumers can start requests and acquire service instances within seconds.

Consumer feedback continues to drive the capabilities that were enabled by the service providers. The evolution of the request portal is shaped by ongoing input from the service consumers. The current iteration of the self-service facility is based on a service-enabled design and includes APIs for the provisioning functions that can be used from any other entry point in the network. The high-level architecture of this capability is shown in Figure 2-1. For more information about the self-service facility, see Chapter 3, “The service provider” on page 13.

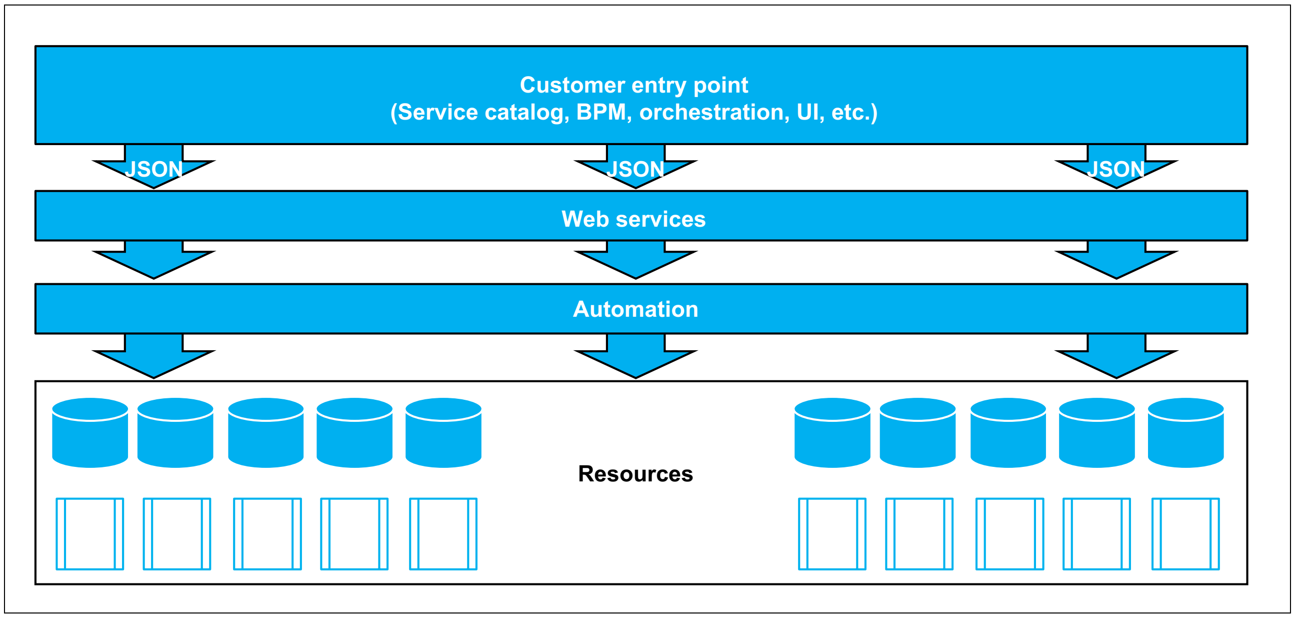


Figure 2-1 Service provisioning architecture

The ability for consumers to quickly and autonomously acquire a particular IT capability is immensely powerful. Immediate benefits include increased agility through the ability to deliver new business capabilities faster and to minimize negative effects by addressing defects and issues more responsively. However, the indirect benefits of this consumer-oriented feature are as compelling, if not more so. On-demand access to services enables experimentation, which allows creativity and innovation to flourish in the IT organization.

2.4 Connecting platforms

Most enterprise IT environments consist of various platforms, operating systems, and programming languages, each with particular features and capabilities that are suited for a specific need in the organization. Often, there are different groups of consumers that are related to each technology, which results in a diverse set of consumer needs. However, there also can be certain requirements that are common across the different groups of consumers.

A prime example of common concern, which is born out of this diversity, is the need for ubiquitous communications mechanisms. It often is necessary for various platforms to communicate and share information. Consumers want the services that they use and develop to integrate easily within the environment.

Communication across platforms is achieved with message queuing products, such as IBM MQ, in an asynchronous manner, or over HTTP, which can be used in synchronous or asynchronous ways. HTTP is most commonly employed by using SOAP web services or the ReST architectural style.

Of the cross-platform communication options available, ReST over HTTP provides the broadest compatibility, while being light-weight, highly flexible, and easy to use. For these reasons, ReST became the method that is expected by service consumers and the de facto standard in cloud computing. For more information about ReST, see Chapter 3, “The service provider” on page 13.

On occasion, there might be some overlap of the service requirements and consumer needs. In the case of the caching service in Walmart, the application required the cross platform communications that are provided by HTTP and the flexible data formats that are provided by ReST.

However, the application developers requested particular functionality with the HTTP requests that was a deviation from the specification. The change was intended to provide ease of use and simplify development activities. The consumers requested the following behavior:

- POST: Insert information into the cache table for new keys (normal behavior). If the particular key exists, replace the information instead of failing the request as can be typical in this scenario.
- PUT: Update information in the cache table for keys (normal behavior). If the key does not exist, POST the new information instead of failing the request as can be typical in this scenario.
- GET: Retrieve information from the cache table by using the provided key (normal behavior).
- DELETE: Delete information from the cache table by using the provided key (normal behavior).

Although this behavior departs from a strict adherence to the HTTP standard, it was implemented to address the needs of the consumer by reducing development effort.

2.5 CICS as a consumer

Although most consumers of the caching service in Walmart are x86-based applications, there are also CICS applications that use the service. It is worth noting the basic requirements here for the use of services from CICS.

As a CICS consumer or client, the following API commands are required:

- `EXEC CICS WEB OPEN`
- `EXEC CICS WEB SEND`
- `EXEC CICS WEB RECEIVE`
- `EXEC CICS WEB CLOSE`

The **`WEB SEND`** and **`WEB RECEIVE`** commands can be consolidated as one by using the **`EXEC CICS WEB CONVERSE`** CICS API.

When a service is consumed, a CICS application must know the URI of the target system and service. This information can be defined to CICS in a client URIMAP.

Figure 2-2 shows an example of a client URIMAP that contains the path name, host name, and port of the target server and service.

```
OVERTYPE TO MODIFY                                        CICS RELEASE = 0690
   CEDA  ALter Urimap( UC01TEST )
    Urimap         : UC01TEST
    Group          : UC01URI
    DEScription  ==>
    STatus       ==> Enabled            Enabled | Disabled
    USAge        ==> Client             Server | Client | Pipeline | Atom
                                        | Jvmserver
   UNIVERSAL RESOURCE IDENTIFIER
    SCheme       ==> HTTP               HTTP | HTTPS
    POrt         ==> 50001              No | 1-65535
    HOST         ==> User0001.cache.hostname.com
    (Mixed Case) ==>
    PAth         ==> /cache/v0.1.0/dept/div/tenant_0001/
    (Mixed Case) ==>
                 ==>
                 ==>
                 ==>
```

Figure 2-2 RDO example of a URIMAP client

Figure 2-3 shows how a CICS consumer program might issue an **`EXEC CICS INQUIRE URIMAP`** command to retrieve URIMAP information, such as the host name, port, and path names, which are used on the **`WEB OPEN`** and **`WEB SEND`** commands.

```
TRANSACTION: RREQ PROGRAM: RESTREQ  TASK: 0031984 APPLID: TEST001  DISPLAY:  00
  STATUS:  COMMAND EXECUTION COMPLETE
  EXEC CICS INQUIRE URIMAP
   URIMAP ('UC01TEST')
   HOST ('user0001.cache.hostname.com:50001                               '...)
   PATH ('/cache/v0.1.0/dept/div/tenant_0001/                             '...)
   PORT (50001)
   SCHEME (1096)
   NOHANDLE
```

Figure 2-3 Example of an EXEC CICS INQUIRE URIMAP command (in CEDF)

2.6 Summary

This chapter highlighted the importance of fully understanding all needs of the service consumer. Before deciding on architecture or design, these requirements must be gathered. They then form the basis of the choices of a service provider when a service is designed and how to deliver it.

Whatever the requirements, z/OS and CICS can be a beneficial place to host services, with no need for your service consumers to understand the intricacies of z Systems.

For more information about the requirements of the consumer and the responsibilities of the service provider, see Chapter 3, “The service provider” on page 13.

3 The service provider

The service provider is the IT department in an organization. Because of the complexity of the role of service provider, this role is not likely to be performed by a single person but by a combination of people with shared understanding.

It is the responsibility of the service provider to work with the service consumers to understand their issues and requirements. After the providers understand the needs of the customers (the service consumers), they can use their own knowledge and experience from working with the resources within the organization to create a service that provides value for the service consumers.

Creating these services also requires the service provider to use coding skills to develop the service and the service infrastructure. The service provider also must understand (based on the needs of the service consumer) the capacity requirements to support a service.

This chapter describes the reasons why Walmart decided to use IBM CICS to create a caching service, the characteristics of a microservices architecture, and how the service is a cloud offering based on the five essential characteristics of cloud, as outlined by the US National Institute of Standards and Technology (NIST).

This chapter includes the following topics:

- 3.1, “Why Walmart chose CICS for the caching service” on page 14
- 3.2, “Foundational underpinnings” on page 17
- 3.3, “Five essential characteristics of cloud” on page 17
- 3.4, “Microservices architecture” on page 24
- 3.5, “Summary” on page 25

3.1 Why Walmart chose CICS for the caching service

Based on the requirements of the development team for a distributed caching solution, the platform engineers team began assessing several off-the-shelf solutions in software and appliance form.

They also assessed the potential of building their own caching solution. The platform engineers built a solution that was stable, cost-effective, and allowed easy integration with components of the application that were being developed by the development team. This solution was realized by using resources and the unique capabilities of the IBM z/OS and IBM CICS Transaction Server platform.

3.1.1 Mixed language support

CICS provides a highly versatile environment for hosting applications and services. The range of language support gives developers or service providers many options, but there are workload considerations that should guide the selection.

Selecting HLASM and COBOL as service development languages

The skill set of the Walmart's software engineers who originally started the CICS based cloud services creation had some bearing on the selection of the programming languages that were used to create the services. The use of the skills and languages that the engineers were familiar with provided benefits to getting started more quickly.

There were other measurable factors that supported selecting these languages. The primary example is execution efficiency. The use of lower-level languages enables more control of the code path and resultant resource usage. Making deep execution adjustments that eliminate microseconds from a process might appear unnecessary (and can be in some use cases). However, the cumulative effect of this resource saving can become meaningful when operating at a substantial scale. At several thousand transactions per second, the difference can result in the need for an entire extra engine. At tens of millions of transactions over the course of a day, it can mean more hours of run time.

The good news is that CICS provides the flexibility to make these choices. With its support for various programming languages, the appropriate tool for virtually any job is available. Service providers can select High Level Assembler (HLASM) or C where low-level particulars or efficiency is important. Higher-level languages, such as COBOL or Java, or even scripting with REXX or PHP, can be selected where quick delivery of functions is needed.

For more information about the languages that are supported by CICS, see *CICS Transaction Server for z/OS* at the IBM Knowledge Center that is available at this website:

https://ibm.biz/Bd4mYf

3.1.2 Everything in the box

CICS on z/OS provides a complete platform that includes all components that might be needed for hosting and managing services, along with consumable entry points into the layers of the overall stack.

Native CICS APIs

To satisfy the requirement for HTTP-based communications with the caching service (and most cloud services), the following relevant APIs were used:

- EXEC CICS WEB EXTRACT: Gets HTTP information about the incoming request.
- EXEC CICS WEB RECEIVE: Gets request information, such as the HTTP body, from the consumer.
- EXEC CICS WEB SEND: Sends the response to the consumer

The following optional APIs proved to be useful with HTTP-based communications:

- EXEC CICS WEB READ HTTPHEADER: Gets HTTP header fields and information.
- EXEC CICS WEB WRITE HTTPHEADER: Creates HTTP header fields and information

The range of native APIs that are available in CICS is extensive and enables greater function and flexibility for programs that are hosted in that environment. API commands that facilitate programming functions and direct integration with hosting environment capabilities deliver a great amount of value.

z/OS APIs

The system-level APIs that are available to CICS and its hosted services further increase the value proposition. The available APIs extend the reach of functions down into the operating system and other subsystems to provide a fully realized, bottom-to-top, integrated environment that service developers can traverse.

The following APIs allow CICS to interface with system level components:

- EXEC CICS VERIFY: Checks with an external security manager to authenticate users.
- EXEC CICS ENQ/DEQ: It is used to manage serialization of various system resources.
- EXEC CICS ASKTIME: Requests information from the sysplex timer.

Software-defined environment

When this extensive range of application, host, and system level APIs is considered (along with the virtualization of nearly every resource in the environment and built-in workload management capabilities), the result is effectively an entire software-defined environment that is available to programmers.

From application logic and CICS APIs, through dynamic hosting environment configuration with resource definition online (RDO) and system programming interface (SPI), down to manipulating low-level system resources with supervisor calls (SVC), anything is possible. These factors make z/OS and CICS a wanted platform for services.

3.1.3 Monitoring and diagnostics

With an extensive history of operational considerations for the most critical workloads, across numerous industries, and for the efficient use of precious assets, CICS and z/OS evolved with a high degree of visibility into even the most granular level of system resource usage. This visibility can be achieved through inherent components of z/OS, such as IBM Resource Measurement Facility™ (IBM RMF™) and System Management Facility (SMF), or with any number of third-party tools. Access to this information enables operational stability and supports the cloud computing characteristic of *measured service* as defined by NIST.

Examples of features and characteristics that are relevant to monitoring and diagnostics are provided in the following sections.

Isolated resources
Isolating resources is a key component of the z/OS architecture and enables a certain degree of assurance to individual processes that their resources are available and can be tracked in a typically highly multi-tenant environment.

Isolation can be applied at all levels of the platform stack. At the lower level, IBM Processor Resource/Systems Manager™ (IBM PR/SM™) can be used to contain hardware resources at logical partition (LPAR) boundaries. At higher levels of the stack, workloads might be isolated to particular regions and controlled at the address space level. At a more granular level, the resource consumption of individual transactions or service instances can be managed with constructs, such as transaction classes.

I/O visibility
One of the strengths of the z Systems platform is its I/O capabilities. The offload of I/O processing to dedicated processors contributes to this reputation, as does the system and software components for tracking and managing I/O, which are also important. Identifying data set level I/O rates, the amount of time waiting on a channel, the rate of cache hits on an array, or any number of other conditions is critical to maintaining a healthy I/O environment and satisfying throughput expectations.

Visibility at TCB level
Low-level processing characteristics is another attribute that is beneficial to service providers. Determining how long a task is in a task control block (TCB) or how many times a task is switching between TCBs is critical to performance and resource management in the environment.

3.1.4 Vendor environment

Although a substantial amount of visibility is delivered by default with the platform, a broad and mature vendor tools system is available for CICS and z/OS environments. Tools are available to complement almost any aspect of the platform. Some of these tools are described in this section.

Monitoring packages
Some third-party monitoring packages can take innate monitoring platform capabilities and supercharge them, which enables even more granularity and related information.

Business continuance
Concerns about processing continuance during or after disruptions are always important, even on platforms with 99.999% availability (often called *five 9s* platform). Meeting the service requirements of the consumer, even if there is any type of disruption, is an essential consideration; many tools are available to help with this goal.

Automation packages
Along with monitoring, automation is relevant in a consumer-oriented delivery model, and many vendors contribute with automation packages. Automation can include the following aspects of delivering a service:

- Provisioning
- Diagnostics
- Governance
- Scaling
- Decommissioning

3.2 Foundational underpinnings

There are some basic underlying principles to establish before delving into the characteristics that are essential for a cloud service. These foundational components greatly facilitate the realization of several, if not all, of the requirements for a cloud service.

3.2.1 Naming conventions

Well-thought out and well-designed naming conventions can be useful and they help to enable the cloud essential characteristics of a service design, as described in 3.3, “Five essential characteristics of cloud” on page 17. These naming conventions create logical relationships between the various resources that comprise a service. For more information about the importance and relevance to the parts of the delivery model, see 3.3.5, “Measured service” on page 23, Chapter 4, “The CICS systems programmer” on page 27, and Chapter 6, “Operational considerations” on page 59.

3.2.2 Standards

The US NIST definition of cloud computing includes the concept of a type of deployment model that is referred to as *hybrid cloud*, which is described as a composite of two or more distinct cloud infrastructures. Each distinct cloud infrastructure remains and retains its unique attributes, but connectivity or communication across the members of the hybrid cloud is made possible. The portability of data and applications is enabled.

This deployment model is the most applicable to most enterprises. Interoperability is key in a hybrid cloud deployment model. Adherence to open standards is critical to maintaining compatibility with a diverse set of services. The importance of standards is relevant, but not limited to, the *broad network access* essential characteristic that is described in 3.3.2, “Broad network access” on page 20.

3.3 Five essential characteristics of cloud

Various demands from IT services consumers led to the evolution of how IT capabilities are provided. These new delivery methods coalesced into what it is now called *cloud computing*. Although a significant amount of ambiguity with this term followed its evolution, a de facto standard for definition of the model was eventually established.

NIST went through numerous iterations in establishing its final definition of cloud computing, which is outlined in NIST publication SP800-145 and became the accepted standard. Aside from the service and deployment models that are outlined in the publication, the following essential characteristics were established as defining attributes for a cloud service:

- On-demand self-service
- Broad network access
- Resource pooling
- Rapid elasticity
- Measured service

These characteristics are used as the guiding principles for design and development of a cloud service. As a service provider, these requirements are central to delivering capabilities going forward.

3.3.1 On-demand self-service

This characteristic specifies that the consumer can autonomously acquire services with no interaction with the service provider. The access point to the service request might be available through some sort of user interface or programmatically; for example, by using an application programming interface (API).

Although not explicitly called out, provisioning the service is expected to be automated. Even with the absence consumer-to-provider interaction, any manual provider actions on the back end should be avoided.

Automation

Providing on-demand self-service often involves a substantial time investment, as automation must be built. Service provider participants with programming knowledge might be needed to contribute to the automation.

A service ultimately is a composition of parts. Examples of parts include data sets, programs, parameter or configuration files, transactions, and definitions. A thorough inventory and review is necessary to identify all of the components that are required for the service instance.

After all components and steps are identified, creating the service can be performed programmatically. The service can be created in various ways by using the tools, languages, and utilities that are most appropriate for the task at hand. This process can include a series of JCL steps, a REXX script, COBOL programs, or some combination of these options and other options. These selections depend on specific use cases and requirements. Details, such as the type of runtime environment that is used, the particular actions to perform, and the type of resources to define determine the appropriate approach and tools to use.

Upon successful automation of the service creation, the capability must be made available to consumers. Generally, use standards-based web services as the channel for invocation of the service provisioning to ensure accessibility to the broadest range of clients.

Resources

Consider the following points regarding automating resources:

- Use the CICS resource definition online (RDO) facility for dynamic resource definition and modification in support of service capabilities.
- Well thought out naming conventions and the ability to use patterns facilitates the automation efforts and supports some of the other essential characteristics of cloud. It is important to fully understand the naming restrictions of all resource types before deciding on a naming convention.
- Use the CICS system programming interface (SPI) commands to automate and modify other system components

For more information about relevant resource configurations and modifications, see Chapter 4, “The CICS systems programmer” on page 27.

Consumer access

The consumer needs some form of entry point or access to provisioning automation. Although some service requests are sent programmatically, most requests often come directly from consumers who are interacting with some form of online menu or catalog of service offerings.

How the customer entry point fits into the on-demand self-service infrastructure is shown in Figure 3-1. This entry point can be a service catalog system, orchestrator, process or workflow system, or home-grown front-end service portal. Similar to cloud services, the use of open standards and a service-oriented approach ensures broad accessibility, manageability, and extensibility of the provisioning framework.

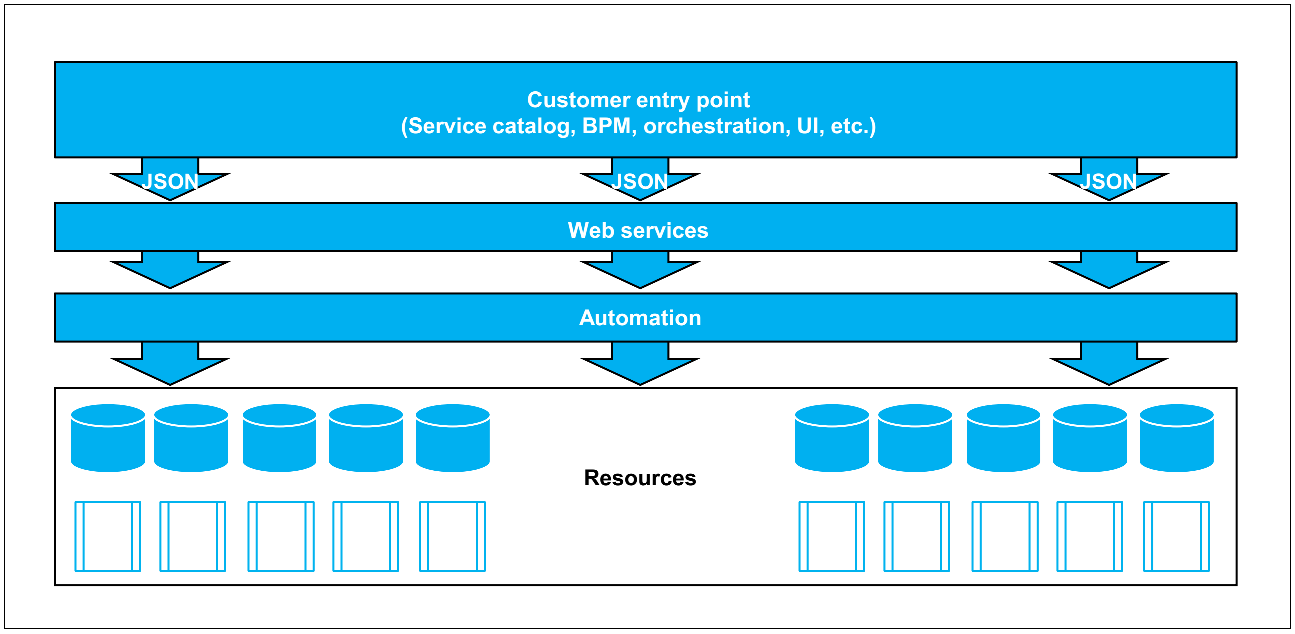


Figure 3-1   On-demand self-service infrastructure

Although this architecture allows provisioning requests to come from any source that can issue the appropriate web service request, the primary entry point for the consumers in Walmart is a self-service portal created by the engineers. This self-service web application is hosted entirely in CICS. It consists of an HTML5/CSS3/Javascript front end that is on a z/OS file system (zFS) and is referenced through URIMAP definitions. The JavaScript code validates form input, formats it as JSON, and sends the request to the back-end web services that start the provisioning automation. Figure 3-2 on page 20 shows the design of Walmart's self-service web application.

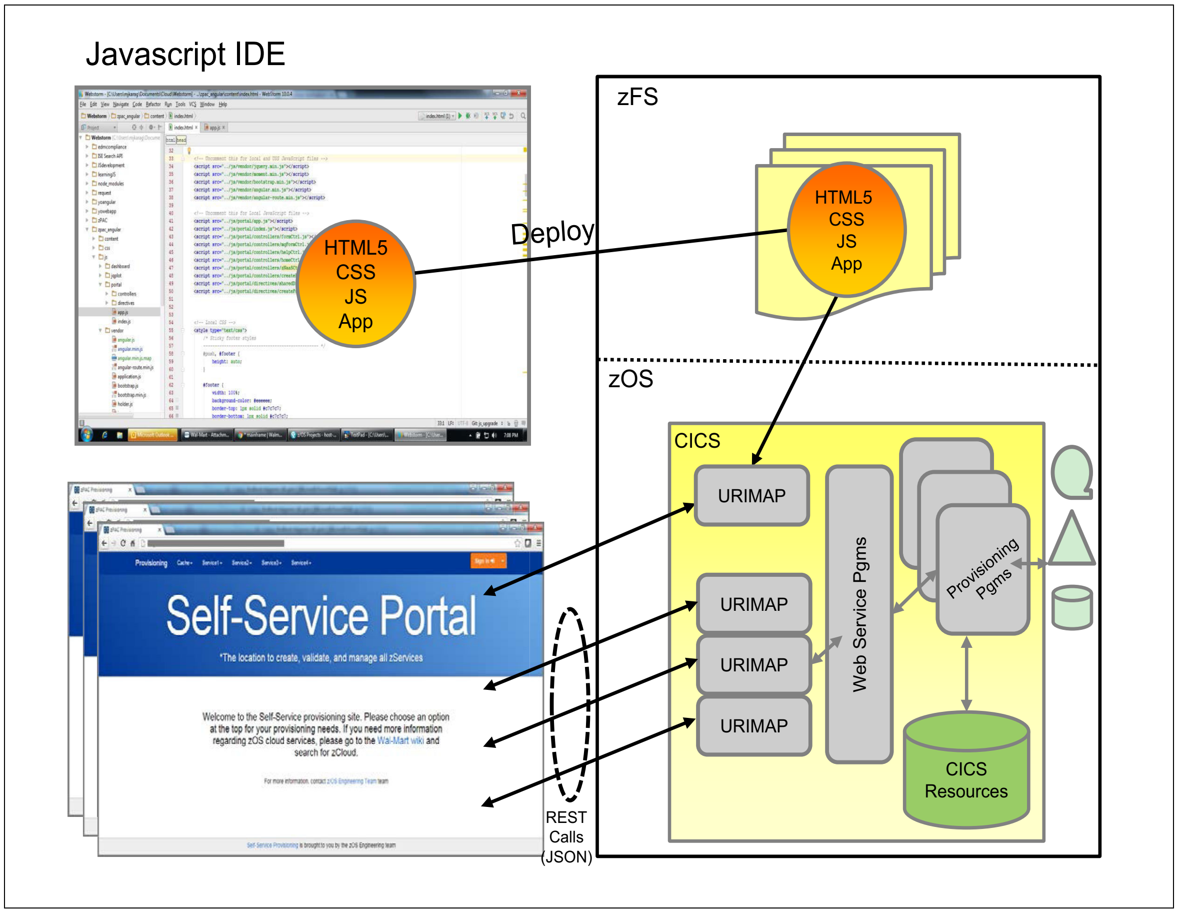


Figure 3-2 Self-service web application design

3.3.2 Broad network access

The essential cloud characteristic of broad network access (ubiquitous access) dictates that capabilities are accessible through standard interfaces over a network. In essence, broad network access is delivering capabilities through web services.

Standards

The use of standards is key to enabling the broad network access cloud characteristic. Use of standards for transport (as in TCP/IP and HTTP) and accessibility (such as URI and JSON) ensures that service capabilities are available to the broadest range of platforms and clients. Another benefit that is provided by this model is the resulting abstraction and loose coupling of the clients from the specific systems of service providers.

ReSTful services

Although standards that are associated with SOAP web services were once a primary option, Representational State Transfer (ReST) became the de facto software architectural style for cloud services today. ReST involves a much lighter weight interface with less supporting infrastructure. Unlike SOAP, ReST includes the following characteristics:

- Runs over only HTTP and does not support context or state management.
- The rules for ReST are much less stringent. SOAP requires XML. ReST often uses JSON, but does not require it.

Access to services via ReST uses the standard Universal Resource Identifier (URI) and Universal Resource Locator (URL) formats across Internet Protocol networks that use HTTP. A URL is a specific type of URI. A URL typically locates a resource on the Internet or within your network, and is used when a web client makes a request to a server for a resource.

A URL for HTTP or HTTPS is made up of up to five components (scheme, host, port, path, and query string). These components are combined and delimited as shown in the following example:

```
scheme://host:[port]/path[?querystring]
```

This example includes the following components:

- Scheme

  The scheme identifies the protocol to be used to access the resource over TCP/IP. It can be HTTP (without SSL or TLS) or HTTPS (with SSL or TLS) followed by a colon and two forward slashes.

- Host

  This component of the URL can be a host name or an IP address and it identifies the host (system) where services are located. A host system provides the environment where services run and access requested resources. Services on this host might direct work to other systems (systems that host other CICS servers and associated resources) that support the operations of the service.

- Port

  A specific port number can also be specified in addition to the host component. If a port number is specified, this number follows the host name, separated by a colon. When omitted, port 80 is the default for HTTP and 443 is the default for HTTPS.

  In the Walmart z/OS cloud environment, the port is used to provide multitenancy, with all service instances directed at a specific TCP/IP service port number. This approach enables the host and port in the URL to direct work to the specific CICS servers where the service and subsequent resources are defined.

- Path

  The path identifies the specific resource on the host that the HTTP client wants to access. The path is defined by using a CICS URIMAP resource. When defining the path, you should place an asterisk (*) after the last forward slash (/) in the URIMAP, which enables generic paths to access the same service. It also allows information, such as a record key, to be included in the URL.

  The path name begins with a single forward slash and can contain multiple nodes (levels), each separated by a forward slash. The last node of the path ends with a single forward slash. Additional information, such as record key, can be included in the URL after the last slash of the path.

- Query string

  If a query string is provided, it follows the path component and provides a string of information that the service can use. The query string is prefixed with a question mark (?) character. The format of the query string is free form and specific for the service. A string of name-value pairs that are separated by an ampersand (&) is shown in the following example:

  ```
  ?FirstName=John&LastName=Doe
  ```

The scheme and host components of a URL are not case-sensitive, but the path and query string are case-sensitive. For simplicity, the entire URL usually is specified in lowercase. The path can contain one or more *nodes* (identifiers), which are separated by a forward slash (/), where information after the last node can represent user supplied information, such as a record key.

A URL with one node in the path with a key provided in the query string is shown in the following example:

```
http://hostdomain:55123/SalesInfo/?key=1234
```

A URL with two nodes in the path with a key provided in the query string is shown in the following example:

```
http://hostdomain:55123/SalesInfo/Regional/?key=1234
```

A URL with two nodes in the path with a key provided in the URL is shown in the following example:

```
http://hostdomain.55123/SalesInfo/Regional/1234
```

For more information about ReST, see *Creating IBM z/OS Cloud Services*, SG24-8324.

Note: The Internet Engineering Task Force (IETF) is the authoritative source and should be referenced to ensure adherence to applicable standards.

3.3.3 Resource pooling

A wanted outcome of efforts to implement cloud services includes providing secure multitenancy in a shared hosting environment. That environment should manage resources that are pooled, and allow sharing and efficient use of resources in a manner that the resource users are oblivious to other users of the same resources.

Computer resources, such as memory, processing capacity, and storage, are represented as aggregates and appropriate amounts of resources are assigned to individual consumers.

These resources are dynamically allocated, deallocated, and reallocated as needed. By this allocation process, the set of resources that are available are used as a pool to allocate from or to be returned to. The pool size can also increase or decrease as needed to support the various services and tenants that are hosted in the environment. The increase and decrease of the pool size can be dynamic or controlled through operator or administrator intervention.

Inherent pools

If you have z/OS experience, you are familiar with the concept of resource pooling. IBM z Systems evolved from a platform that was designed as a multi-tenant environment with access to pooled-resources. Resource pooling is part of z Systems.

This inherent characteristic of the platform makes it straightforward to enable some amount of resource pooling. Virtually any processing (even existing processes) uses pooled resources to some extent.

Even beyond the resource pooling that is provided as part of the platform, other forms of resource pooling can be configured and used for service delivery; for example, a group of highly available CICS regions to serve tenant requests.

The use of IBM Parallel Sysplex® can greatly expand the available resources of a hosting environment by creating a collection of pools for some resources. Also, larger, broader pools of other resources are distributed across the sysplex.

A combination of automated processes and controls are necessary to assign and distribute resources.

These controls include workload management (WLM) policies and service classes. They facilitate controlling workload priorities. Storage management system (SMS) constructs can be used to control disk storage allocations and performance characteristics. This is not to say that more mechanisms are not needed to accomplish the required level of resource management. Instead, most general needs are covered by default and specific custom needs can be addressed individually.

3.3.4 Rapid elasticity

Elasticity refers to the ability of capacity to adapt to increases and decreases in workload demand. This characteristic is closely related to the pooled resource characteristic in that resource assignment and reassignment are necessary to achieve elasticity.

Scale up or out, and scale down or in

An interesting component of the elasticity characteristic is the scale-back or scale-down action and the primary driving factor behind it. The scale-back or scale-down action is a cost controlling measure that is primarily concerned with avoiding over-provisioned resources. This area is another instance in which the unique nature of z/OS provides some inherent relief in satisfying this characteristic because of negligible cost that is associated with certain resources being over provisioned.

For more information about the role of WML in satisfying rapid elasticity on workload manager, see 4.2, “Service owning region” on page 28 and 5.2, “Workload manager” on page 47.

3.3.5 Measured service

This characteristic is concerned with tracking who is using which resources and reporting the respective information back to the involved parties, such as consumers, providers, operations team, and capacity planners. Resource usage must be monitored and reported at the appropriate level of granularity for the type of service and the intended audience.

Consumers often receive an abstracted, higher-level contextual view of the usage data. From this perspective, the information usually is provided on a per unit basis, such as number of requests, number of users, and megabytes used. This abstracted level of per unit reporting requires that a model or an approach is created to aggregate the costs of the various lower-level resources that are required to provide the service.

This abstracted view also can be useful for service providers, but providers also require more granular visibility. The service providers often are concerned with the operational health of the environment and the services that are being hosted there, so a deeper view of resource utilization is needed.

From these different perspectives, metering information becomes used for the following purposes:

- Provides information about the IT costs that result from resource use and enables service providers to put in place chargeback or showback mechanisms.

- Is used to direct resource assignment decisions.
- Is relied upon to determine health of the environment.
- Is used for other various operational considerations and decisions.

For the caching service, Walmart's engineers elected to aggregate the reporting view at peak request volume to keep it simple for the consumer.

The service consumers receive a view of total requests volume per 10-minute interval over time. The service consumers are expected to maintain peak activity within a 24-hour period close to projected values.

The service providers monitor this same information. They also review capacity usage per transaction (that is, per service request) at the region level, and at the LPAR and sysplex levels to capture different views into the operational state.

A benefit of this visibility (particularly from the consumer perspective) is that it should ultimately promote self-governance of resource usage. Consumers can monitor the cost of their usage on an on-going basis, and they have the data that they need to make informed decisions on continued operational cost assignments.

Naming conventions

Well thought out naming conventions are important and their use is highly recommended to identify and report on resources. Having identifying characteristics within the names of resources, such as data set names and transaction IDs, can make it easier to tie together resources and a consumer with native reporting tools. This ability must be balanced with the naming restrictions that are associated with each resource type. Use of naming conventions might not be possible in all cases; therefore, other mechanisms must be employed to track resource usage.

3.4 Microservices architecture

The microservices architecture style is an emerging approach to application development with applications being composed of collections of independent, limited function services that are commonly accessed via HTTP.

3.4.1 Application design pattern

The overall design pattern of a microservices-based application involves a more compartmentalized approach to development than traditional monolithic methods. It is similar in some regards to service-oriented architecture (SOA), but it is lighter weight, less rigid, and more flexible.

3.4.2 Independent, limited function services

Historically, applications tend to consist of logical components. The microservices design can be viewed as the externalization of these logical components to individual independent functions delivered as services. As shown in Figure 3-3 on page 25, the result is the ability to compose applications in a building block type manner.

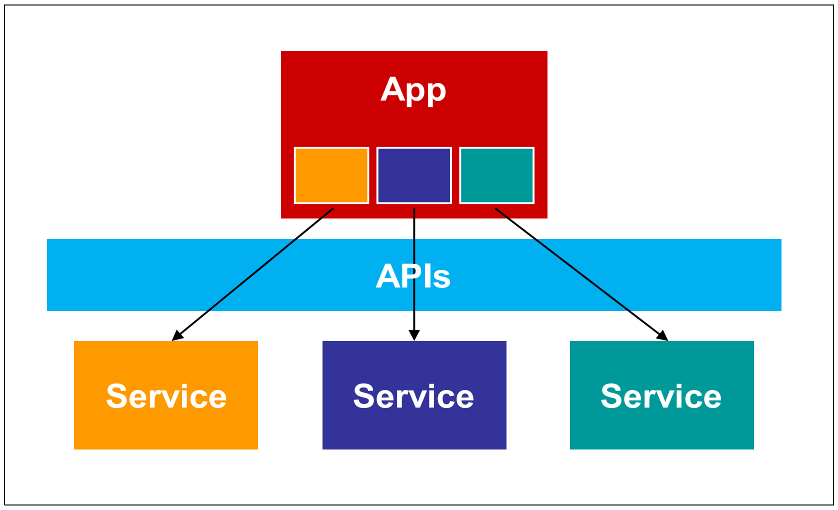


Figure 3-3 Microservices architecture

Although this approach can introduce some complexity, it also provides significant value. Flexibility in deployments, avoidance of rework, greater manageability of individual functions, and scalability of distinct components all contribute to more dynamic and agile applications.

In this IBM Redbooks publication, this concept applies to net new capabilities that are delivered as services. It also likely involves the decomposition of applications into modularized services to tap into their value. For more information, see *Creating IBM z/OS Cloud Services*, SG24-8324.

3.4.3 Encouraged by cloud delivery model

The modularized microservices also must be acquired or consumed in a dynamic fashion. The cloud delivery model provides this capability. The ability for developers to quickly attain broadly accessible, efficient, measured, and scalable service instances for composition into applications unlocks creativity and responsiveness to business needs.

3.5 Summary

The service provider is a multifaceted role that requires numerous technical disciplines to deliver useful services. Service providers are responsible for supplying a capability and delivering that capability in a way that is beneficial to the consumer.

The creation of a capability comes from industrious engineers and is guided by technical requirements from the consumer. A valuable delivery approach for that capability can be achieved by following the cloud computing model.

This chapter reviewed numerous aspects of how CICS on z/OS can be an ideal platform for both service creation and service delivery via the cloud model.

For more information about various features of the platform that should be considered for providing services, see Chapter 4, “The CICS systems programmer” on page 27 and Chapter 5, “The z/OS systems programmer” on page 45.

4

The CICS systems programmer

The CICS systems programmer has a broad range of responsibilities that require many diverse skill sets, including architecture, engineering, network communications, database access, and application development.

The traditional role of systems programmers over time became focused on system administration functions. But, to extend the capabilities of CICS, the *old-school*, multi-role technician must reemerge and embrace the latest technologies. Although activities, such as defining CICS resources, analyzing performance, installing new releases, and applying maintenance, are essential to establishing and maintaining a stable platform, CICS systems programmers must take a technical leadership role in the organization. This leadership is not only critical in platform and middleware architecture, but equally important, in assisting application programmers with designing and developing services in a cloud-enabled environment.

CICS services are an entry point into the environment and make available IBM z/OS resources, such as IBM DB2®, Information Management System (IMS™), Virtual Storage Access Method (VSAM), and other CICS managed resources to applications across all platforms within your enterprise, and to the Internet and mobile clients.

This chapter provides insight from the experiences of Walmart’s CICS systems programmers who also filled the roles of architect, systems engineer, and software developer, to deliver a cloud model in a z/OS IBM Parallel Sysplex using CICS.

This chapter includes the following topics:

4.1 CICS architecture

CICS adapted to the many different access channels that were introduced over the years: from 3270 and SNA to TCP/IP based protocols and APIs, such as IBM MQ, TCP/IP sockets, and HTTP. HTTP evolved from primarily providing HTML to delivering SOAP services that support SOA, to enabling ReST services, which widely support the cloud delivery model. Hosting the services in CICS within a z/OS Parallel Sysplex helps to enable the essential characteristics of the cloud delivery model, such as broad network access, resource pooling, metered service, and rapid elasticity.

The cloud services that were developed at Walmart were designed to be accessible from any platform or language via HTTP and ReST API calls. CICS resources for each user or service instance are created through provisioning services. The CICS resources that are required to support the services were identified by the service developer, while the automated provisioning services were created by the CICS systems programmer. In this case, the z/OS cloud services team at Walmart is composed of engineers that fill the roles of service developer and CICS systems programmer for the different services, which enables a DevOps environment.

The remaining sections of this chapter review the concepts and features that the CICS systems programmer uses to provide services that exhibit the essential characteristics of the cloud delivery model.

4.2 Service owning region

The concept of resource owning regions, such as terminal-owning regions (TOR), application-owning regions (AOR), file-owning regions (FOR), and queue-owning regions (QOR) provide high availability (HA) for CICS applications by using various input channels, such as 3270, SNA, and IBM MQ channels. More recently, CICS HA facilities were considerably improved through the provision of shared data facilities, such as VSAM record-level sharing (RLS), temporary storage (TS) server regions and coupling facility (CF) server regions. The result is that traditional FOR and QOR designs now have less relevance because it is possible to access shared data directly from AORs.

With services (whether SOAP, ReST, or cloud), two methods of resource owning configuration were considered.

The first method involves multiple web owning regions (WOR) and multiple AORs, where the WOR receives the HTTP request and a distributed program link (DPL) is issued to the AOR to run the program. Sysplex distributor and workload manager (WLM) provide HA to the WORs, and IBM CICSPlex® System Manager (CICSPlex SM) workload management provides HA to the AORs.

The second method includes multiple regions with combined WOR/AOR responsibility that Walmart refers to as service owning regions (SOR). Each SOR shares the IP endpoint, with TCP/IP port sharing and sysplex distributor providing HA with WLM to these regions.

The configuration that was selected to host the caching services and other services was the SOR model. This configuration reduced the number of CICS servers and the overhead of DPLs between WOR and AOR. It also reduced the number of calls to WLM.

An architectural view of the SOR configuration is shown in Figure 4-1. For more information about the supporting z/OS Parallel Sysplex configuration, see Chapter 5, “The z/OS systems programmer” on page 45.

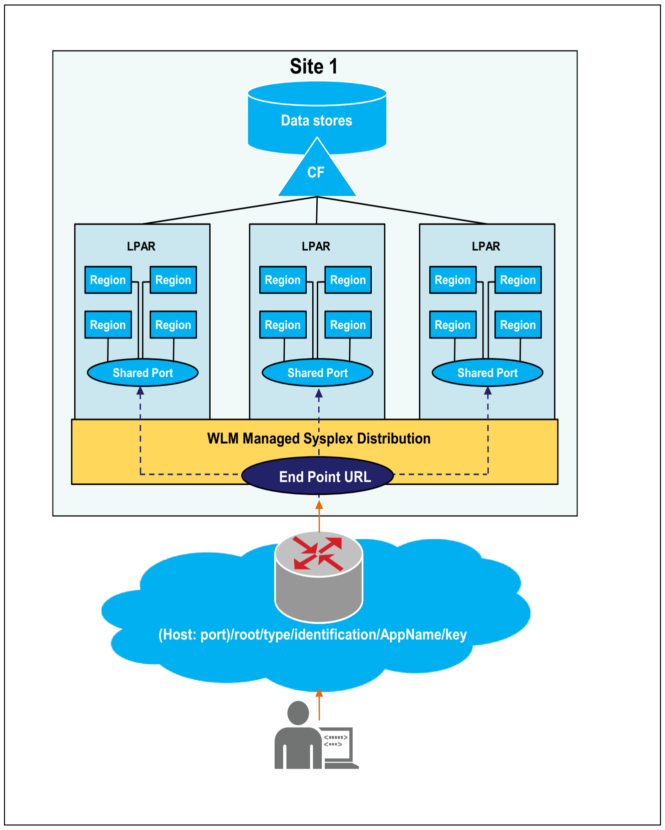

Site 1
Data stores
CF
LPAR
LPAR
LPAR
Region
Shared Port
WLM Managed Sysplex Distribution
End Point URL
(Host: port)/root/type/identification/AppName/key

Figure 4-1 SOR environment

4.3 CICS resource definitions

To provide HA for Walmart cloud services, multiple SORs were defined within a cluster by using a common shared port that was assigned to each server in the cluster across the sysplex. The CICS resource definitions that were created to host this environment are listed in Table 4-1 on page 30.

The only mandatory parameters are `TCPIP=YES` and a TCP/IP service; however, consideration should be given to defining all of the other resources that are listed in Table 4-1 on page 30 to provide for more security and control.

Table 4-1 CICS resource definitions for server provisioning

SIT parameter	Description
TCPIP=YES	Enable usage of TCP/IP services. This parameter is mandatory for any web access.
KEYRING	Name of IBM RACF® key ring for location of SSL certificates.
MAXOPENTCBS	Maximum number of open TCBs for threadsafe applications.
MAXSSLTCBS	Maximum number of TCBs for SSL and TLS handshake processing.
MINTLSLEVEL	Minimum level of TLS for use with TCP/IP services.
MXT	Maximum number of tasks per region.
SSLCACHE	Controls how SSL session IDs are cached, when partial SSL handshakes are used.
SSLDELAY	Performance optimization for SSL connection processing that determines the length of time for which CICS retains session IDs.
RDO definition	**Description**
TCPIPSERVICE	Definition of a TCP/IP service that listens for protocol-specific requests on a specific IP endpoint.

4.3.1 CICS resource definitions for services

Service resources are defined by automated processes when a consumer requests a service through a self-service provisioning portal that was developed by Walmart. The portal calls numerous CICS ReST services that drive the automation. The automation creates the resources by using standard EXEC CICS API and SPI commands and standard z/OS utilities, such as IDCAMS.

To share workloads, all logical components that comprise a cloud service must be defined in all CICS regions within a cluster. Whether through manual or automated processes, all resources for a service instance should be defined in their own (CSD) group, and then defined to the appropriate servers. Because the services are defined in individual groups, moving a service from one cluster to another (whether it is within the same LPAR, to a different LPAR, or even to a different sysplex) involves porting the group to the appropriate CSD for the new cluster and installing it in the servers.

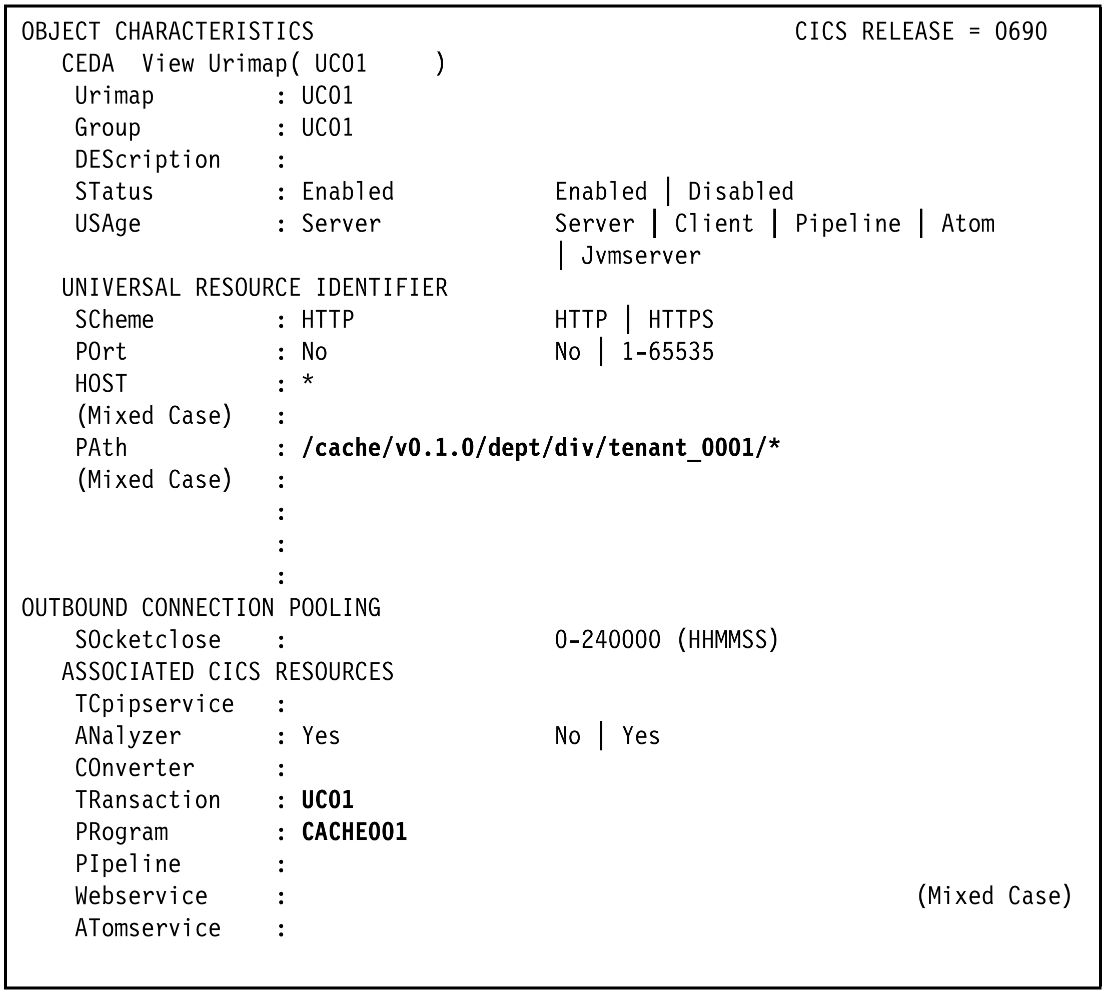
The minimum resources that are required to define a CICS cloud or ReST service are URIMAP, TRANSACTION, and TRANCLASS, as listed in Table 4-2.

Table 4-2 CICS resource definitions for cloud services

RDO definition	Description
URIMAP	Maps URIs to associated transaction and programs.
TRANSACTION	Defines the transaction attributes.
TRANCLASS	Transaction class defining the scheduling constraints for transaction execution.

URIMAP definitions

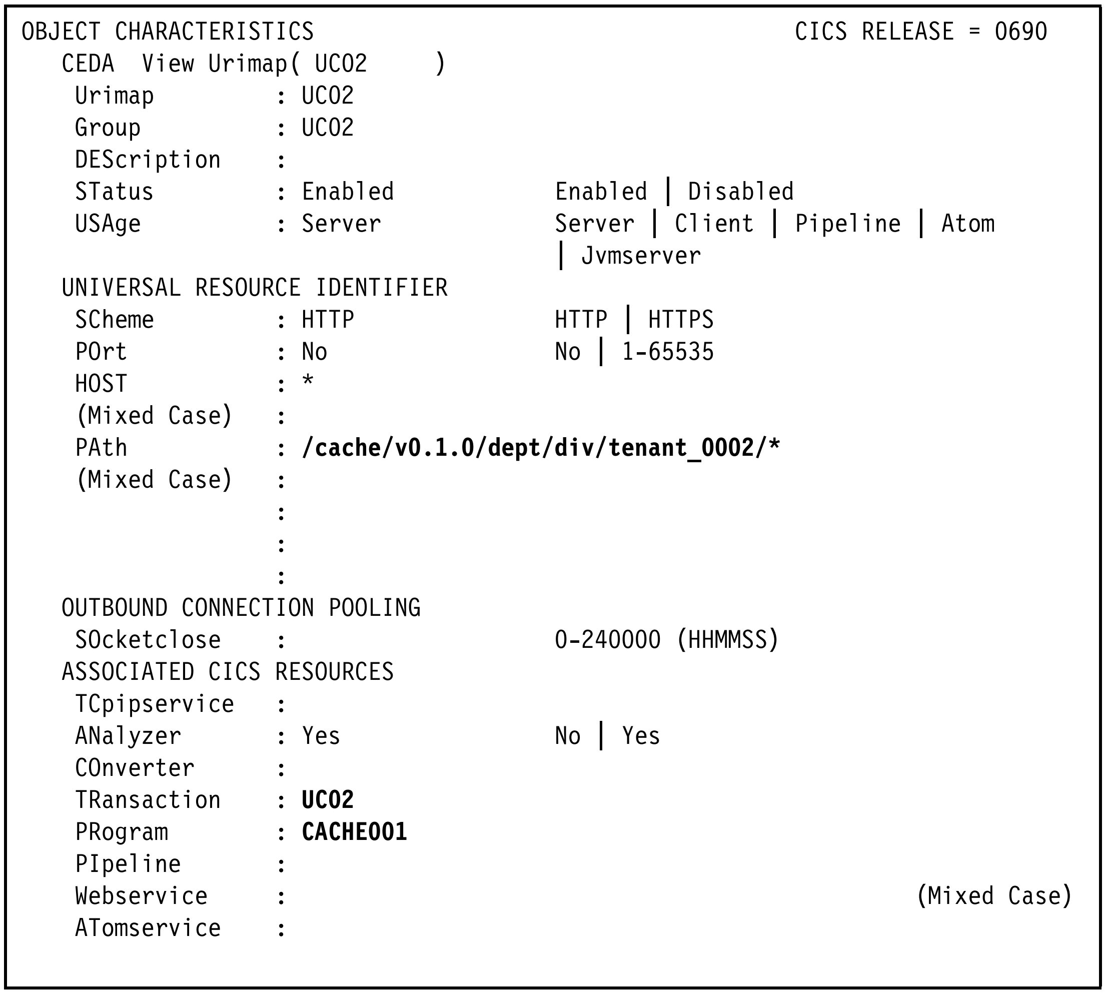
The URIMAP contains the path name of the URI and associates the CICS transaction and program that are to be run when the request is received by the TCP/IP service. The CEDA panel that is shown in Figure 4-2 shows the definition of the UC01 URIMAP definition, which is used to define the first instance of a service.



```
OBJECT CHARACTERISTICS                                    CICS RELEASE = 0690
   CEDA  View Urimap( UC01     )
    Urimap         : UC01
    Group          : UC01
    DEScription    :
    STatus         : Enabled            Enabled | Disabled
    USAge          : Server             Server | Client | Pipeline | Atom
                                        | Jvmserver
   UNIVERSAL RESOURCE IDENTIFIER
    SCheme         : HTTP               HTTP | HTTPS
    POrt           : No                 No | 1-65535
    HOST           : *
    (Mixed Case)   :
    PAth           : /cache/v0.1.0/dept/div/tenant_0001/*
    (Mixed Case)   :
                   :
                   :
                   :
OUTBOUND CONNECTION POOLING
    SOcketclose    :                    0-240000 (HHMMSS)
   ASSOCIATED CICS RESOURCES
    TCpipservice   :
    ANalyzer       : Yes                No | Yes
    COnverter      :
    TRansaction    : UC01
    PRogram        : CACHE001
    PIpeline       :
    Webservice     :                                          (Mixed Case)
    ATomservice    :
```

Figure 4-2 URIMAP for the first instance of a service

More URIMAP definitions are then used to identify further instances of each service with dedicated PATH and TRANSACTION attributes for each instance. Figure 4-3 shows the URIMAP definition that was used to define the second instance of the service.



```
OBJECT CHARACTERISTICS                                    CICS RELEASE = 0690
   CEDA  View Urimap( UC02     )
    Urimap         : UC02
    Group          : UC02
    DEScription    :
    STatus         : Enabled           Enabled | Disabled
    USAge          : Server            Server | Client | Pipeline | Atom
                                       | Jvmserver
   UNIVERSAL RESOURCE IDENTIFIER
    SCheme         : HTTP              HTTP | HTTPS
    POrt           : No                No | 1-65535
    HOST           : *
    (Mixed Case)   :
    PAth           : /cache/v0.1.0/dept/div/tenant_0002/*
    (Mixed Case)   :
                   :
                   :
                   :
   OUTBOUND CONNECTION POOLING
    SOcketclose    :                   0-240000 (HHMMSS)
   ASSOCIATED CICS RESOURCES
    TCpipservice   :
    ANalyzer       : Yes               No | Yes
    COnverter      :
    TRansaction    : UC02
    PRogram        : CACHE001
    PIpeline       :
    Webservice     :                                           (Mixed Case)
    ATomservice    :
```

Figure 4-3   URIMAP for the second instance of a service

Transaction definitions

The path name and transaction ID attributes in a URIMAP resource can be used to identify each instance of a specific service while running a common program. Identifying each instance of a service enables the metered and measured aspect of the cloud delivery model, and provides multitenancy within CICS servers.

Figure 4-4 shows the UC01 TRANSACTION definition for the first instance of a service. All service requests that use the UC01 URIMAP definition run under this transaction ID and are subjected to the scheduling constraints as defined in the TCLUC01 transaction class.

```
OBJECT CHARACTERISTICS                                    CICS RELEASE = 0690
   CEDA  View TRANSaction( UC01 )
    TRANSaction    : UC01
    Group          : UC01
    DEScription    :
    PROGram        : CACHE001
    TWasize        : 00000              0-32767
    PROFile        : DFHCICST
    PArtitionset   :
    STAtus         : Enabled            Enabled | Disabled
    PRIMedsize     : 00000              0-65520
    TASKDATALoc    : Any                Below | Any
    TASKDATAKey    : User               User | Cics
    STOrageclear   : No                 No | Yes
    RUnaway        : System             System | 0 | 500-2700000
    SHutdown       : Disabled           Disabled | Enabled
    ISolate        : Yes                Yes | No
    Brexit         :
   REMOTE ATTRIBUTES
    DYnamic        : No                 No | Yes
    ROutable       : No                 No | Yes
    REMOTESystem   :
    REMOTEName     :
    TRProf         :
    Localq         :                    No | Yes
   SCHEDULING
    PRIOrity       : 001                0-255
    TClass         : No                 No | 1-10
    TRANClass      : TCLUC01
```

Figure 4-4 TRANSACTION definition for the first instance of a service

More transaction definitions are then used to identify further instances of each service that references a dedicated transaction class for the service. For example, transaction UC02 defines the TCLASS attribute as TCLUC02, as shown in Figure 4-5 on page 34.

```
OBJECT CHARACTERISTICS                                        CICS RELEASE = 0690
  CEDA  View TRANSaction( UC02 )
   TRANSaction    : UC02
   Group          : UC02
   DEScription    :
   PROGram        : CACHE001
   TWasize        : 00000            0-32767
   PROFile        : DFHCICST
   PArtitionset   :
   STAtus         : Enabled          Enabled | Disabled
   PRIMedsize     : 00000            0-65520
   TASKDATALoc    : Any              Below | Any
   TASKDATAKey    : User             User | Cics
   STOrageclear   : No               No | Yes
   RUnaway        : System           System | 0 | 500-2700000
   SHutdown       : Disabled         Disabled | Enabled
   ISolate        : Yes              Yes | No
   Brexit         :
  REMOTE ATTRIBUTES
   DYnamic        : No               No | Yes
   ROutable       : No               No | Yes
   REMOTESystem   :
   REMOTEName     :
   TRProf         :
   Localq         :                  No | Yes
  SCHEDULING
   PRIOrity       : 001              0-255
   TClass         : No               No | 1-10
   TRANClass      : TCLUC02
```

Figure 4-5 TRANSACTION definition or the second instance of a service

Whether cloud or enterprise, CICS services enable applications across various channels access to the wide array of z/OS resources, such as DB2, IMS, VSAM, and all CICS managed resources. CICS applications and resources then become available to Internet, mobile, non-z/OS and z/OS consumers via standard HTTP requests.

4.4 Security

Security is a critical consideration when z/OS resources are made available through CICS services because the service is an entry point into the environment. The scope of the network to which CICS services are made available has a significant bearing on how security (specifically, authentication) is implemented. Basic authentication might be sufficient when access to CICS services are protected by a DMZ (a perimeter network); however, extra security layers, such as IBM DataPower® Gateway Appliance, should be considered if the services are made available directly beyond the DMZ.

CICS supports HTTP and provides encryption at the transport layer (SSL/TLS) when requests are received on a port that is defined as HTTPS. Basic Authentication, when specified for a service instance or at the port level, is provided by calls to RACF. The user ID presented for basic authentication in the HTTP header can also assist with tracking resource usage for the measured service characteristics. For example, the `MNR_ID_USERID` field in the CICS transaction monitoring records can be used to identify resource usage by a user.

4.4.1 SSL certificates and TCP/IP services

Although there are several methods for defining and assigning SSL certificates, the technique Walmart implemented uses a wildcard certificate that is defined to the ports in the CICS servers that are hosting their cloud services.

During the processing of the SSL handshake, the server verifies the common name in the Subject distinguished name (DN) from the server's certificate with the host name of the server. If the common name and the host name do not match, the SSL connection is dropped. This issue is problematic in scenarios where a single certificate must be shared across multiple host names, such as used in DNS aliasing.

SSL wildcard certificates offer a solution to this issue by securing access to multiple subdomains underneath a single common subdomain by adding an asterisk (*) in the subdomain area to the left of the common subdomain name. For example, creating a DNS alias that uses a naming convention in which each consumer is assigned a unique subdomain name, such as `user####`, and a common higher-level subdomain that represents the service type, such as cache, a single wildcard certificate for `*.cache.hostname.com` can secure all cache consumers, regardless of which DNS alias they originally referenced.

This technique of wildcard certificates was used to secure the SSL TCP/IP services, as shown in Figure 4-6 on page 36 and Figure 4-7 on page 37 by using the `CERTIFICATE` attribute to refer to a certificate label in the SSL key ring that references a wildcard certificate for the relevant subdomain.

```
OBJECT CHARACTERISTICS                                    CICS RELEASE = 0690
   CEDA  View TCpipservice( UC500001 )
    TCpipservice   : UC500001
    GROup          : TCPUC
    DEScription    : SHARED PORT FOR CACHE GROUP 01
    Urm            : DFHWBADX
    POrtnumber     : 50001             1-65535
    STatus         : Open              Open | Closed
    PROtocol       : Http              Http | Eci | User | IPic
    TRansaction    : CWXN
    Backlog        : 00050             0-32767
    TSqprefix      :
    Host            : ANY
    (Mixed Case)   :
    Ipaddress      : ANY
    SPeciftcps     :
    SOcketclose    : 000005            No | 0-240000 (HHMMSS)
    MAXPersist     : No                No | 0-65535
    MAXDatalen     : 032760            3-524288
   SECURITY
    SSl            : Yes               Yes | No | Clientauth
    CErtificate    : cache.hostname.com
    (Mixed Case)
    PRIvacy        : Supported         Notsupported | Required | Supported
    CIphers        : 35363738392F303132330A1613100D0915120F0C
    (Mixed Case)
    AUthenticate   : No                No | Basic | Certificate | AUTORegister
                                       | AUTOMatic
```

Figure 4-6 TCPIPSERVICE for the cache services with wildcard certificate

```
OBJECT CHARACTERISTICS                                      CICS RELEASE = 0690
   CEDA  View TCpipservice( UD500002 )
    TCpipservice   : UD500002
    GROup          : TCPUD
    DEScription    : SHARED PORT FOR KEY/VALUE SERVICE GROUP 01
    Urm            : DFHWBADX
    POrtnumber     : 50002             1-65535
    STatus         : Open              Open | Closed
    PROtocol       : Http              Http | Eci | User | IPic
    TRansaction    : CWXN
    Backlog        : 00050             0-32767
    TSqprefix      :
    Host           : ANY
    (Mixed Case)   :
    Ipaddress      : ANY
    SPeciftcps     :
    SOcketclose    : 000005            No | 0-240000 (HHMMSS)
    MAXPersist     : No                No | 0-65535
    MAXDatalen     : 032760            3-524288
   SECURITY
    SSl            : Yes               Yes | No | Clientauth
    CErtificate    : kvdb.hostname.com
    (Mixed Case)
    PRIvacy        : Supported         Notsupported | Required | Supported
    CIphers        : 35363738392F303132330A1613100D0915120F0C
    (Mixed Case)
    AUthenticate   : No                No | Basic | Certificate | AUTORegister
                                       | AUTOMatic
```

Figure 4-7 TCPIPSERVICE for the key-value database services with wildcard certificate

4.4.2 Wildcard certificates and host names

Walmart's provisioning automation assigns the DNS alias and URL for each service by using the following naming convention:

```
ConsumerID.ServiceTypeSubdomain.HostSubdomain.TopLevelDomain/path
```

This naming convention ensures that all instances of each service are given a unique consumer ID for each service type. For example, the first three cache instances that are provisioned would be assigned the following URLs:

- `user0001.cache.hostname.com/cache/v0.1.0/dept/div/tenant_0001/`
- `user0002.cache.hostname.com/cache/v0.1.0/dept/div/tenant_0002/`
- `user0003.cache.hostname.com/cache/v0.1.0/dept/div/tenant_0003/`

In this case, all consumers of the cache service can be secured with a `*.cache.hostname.com` wildcard certificate.

Another z/OS cloud service that was created by Walmart is a key-value database service, which is assigned `kvdb` as the service type. For this service type, the first three instances that are provisioned would be assigned the following URLs:

- user0001.kvdb.hostname.com/kvdb/v0.1.0/dept/div/tenant_0001/
- user0002.kvdb.hostname.com/kvdb/v0.1.0/dept/div/tenant_0002/
- user0003.kvdb.hostname.com/kvdb/v0.1.0/dept/div/tenant_0003/

In this case, all consumers of the `kvdb` service can be secured with a `*.kvdb.hostname.com` wildcard certificate.

For more information about how Walmart's provisioning automation scheme assigns the DNS aliases and URLs for each server, see 4.6, "Network access" on page 42.

4.4.3 Basic authentication

Basic authentication is one of the simplest techniques for security access controls to web resources. It is based on the use of user ID and password fields in the HTTP authorization header, which are base64 encoded. It supports machine-to-machine communication where credentials are pre-supplied on each request and user interaction by using a simple challenge dialog in a web browser to prompt for input credentials, as shown in Figure 4-8.

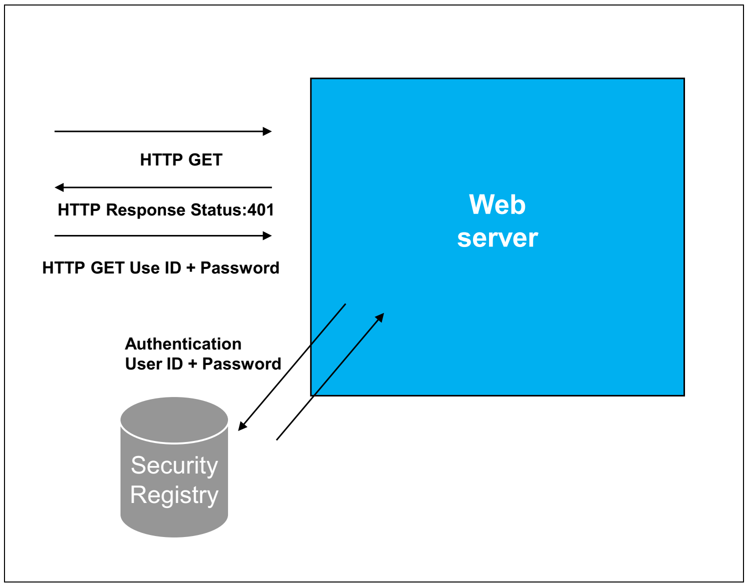


Figure 4-8 Basic authentication

4.4.4 Authentication at the server level

When a port is defined to CICS in a `TCPIPSERVICE` resource with the `AUTHENTICATE` attribute set to `BASIC` all services are required to provide user ID and password credentials, which are validated against RACF; otherwise, the request is rejected with an HTTP status code 401.

Service consumers in Walmart can request surrogate *service IDs* that include randomly generated passwords by using another service that is offered through the provisioning portal. The consumer can also choose to enable authentication and network encryption (HTTPS) as options when requesting a service instance through the provisioning portal.

When the TCPIPSERVICE resource is defined with authentication disabled on an HTTPS port, transport level encryption is still provided and requests are allowed to enter through the port with or without RACF credentials.

4.4.5 Authentication at the service level

Basic authentication at the port level requires all requests entering through that port to provide credentials. However, this method lacks the required flexibility because the consumers require the ability to determine whether authentication is required for each service instance.

To provide flexibility and reduce the number of TCPIPSERVICE definitions, the Walmart cloud service disables basic authentication at the port level and enables security at the service level for all service types. To perform this function, a custom security program was developed by Walmart to extract the basic authentication credentials from the HTTP header, perform Base64 decoding, and call RACF to authenticate the credentials by using the EXEC CICS VERIFY command.

When a consumer requires access to the cloud service to be authenticated, the URIMAP CONVERTER field for the specific service instance is defined with this custom basic authentication program name BASEAUTH (see Figure 4-9 on page 40). When the credentials are authenticated successfully by RACF, control is given to the cloud service program; otherwise, an exception program is invoked that returns an HTTP status code 401.

```
OVERTYPE TO MODIFY                                       CICS RELEASE = 0690
   CEDA  ALter Urimap( UC01     )
    Urimap        : UC01
    Group         : UC01
    DEScription  ==>
    STatus       ==> Enabled           Enabled | Disabled
    USAge        ==> Server            Server | Client | Pipeline | Atom
                                       | Jvmserver
   UNIVERSAL RESOURCE IDENTIFIER
    SCheme       ==> HTTP              HTTP | HTTPS
    POrt         ==> No                No | 1-65535
    HOST         ==> *
    (Mixed Case) ==>
    PAth         ==> /cache/v0.1.0/dept/div/tenant_0001/*
    (Mixed Case) ==>
                 ==>
                 ==>
                 ==>
   OUTBOUND CONNECTION POOLING
    SOcketclose  ==>                   0-240000 (HHMMSS)
   ASSOCIATED CICS RESOURCES
    TCpipservice ==>
    ANalyzer     ==> Yes               No | Yes
    COnverter    ==> BASEAUTH
    TRansaction  ==> UC01
    PRogram      ==> CACHE001
    PIpeline     ==>
    Webservice   ==>                                           (Mixed Case)
    ATomservice  ==>
   SECURITY ATTRIBUTES
    USErid       ==>
    CIphers      ==>
                                                               (Mixed Case)
    CErtificate  ==>                                           (Mixed Case)
    AUthenticate ==>                   No | Basic
```

Figure 4-9 URIMAP with BASEAUTH converter program defined

Another level of security is provided by Walmart's cloud services. When a service instance is provisioned for a consumer, a custom security definition resource is created in an external data store with a list of user IDs that can access this instance and the HTTP method level access authority for each user ID. After the custom basic authentication program completes, the cloud service program receives control and compares the RACF user ID and method from the HTTP header with entries in the custom security resource table. If the user ID and method combination are not allowed, the service returns an HTTP status code 401, as shown in Figure 4-10 on page 41.

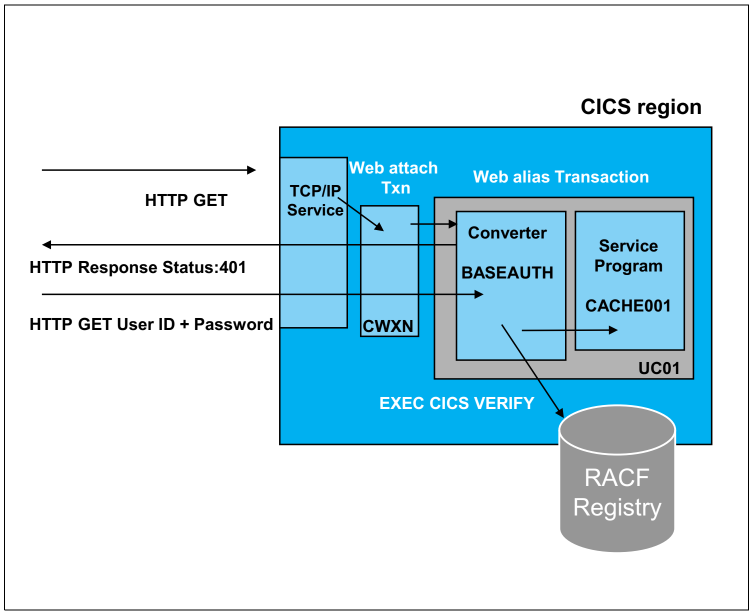


Figure 4-10 Custom basic authentication converter

4.5 Multitenancy

The ability to host multiple instances of the same type of service creates a multitenancy environment that supports the cloud delivery model. An important and highly recommended approach is to establish naming conventions for all resources that form a cloud service, regions that host the services, and network DNS alias names. Developing a naming convention for services and related resources enables each instance of a service to be identifiable by a common identifier. For example, provisioning the first instance of a service that requires access to a VSAM file generates a group with a transaction, transaction class, file, and URI map, and other required resources, as shown in Figure 4-11.

```
EX G(UC01)
 ENTER COMMANDS
  NAME     TYPE         GROUP                                       LAST CHANGE
  UC01VSAM FILE         UC01                                     01/15/16 13:16:05
  TCLUC01  TRANCLASS    UC01                                     01/15/16 13:16:23
  UC01     TRANSACTION  UC01                                     01/15/16 13:16:38
  UC01GENQ ENQMODEL     UC01                                     01/15/16 13:16:50
  UC01     TDQUEUE      UC01                                     01/15/16 13:17:05
  UC01DOCT DOCTEMPLATE  UC01                                     01/15/16 13:17:20
  UC01     URIMAP       UC01                                     01/15/16 13:17:38
```

Figure 4-11 Logical components for the first instance of a service

Tenant isolation is provided within the cluster through adherence to a naming convention where the group name is part of the resource name. More instances of the service can be defined to the cluster by using a different identifier. For example, provisioning the second instance of the same service resources generates a group that uses the same naming convention but with a different unique suffix as shown in Figure 4-12 on page 42.

```
EX G(UC02)
  ENTER COMMANDS
   NAME     TYPE         GROUP                                  LAST CHANGE
   UC02VSAM FILE         UC02                                01/15/16 13:19:55
   TCLUC02  TRANCLASS    UC02                                01/15/16 13:20:06
   UC02     TRANSACTION  UC02                                01/15/16 13:20:25
   UC02GENQ ENQMODEL     UC02                                01/15/16 13:20:38
   UC02     TDQUEUE      UC02                                01/15/16 13:20:51
   UC02DOCT DOCTEMPLATE  UC02                                01/15/16 13:21:04
   UC02     URIMAP       UC02                                01/15/16 13:21:20
```

Figure 4-12 Logical components for the second instance of a service

4.6 Network access

The DNS alias provides an abstraction that allows the target host name or system to be changed and workloads routed to different CICS regions loaded in alternate LPARs or sysplexes without the need for clients (consumers) to change their URL. This technique allows workload to be moved for various reasons, such as load balancing, service isolation, and disaster recovery.

Table 4-3 shows an example of a DNS alias scheme and the corresponding host names that resolve to the dynamic virtual IP address (DVIPA) endpoints in a z/OS Parallel Sysplex. The accompanying port number on the z/OS IP endpoint provides the entry point into the CICS region cluster, as defined in the TCP/IP service, and is mapped by a network load balancer.

Table 4-3 Table 1.DNS alias and mapping to z/OS TCP/IP host names

Service type	**DNS alias**	**z/OS TCP/IP host name: port**
Cache	user0001.cache.hostname.com	zos.sysplex01.com:50001
	user0002.cache.hostname.com	zos.sysplex01.com:50001
Key-value database	user0001.kvdb.hostname.com	zos.sysplex01.com:50002
	user0002.kvdb.hostname.com	zos.sysplex01.com:50002

This table shows how each service type maps to a dedicated port serviced by a cluster of CICS regions with a specific TCP/IP service. Then, how each consumer instance (such as user0001) is directed to the same TCP/IP service but isolated through use of dedicated CICS resources as described in 4.5, “Multitenancy” on page 41.

End-to-end service request resolution

Figure 4-13 shows the resolution of a request for a cache service from the service consumer through the network into the CICS TCP/IP service and the specified CICS URIMAP and transactions.

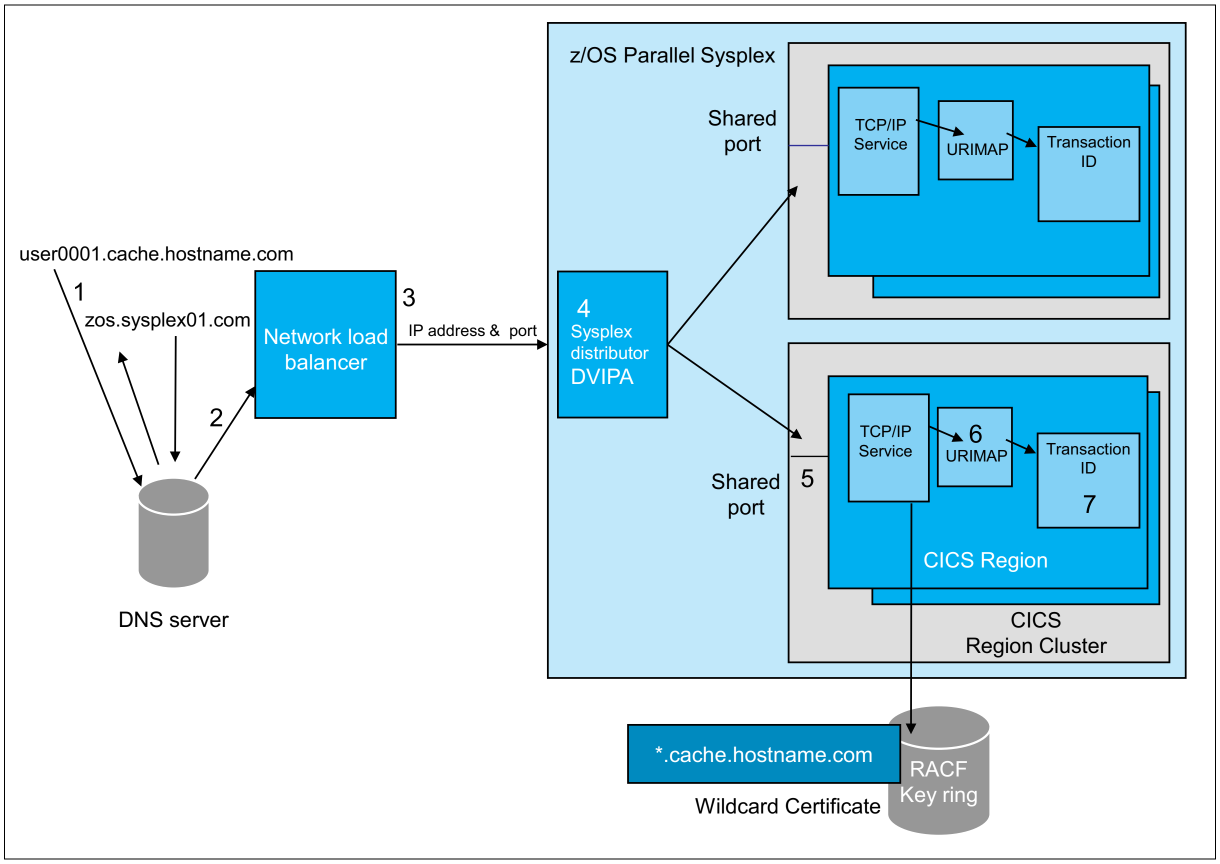


Figure 4-13 Resolution of request flow for a cache service

The flow of control at each point is summarized in the following process:

1. The DNS alias in a CNAME record maps the DNS alias to z/OS host name.
2. z/OS host name resolves to a sysplex distributor DVIPA address.
3. Network load balancer directs the connection to the z/OS port for the specific service type.
4. A shared DVIPA and port is assigned to the cluster of CICS regions across multiple LPARs.
5. CICS TCP/IP services defines the port and wildcard certificate for the specified service type.
6. CICS URIMAP defines the URL path for each instance of a service and assigns the unique transaction ID and the common service program.
7. CICS TRANSACTION is defined for each unique instance of a service and is assigned a dedicated TRANCLASS to control metered use and protect the server from excessive transaction volume.

4.7 Summary

The CICS systems programmer role must continue to evolve to provide a robust platform for modern applications.

The CICS systems programmers should keep current their knowledge of z/OS resources, database capabilities, programming languages, network communications, and the CICS platform.

Building relationships and communications with service consumers, service providers and z/OS support teams is essential in understanding how to design and build the appropriate landscape for enterprise and cloud services.

5 The z/OS systems programmer

The IBM z/OS systems programmer often is the heart of the shop. Historically, the z/OS system programmers held responsibility for managing the z Systems hardware configuration and installing, customizing, and maintaining the z/OS operating system. It is their responsibility to ensure that the z Systems services are available and operating according to service level agreements. In some organizations, the z/OS system programmer might also have responsibility for capacity planning.

This system expertise and view of the greater picture makes a z/OS system programmer a foundational resource in the multi-person role of service provider. As a part of the service provider's role, the modern z/OS system programmers must branch outside of their typical responsibilities to team with other supportive roles in their organization to understand the needs of the service consumers. They must be willing to look at the needs of the service consumer objectively and creatively to find a way to provide the services they require and support them based on their traditional expertise. This challenge can be exciting for the z/OS system programmer and provide resource savings and increased operational efficiencies.

This chapter provides a look into Walmart's environment and how they used key pieces to create z/OS cloud service. It also describes the role that each part of the system plays in providing a key resource and how the components can be used for the cloud service.

This chapter includes the following topics:

5.1 Sysplex architecture

A key enabler of the cloud delivery model for z/OS is Parallel Sysplex. Although it is not required, Parallel Sysplex greatly facilitates the characteristics that are needed for cloud services.

5.1.1 Resource pooling, rapid elasticity, and measured service

Two of the essential characteristics of cloud, resource pooling, and rapid elasticity can be achieved primarily through configuring the z/OS Parallel Sysplex environment with little need for extra components to be developed or installed.

Resource pooling is an inherent characteristic of the platform since it was created. Because of the virtualized nature of the system, nearly any processing on z/OS is performed atop resource pools. Examples of these pooled resources include logical compute cycles, virtual storage (memory), and Data Facility Storage Management Subsystem (DFSMS) storage groups. Even beyond the resource pooling that is provided as part of the platform, other forms of resource pooling can be configured and used for service delivery; for example, high availability (HA) CICS regions can be considered a pool.

Elasticity can be relatively easily achieved in z/OS by over-provisioning the virtual runtime environments that host the services. Then, rely on the platform capabilities, such as Workload Manager (WLM), to ensure that those address spaces receive enough resources to satisfy demand, and then, reassign those resources elsewhere as needed. For more information about the more relevant WLM concepts, see 5.2, “Workload manager” on page 47.

The *measured service* characteristic is also largely enabled by configuring the environment, but likely needs some investment to fully realize. Configuring System Management Facilities (SMF) properly enables capturing most of the necessary data. However, this data is not immediately consumable. More tools or products are needed to make the information available to the consumers, providers, and other operational roles. For more information about this concept, see Chapter 6, “Operational considerations” on page 59 and the IBM Redbooks publication, *Creating IBM z/OS Cloud Services*, SG24-8324.

5.1.2 Unique characteristics

The existence of shared resources across disparate systems is a unique characteristic of the z/OS Parallel Sysplex architecture. By using concepts, such as distributed dynamic virtual IP addressing (DVIPA) to simplify endpoint management or data sharing to reduce data replication requirements provides particular value in a service-oriented delivery model. These attributes must be considered when determining the types of services that are a good fit for the platform.

Walmart used these and other characteristics of the platform to provide a differentiated experience to consumers with their caching service.

5.1.3 Important pieces

Numerous pieces are required to set up a z/OS Parallel Sysplex and volumes of documentation are available on the subject. The complete setup of a parallel sysplex is beyond the scope of this publication and is not described herein. This material is built on the assumption that a base parallel sysplex is defined, and the remainder of this chapter focuses on particular pieces of the z/OS and parallel sysplex configuration that are more specifically relevant to cloud service delivery. Many of these settings are directly related, and complementary to the items that are described in Chapter 4, “The CICS systems programmer” on page 27.

5.2 Workload manager

The WLM component of the z Systems platform plays an important role in supporting the pooled resources and rapid elasticity characteristics of the cloud delivery model. These characteristics rely on automated assignment and reassignment of resources within predefined rules. WLM provides this function through policy-based resource allocations across different workload categorizations and groups.

There are various WLM-related features that can be used to ensure a dynamic hosting environment. In Walmart’s case, the following particular features were used:

- LPAR weighting
- Service class
- WLM ASID weights

5.2.1 LPAR weighting

Walmart uses logical partition (LPAR) weighting at the Processor Resource/System Manager (PR/SM) level. This feature is used to ensure that a particular amount of capacity is available to each system that is hosting services, while also allowing these systems to make use of unused capacity on the central processor complex (CPC), if needed.

5.2.2 Service class

The new cloud services (and web services in general) are regarded much like traditional online workloads, so WLM service classes were used as models for management of newer service-oriented workloads. In fact, Walmart adopted SOA-style web services many years earlier, and the same WLM service classes were used to manage the regions that host cloud services.

The address space identifiers (ASIDs) that are associated with the regions are included in service classes that carry a very high importance (1) and higher than average velocity (60) with NORMAL I/O priority, as shown in Figure 5-1.

```
Base goal:
CPU Critical - NO    I/O Priority Group- NORMAL

    # Duration   Imp Goal description
    - --------   -   ----------------------------------------------------
    1            1   Execution velocity of 60
```

Figure 5-1 WLM service class for cloud service workloads

The ASID service class is used by the z/OS Workload Manager to ensure that performance goals (which are specified in business terms) are met. The activity that is associated with meeting that performance goal is the responsibility of the operating system. The actions that the system takes are related to the allocation of CPU and storage.

5.2.3 WLM ASID weights

WLM also maintains an awareness of how well each ASID within a service class is meeting its goals. This information is used through WLM managed sysplex distribution to direct new work to the regions that are meeting or exceeding their associated goals, and are most suited to accommodate new requests. For more information about the use of this capability, see 5.3, "TCP/IP" on page 48.

5.3 TCP/IP

Chapter 4, "The CICS systems programmer" on page 27 describes many aspects of this cloud service delivery implementation from the CICS perspective that rely upon the configuration of TCP/IP that is provided by the z/OS systems programmer. In particular, configuring sysplex distributed DVIPA to be managed by WLM is needed for this design.

There are three general tasks that comprise this setup: defining the TCP/IP configuration to enable sysplex distribution, establishing shared ports, and assigning the DVIPA with ASID-level WLM management. Each of these tasks, as described in the following sections, is configured in an installation's TCP/IP profile data set.

5.3.1 Enabling sysplex distribution

Two statements are required in the TCP/IP profile IPCONFIG (or IPCONFIG6) to initially enable this functionality: DYNAMICXCF and SYSPLEXROUTING, as shown in Figure 5-2.

```
IPCONFIG
      DYNAMICXCF 123.45.67.89 255.255.255.224 1
      SYSPLEXROUTING
```

Figure 5-2 TCP/IP profile configuration for sysplex distribution

The DYNAMICXCF statement establishes a single address by which all stacks in a sysplex can communicate with the other stacks. It includes the IP address, a subnet mask, and a cost metric identifier (unless OMPROUTE is used; then, it is overridden).

The SYSPLEXROUTING statement establishes the TCP/IP stack as part of a sysplex and enables the use of WLM consultation for distributing requests to the stacks in the sysplex.

5.3.2 Establish shared ports

To set up shared ports, assign the same port number to each region in a cluster or group of regions and identify the port as shared. The port assignment is configured in the TCP/IP profile, as shown in Figure 5-3 on page 49.

```
PORT 12345   TCP  CICSRGI    NOAUTOLDELAYA SHAREP
PORT 12345   TCP  CICSRG2    NOAUTOLDELAYA SHAREP
PORT 12345   TCP  CICSRG3    NOAUTOLDELAYA SHAREP
PORT 12345   TCP  CICSRG4    NOAUTOLDELAYA SHAREP
PORT 12345   TCP  CICSRGS    NOAUTOLDELAYA SHAREP
PORT 12345   TCP  CICSRG6    NOAUTOLDELAYA SHAREP
```

Figure 5-3 PORT statements in TCP/IP profile

Figure 5-3 shows the explicit assignment of a single port number to multiple regions. To reduce ongoing maintenance, the region name can be wild-carded so that any other regions that are created under the same name pattern acquire the port assignment without further updates to the TCP/IP profile data set, as shown in the following example:

```
PORT 12345  TCP  CICSRG*  NOAUTOL DELAYA  SHAREP
```

The NOAUTOL, DELAYA, and SHAREP parameters of the PORT statement are used to specify the following parameters:

- NOAUTOL(OG): Prevent the ASID from being restarted after it is stopped.
- DELAYA(CKS): Explicit assignment of default setting that delays transmission of acknowledgements of packets that were received with the PUSH bit on in the TCP header.
- SHAREP(ORT): Required to share port across multiple listeners.

5.3.3 Assign DVIPA

The DVIPA can then be set up for an address by using the shared port that the groups of regions listen on, as shown in Figure 5-4.

```
VIPADYNAMIC
VIPADIST DISTM SERVERNLM 123.45.67.89
  PORT  12345 XXXXX XXXXX XXXXX XXXXX
  DESTIP ALL
ENDVIPADYNAMIC
```

Figure 5-4 VIPADYNAMIC-VIPADISTRIBUTE statements in TCP/IP profile

This configuration statement includes the following components:

- VIPADYNAMIC VIPADIST(RIBUTE) (DEFINE implicit): Enables the sysplex distributor function for DVIPA.
- DISTM(ETHOD) SERVERWLM: Specifies that server-specific values should be collected for this group of DVIPA ports. This specification allows WLM to assign weights to individual ASIDs for request distribution decisions.
- DESTIP ALL: Specifies that all TCP/IP stacks in the sysplex are targets for the DVIPA/Ports in this statement.

The combination of shared ports and WLM-managed sysplex distribution ensures that requests are routed to the most appropriate system and region in the sysplex to handle that request at any specific time. The design of this type of environment is shown in Figure 5-5 on page 50.

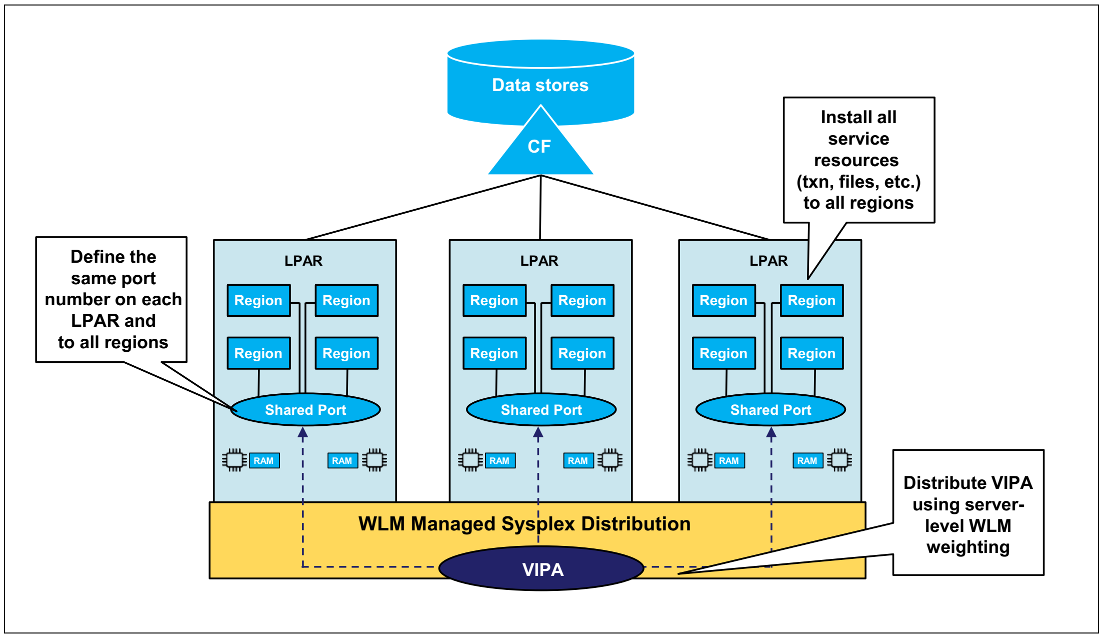

Data stores
CF
Install all service resources (txn, files, etc.) to all regions
Define the same port number on each LPAR and to all regions
LPAR
Region
Shared Port
RAM
WLM Managed Sysplex Distribution
VIPA
Distribute VIPA using server-level WLM weighting

Figure 5-5 WLM managed sysplex distribution

5.3.4 Domain Name System

The responsibility for Domain Name System (DNS) management often is outside of the z/OS-related roles. However, the z/OS systems programmer and the CICS systems programmer often must interact with and use the services of the DNS administrators in their organization.

A DNS name makes IP addresses more logical and consumable. To establish an accessible entry point for the defined VIPA, the z/OS systems programmer should request a DNS name to be associated with it. This association is accomplished with a DNS A record.

Walmart employs a model in which a DNS CNAME alias is also defined and associated with the DNS A record that was assigned to the VIPA. Network load balancers translate a request for the alias and forward the request to the VIPA DNS and port number of the service-hosting region cluster. This design provides several benefits through abstraction and manageability. The inclusion of the DNS constructs to the TCP/IP configuration is shown in Figure 5-6 on page 51.

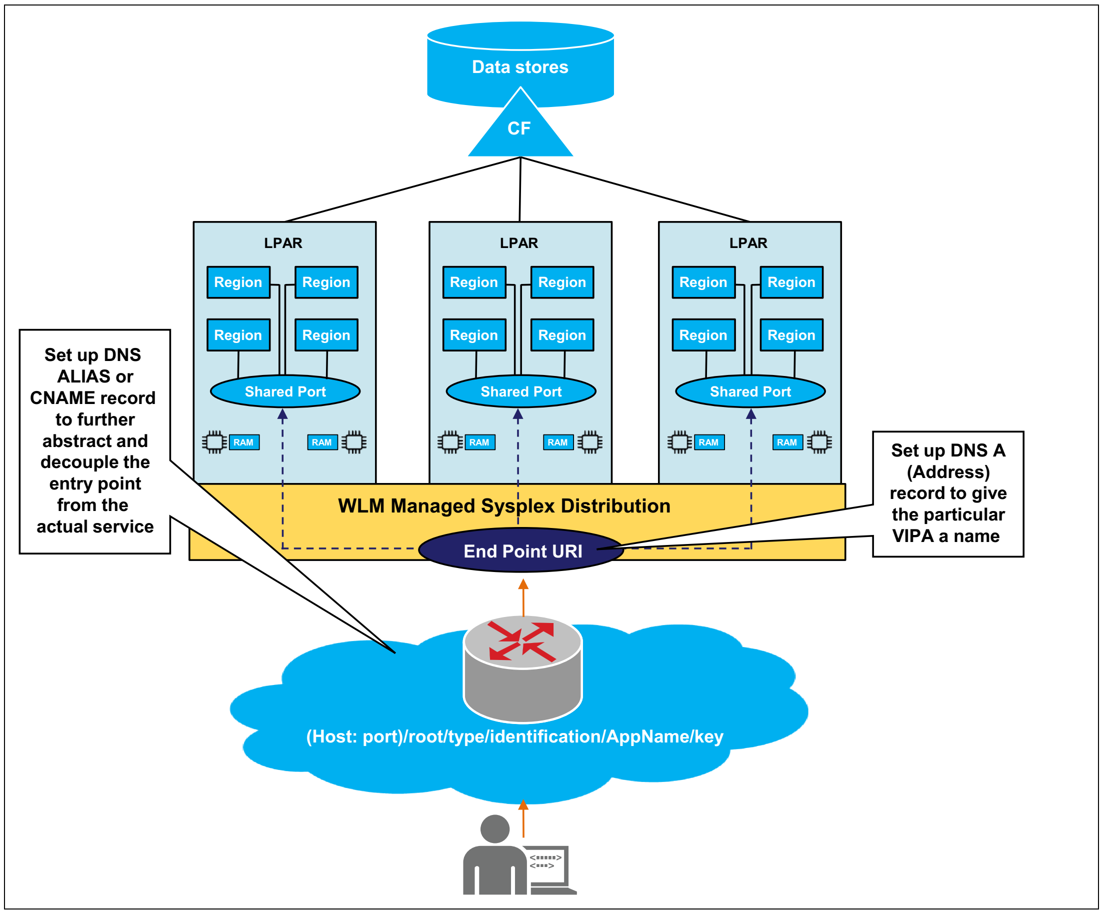

Data stores
CF
LPAR
Region
Shared Port
RAM
LPAR
Region
Shared Port
RAM
LPAR
Region
Shared Port
RAM
Set up DNS ALIAS or CNAME record to further abstract and decouple the entry point from the actual service
WLM Managed Sysplex Distribution
End Point URI
Set up DNS A (Address) record to give the particular VIPA a name
(Host: port)/root/type/identification/AppName/key

Figure 5-6 DNS assignments provide abstraction

For more information about the various ways that value is derived from this model, see Chapter 4, "The CICS systems programmer" on page 27. A particular aspect of this design that remains relevant to the z/OS systems programmer is its role in ensuring availability of service. For more information, see 5.4, "High availability".

5.4 High availability

Along with enabling the resource pooling and rapid elasticity characteristics, the design that is described in section 5.3, "TCP/IP" on page 48 and shown in Figure 5-5 on page 50 and Figure 5-6 on page 51 contributes substantially to high availability (HA).

Distributing requests across multiple systems and multiple regions per system provides resiliency. A single region, multiple regions, or even an entire LPAR or system can be removed (in a planned or unplanned manner), and service requests can still be satisfied. The workload can continue to be distributed to the remaining systems and regions, as shown in Figure 5-7 on page 52.

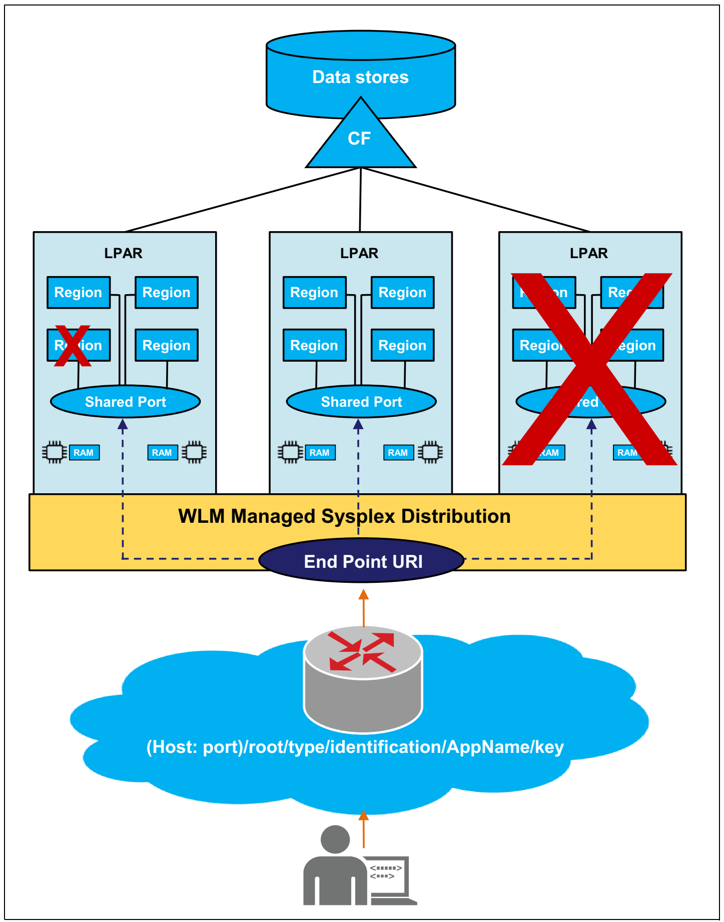


Figure 5-7 Service availability if there are outages

In a situation where multiple regions or an entire system is removed from service, performance degradation can occur because of limited resources. These types of scenarios should be considered when planning the initial design of the hosting environment. Sufficient resources should be defined to handle expected workloads if a portion of the environment is unavailable.

Walmart elected to provide tiers of availability assurances that were related to their services. The basic availability configuration is sufficient for many consumers, but some consumers wanted greater availability assurances. As a result of this consumer feedback, the platform service engineers developed features that gave consumers availability across geographic fault zones.

The Walmart engineers developed mechanisms to replicate a service instance's data commits, at the request level, to other data centers. The consumer can choose a tier with synchronous replication and active multi-site processing, or asynchronous replication and active/standby site processing, depending on their need and propensity to fund the level of service they want.

This capability is made possible by the design and configuration of the environments. By defining similar resources and TCP/IP environments at each site, the engineers can apply a little creativity and greatly increase the scope of the processing environment, and subsequently the availability of a service instance. With this design (as shown in Figure 5-8), service requests can continue to be satisfied even with the loss of an entire data center.

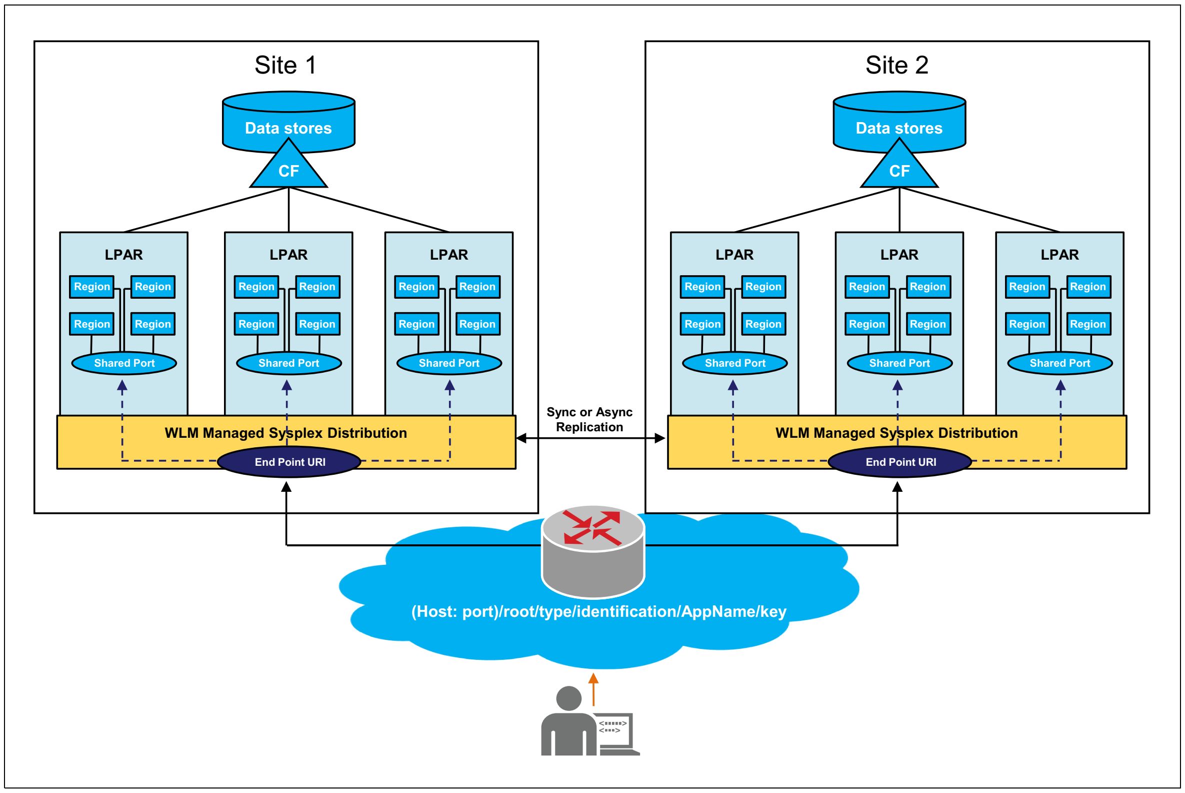

Site 1
Data stores
CF
LPAR
Region
Shared Port
WLM Managed Sysplex Distribution
End Point URI
Sync or Async Replication
Site 2
(Host: port)/root/type/identification/AppName/key

Figure 5-8 Multi-site hosting

5.5 VSAM RLS

The value that is associated with data sharing that is afforded by z/OS Parallel Sysplex was briefly described in 5.1.2, “Unique characteristics” on page 46. Although the relatively simple concept of serialized access to files or data sets from disparate systems is impressive, serialization at a more granular scope (such as at an individual record level within a data structure) is remarkable. VSAM record-level sharing (RLS) provides this unique capability.

The Walmart platform services engineers recognized the value of this feature and used the capability to provide a differentiated experience to their service consumers. However, configuration of the components that were related to RLS needed some focus to ensure a stable experience for consumers.

Great care must be given to ensure that this portion of the environment is configured appropriately for particular usage characteristics. This configuration can be complicated to get tuned for a particular use case, but some areas to focus on and some general guidance do exist.

5.5.1 Feature enablement

Consider the following key features:

- RLS_MaxCFFeatureLevel(Z|A)

  This parameter is defined in the IGDSMSxx PARMLIB member and controls the size of data that is placed in the coupling facility (CF) cache structures. A value of `Z` specifies that only control intervals (CI) less than 4 KB are placed in the cache structures. This setting generally works best for data that is mostly read-only. A value of `A` allows for CIs up to 32 KB to be placed in cache and often is related to data that is heavily updated.

- RLS CF CACHE in data class

  Values determine which components are cached during different types of activity. Walmart sets this value to ALL for all data classes that are associated with RLS data sets. This setting indicates that all data for the sphere is eligible for caching. Other options include NONE, UPDATESONLY (only updated control intervals are cached), and DIRONLY (only the directory is cached).

5.5.2 Sizing

Consider the following components for sizing:

- Buffer pools

  The RLS_MAX_POOL_SIZE parameter in IGDSMSxx limits the size of the RLS local buffer pool per system. The value of this setting should not exceed real storage availability and is recommended to be set less than or equal to 850 MB.

- Lock structures

  It is recommended that the following calculation is used for the RLS lock structure size:

  ```
  (10 MB * Number of Systems * Lock Entry Size)
  ```

  No less than 13 MB should be used. The `Lock Entry_Size` is determined by the MAXSYSTEM value that is defined in the coupling facility resource management (CFRM) policy and can be identified as listed in Table 5-1.

  Table 5-1 MAXSYSTEM values

MAXSYSTEM	Lock_Entry_Size
<=7	2
>=8 & <24	4
>=24 & <=32	8

- Cache structures

  The total RLS cache size for the environment is commonly recommended to be calculated by using the following equation:

  ```
  ((RLS_Max_Pool_Size) * number of systems)
  ```

  If multiple cache structures are defined, this aggregate size should be split among the cache structures.

5.5.3 System managed assignments

DFSMS can be configured to control cache structure assignments in several ways. Walmart elected to allow system managed assignments by associating each cache set with all available cache structures in the DFSMS configuration, as shown in Figure 5-9.

```
Cache Set                     CF Cache Structure Names
VSAMRLS1  VSAMRLS1         VSAMRLS2      VSAMRLS3      VSAMRLS4
VSAMRLS2  VSAMRLS1         VSAMRLS2      VSAMRLS3      VSAMRLS4
VSAMRLS3  VSAMRLS1         VSAMRLS2      VSAMRLS3      VSAMRLS4
VSAMRLS4  VSAMRLS1         VSAMRLS2      VSAMRLS3      VSAMRLS4
```

Figure 5-9 DFSMS cache set to cache structure associations

5.6 z/OS configuration

There are a couple of CICS parameter specifications that are described in Chapter 4, "The CICS systems programmer" on page 27 that require corresponding z/OS configuration settings to avoid issues. In particular, the MAXOPENTCBS and MAXSSLTCBS settings in CICS must be complemented by z/OS parameters in BPXPRMxx.

The MAXPROCSYS and MAXPROCUSER settings in BPXPRMxx establish higher-level limits on the numbers of processes or TCBs that are allowed in the environment. Some projections and planning are necessary to determine values for these settings that accommodate the expected CICS workloads. These parameters are described and example definitions are provided in the following sections.

MAXPROCSYS controls the maximum number of processes per system or LPAR. An example of this setting is shown in Figure 5-10.

```
/*************************************************************** */
/*************************************************************** */
/*    Specify the maximum number of processes that z/OS UNIX      */
/*    will allow to be active concurrently.                       */
/*                                                                */
/*                                                                */
/*    Notes:                                                      */
/*                                                                */
/*        1. Minimum allowable value is 5.                        */
/*        2. Maximum allowable value is 32767.                    */
/*        3. If this parameter is not provided, the system default */
/*          value for this parameter is 900.                      */
/*                                                                */
/*                                                                */
/*                                                                */
/*****************************************************************/
MAXPROCSYS 3000                  /* System will allow at most 3000
                                    processes to be active
                                    concurrently             @P9C*/
```

Figure 5-10 MAXPROCSYS sets maximum number of processes per system

MAXPROCUSER controls the maximum number of processes per user ID. Walmart uses a unique user ID per region, so this setting equates to the maximum number of processes per region in this case. An example of this setting is shown in Figure 5-11.

```
/*****************************************************************/
/*                                                                */
/*   Specify the maximum number of processes that a single user    */
/*   (that is, with the same UID) is allowed to have concurrently  */
/*   active regardless of origin.                                 */
/*                                                                */
/*   Notes:                                                       */
/*                                                                */
/*        1. This parameter is the same as the Child_Max variable  */
/*           defined in POSIX 1003.1.                             */
/*        2. Minimum allowable value is 3.                        */
/*        3. Maximum allowable value is 32767.                    */
/*        4. If this parameter is not provided, the system default */
/*           value for this parameter is 25.                      */
/*                                                                */
/*                                                                */
/*                                                                */
/*****************************************************************/
MAXPROCUSER(2500)                /* Allow each user (same UID) to
                                    have at most 25 concurrent
                                    processes active              */
```

Figure 5-11 MAXPROCUSER sets maximum number of processes per user ID

5.7 Security

Security for the z/OS cloud services in the Walmart environment is provided by Resource Access Control Facility (RACF). A dedicated team manages RACF in Walmart; therefore, this area is not technically the purview of the z/OS systems programmer. However, these entities work closely together and experience some overlap in concern. In many shops, these roles are often within the same group or team.

The cloud services, particularly the caching service, might not require security controls in all cases (cache can contain mundane, inconsequential data sometimes). However, in most cases, some basic authentication is needed.

With the fact that many users of the caching service were not z/OS developers, they did not have RACF credentials to use for authentication. Walmart system and security engineers established “service IDs” for this purpose. These IDs can be used to authenticate a consumer’s access to the service, but did not allow for logging on to Time Sharing Option (TSO). This issue worked out well considering that the cloud services are accessible only through ReSTful APIs, so access to the actual systems was not necessary.

The on-demand self-service cloud characteristic still needed to be provided for these consumers; therefore, immediate access to these IDs was a requirement. Through collaboration between the platform and system engineers and the security engineers, a process was initially established in which the security engineers pre-defined groups of these service IDs for use during service provisioning. The platform engineers pulled from this pool of IDs as needed for service deployments. Then, as the pool diminished, more IDs were created and added to the pool.

Over time, as the delivery model and credibility were solidified, an agreement was established between the platform and security engineers to allow for dynamic provisioning of the IDs along with the service instances. These IDs are now provisioned and managed through the same self-service channels and with the same suite of provisioning automation as the other cloud services that are provided by the z/OS platform service engineers.

5.8 Summary

A well-designed operating environment is necessary for effective service hosting. The environment should be configured for efficiency, resiliency, scalability, and flexibility to provide a solid foundation for the runtime environment to deliver quality services.

In this chapter, numerous aspects of how these qualities can be achieved on the z/OS platform were described, which established the z/OS systems programmer as a key resource in the service provider role.

6

Operational considerations

This chapter describes operational considerations relating to hosting cloud services on IBM z/OS and IBM CICS Transaction Server for z/OS. The authors share the experience of the Walmart team and demonstrate how Walmart dealt with these challenges.

This chapter includes the following topics:

6.1 Capacity planning

Capacity planning is an essential component of any mainframe environment. Capacity planners are primarily concerned with ensuring that applications perform reliably and can handle spikes or increases in usage without negative effects on quality of service; that is, still operating under agreed-upon service level agreements (SLAs). Another concern is to ensure that the CPU usage for the software components is within the desired constraints.

Capacity planners can use many tools to achieve these goals. In a CICS environment, many of the tools are focused on the collection and analysis of System Management Facility (SMF) records that are output by CICS Monitoring Facility (CMF) that uses SMF type 110 records. It is essential in a service provider situation to properly understand and have good communication with your capacity planners. This relationship can yield the following benefits:

- Sharing of tooling and data. Within the Walmart organization, many of the same tools that are used by the capacity planners can be used in monitoring and measurement. For more information, see 6.3, “Metering and measurement” on page 61.
- Mitigate any concerns that capacity planners might have by being transparent in what services are hosted in CICS.
- Ensure that your environments have the capacity to grow with your needs.

6.1.1 The concerns of a capacity planner

It is essential to understand the concerns that a capacity planner might have and do all you can to mitigate these concerns.

CPU usage

One initial reaction from adding these services to CICS, or making these services available, might be the higher cost because of an increase in CPU consumption.

Through the implementation of the caching service, Walmart found the opposite was true; that is, CPU usage did not increase. The caching service reduced the processing that is required to fetch resources that are now cached, which resulted in a net reduction in CPU usage.

An example within Walmart was a CICS web service application that ran a complex IBM DB2 query, which, because of its complexity, was run for a two-week period only, six times per year. During the morning batch runs of updates to the DB2 tables, the information was also loaded to this application’s caching table. The query to DB2 in the CICS web service was replaced by a call to the caching service, which significantly reduced I/Os and subsequent CPU cycles.

Monitoring service usage

All current application and system monitoring tools, including tools that use SMF data, are still applicable to monitoring services. These tools give capacity planners the ability to view the usage of the environment as a whole. With an intelligent, well thought out design, these tools allow capacity planners, CICS systems programmers, and service providers the ability to view on a granular level which consumer is using which resources.

6.1.2 Partnership with capacity planners

Capacity planners focus on the mainframe environment at a high level. Within Walmart, the addition of microservices has not affected the mainframe environment at this level.

Although this is the case (as you have seen through multiple examples in this book) the true key to success in the new mainframe world is developing partnerships with the teams throughout your organization. With these partnerships in mind, building notifications into the tool to notify interested parties can greatly increase the visibility of the environment usage among teams as part of an overall provisioning solution.

For example, when a new microservice is provisioned, notifying the capacity planners allows them to properly monitor trends in the environment usage and provide data, such as MIPS usage, which can be used to project future requirements on your systems.

6.2 Scaling and elasticity

Elasticity or elastic scale is one of the key concepts involved when building cloud services. It is also one of the five National Institute of Standards and Technology (NIST) essential characteristics of cloud. For more information about these characteristics, see 1.3, “How z/OS and CICS are relevant to cloud” on page 3, and 3.3, “Five essential characteristics of cloud” on page 17.

Elasticity, or *elastic scale*, refers to the need for the service to be scalable to meet the needs of the service consumer. For z/OS based services, this scaling should happen on-demand (within pricing restrictions, see 6.3, “Metering and measurement” on page 61) and seamlessly.

The service consumer should not be aware on any level of the requirements of the system and the environment should grow according to needs. In addition to the service consumer requiring the service to grow, they are also concerned with the inverse; that is, having a provisioned environment that scales back when demand for a service decreases.

For more information about scaling and elasticity in a cloud environment, see *Creating IBM z/OS Cloud Services*, SG24-8324.

The following methods are suggested in this section to achieve elasticity of services:

- Run with spare capacity: Create a large environment that can handle increases in demand to maximums that the consumer requires without the requirement to dynamically grow.
- On-demand growth: Create environments that run at capacity that can grow and shrink to meet increases and decreases in demand.

6.3 Metering and measurement

The key to creating a successful cloud service is good service design. The design of the cloud services was described at great length throughout this book. This section describes how effective design makes measurement of usage of provisioned services easy to monitor and control.

Within the Walmart environment, monitoring of service usage and control of that usage based on limits is achieved on multiple levels. Service providers at Walmart monitor system usage by using the same tools as capacity planners use to monitor the IBM z Systems environments. These tools analyze SMF data, which includes SMF type 110 records output by CICS, SMF type 30 records for common address space work, and SMF type 7x records for IBM Resource Measurement Facility (RMF) data.

How can service providers use SMF data to measure usage for individual consumers? In the Walmart multi-tenant environment, work is isolated, based on enforcing strict naming conventions throughout the entire service. Figure 6-1 shows how a workload is tracked by using the transaction ID that is unique to a particular consumer.

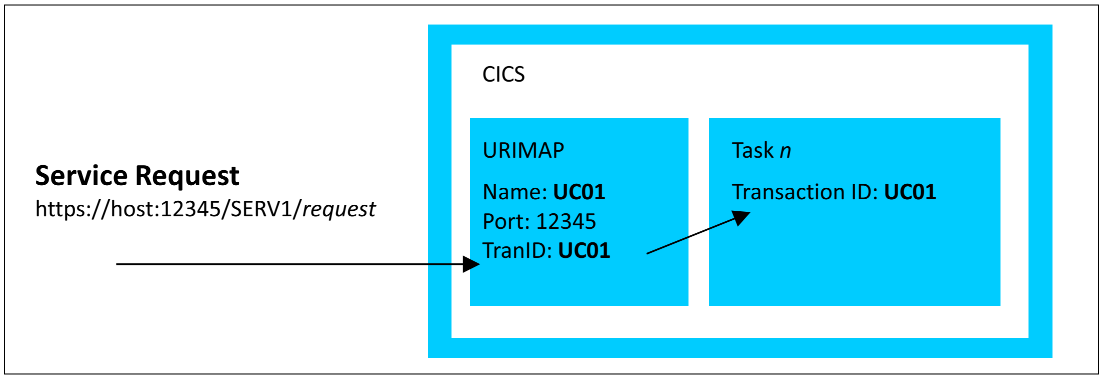


Figure 6-1 Metering service usage with unique transaction IDs

Each request that comes into the environment is routed by using a URIMAP resource (unique to a consumer) to a unique transaction ID. SMF data for this transaction is then output to SMF log streams. For live analysis of the environment, Walmart does not wait for the data to be output to log streams. By making use of an exit at the XMNOUT Global User Exit Point, Walmart captures transaction data as it is being sent to SMF and avoids the need to decompress the data output to SMF log streams.

The ability to monitor each individual tenant's service usage is essential when calculating how much to charge for service usage.

6.3.1 Charging for service usage

Through monitoring the transaction information output from SMF data, a service provider can calculate how many times a transaction is started. This transaction count can be used for billing and usage capping. The following approaches are used for billing:

- Projected-usage charging
- Pay per transaction

Projected-usage charging

By using projected-usage charging, the service consumer provides an estimate for how many requests (which directly map to transactions) they expect to use in a specific period. Consumers then pay based on this projected rate.

If a consumer then exceeds this prepaid limit, policies can be implemented to restrict transaction flow rate until more capacity is arranged. These restrictions are implemented by using transaction classes.

When a service consumer exceeds an agreed capacity by a certain percentage, the service is then halted and it returns an HTTP return code 404 to the requester. The service consumer can then purchase more capacity, service restrictions are removed, and limits are reset based on this newly defined capacity.

Pay per transaction

An alternative to charging based on projected-usage is to charge per transaction; that is, charging for actual usage. This pricing model can be restricted by the organization's internal charging capabilities if it cannot charge on per usage basis. The same tools can be used to measure transaction throughput by using SMF data and then charging the consumer for their usage at a specific interval.

6.3.2 Use of measurement information for scaling

In addition to using the monitoring data output by CICS for billing, you can use this data to enable further automation within your environment. By sing this SMF data, you can automatically provision and deprovision CICS as required.

6.3.3 Evolution of service-oriented architecture to cloud service

It is important to highlight how the thinking on this topic evolved to meet the new cloud service model. It is not the first time Walmart made core services available through a web services interface. Walmart used technologies, such as CICS web services, in the past to make functions available within CICS.

These previous experiences highlighted to Walmart how metering and measurement are essential for a reliable services interface. An example of this necessity was a previous CICS web services application that was used by an internal development team within Walmart. The use of this service was steady for some time, and then the CICS operations team noticed a significant increase in the usage of the service. The application owners insisted that the new traffic was not because of their application. After some investigation, the operations team realized that it was another development team that was using the service through discovery of the published WSDL.

This experience was taken into account in the design of Walmart's recent microservices. The multi-tenant, isolated, and consumer-provisioned services provide the ability to monitor exactly who is using the service (through individual URIs, transactions, and user IDs). The consumer is responsible for controlling access to that instance of the service and the credentials to access the service.

6.4 Maintenance

This section describes the considerations in maintaining your CICS environments that host your provisioned services.

The main principles of maintaining this type of environment are the same as maintaining any other highly available CICS environment. The same considerations of maintaining availability of your services through any maintenance cycle are still applicable. As a guide, the points the are described in this section highlight some of these considerations.

z/OS maintenance and upgrade

It is unlikely that z/OS maintenance will be a consideration of the service providers, unless they share z/OS operations responsibilities. However, it is likely that the z/OS operations teams will need to perform rolling maintenance and upgrades to the production environment, performing maintenance on each LPAR, one at a time.

This rolling upgrade approach allows you to use the high availability (HA) features that are built in z/OS to allow normal operations to continue. In Walmart, the service providers used DNS forwarding (allowing switching between sysplexes), IBM Parallel Sysplex, sysplex distributor, virtual IP addresses (VIPAs), and TCP/IP port sharing to achieve HA at the sysplex and LPAR levels.

By using this approach, Walmart performed a full z/OS upgrade with zero downtime while the production environment was running.

CICS and CICSPlex SM maintenance and upgrade

As with the z/OS maintenance and upgrades being performed by a z/OS operations team, CICS and IBM CICSPlex SM maintenance and upgrades are likely to be performed by a CICS operations team, which might be the same as the service provider.

The same HA tools that you use for z/OS upgrades also apply here. You can take a rolling upgrade approach and upgrade each CICS region independently allowing work to be handled by other regions while a region is being upgraded and cycled.

Initial program load (IPL) of LPAR and sysplex

If you use HA tools, such as sysplex distributor in a multi-LPAR environment, you can re-IPL an LPAR while still allowing service consumers to use the services. During the restart of the LPAR, the workload fails over to the other LPARs in the sysplex. Be sure to shut down all CICS regions on the LPAR before you restart.

If it is required that you IPL the entire sysplex, the DNS forwarding configuration (as described in Chapter 5, “The z/OS systems programmer” on page 45) can failover to a backup sysplex if configured. It is unlikely that you must to complete this process as part of a planned outage.

6.5 Problem determination

This section describes how service providers can aid themselves, systems programmers, and operations teams in analyzing problems that can occur in a service environment.

As described in 6.4, “Maintenance” on page 63, all of the tools that you use for problem determination in a CICS environment still apply to problem determination in a services environment. The extra challenge with a multi-tenancy service environment is how to determine which consumer of the service is being affected by an issue, and resolving that issue for that consumer.

The key to overcoming this challenge (see in 6.3, “Metering and measurement” on page 61) is clear, consistent, and logical isolation of service consumers. Isolation of service consumers can be easily achieved through enforcing strict naming conventions.

6.5.1 Isolation of consumers in a multi-tenant environment

In designing isolation of service consumers, Walmart used the method of imbedding all CICS resource definitions with the same consistent ID, and by doing so, uniquely identifying the service consumer. This convention also extends to all z/OS resources that are associated with that consumer, such as including a unique ID in a data set qualifier or prefixing all DB2 table names with this identifier.

When Walmart services are provisioned, a new service instance is created that defines the following minimum CICS resources:

- URIMAP
- TRANSACTION
- TRANCLASS

Other z/OS resources can also be created during this provisioning process; for example, the definition of a new VSAM Record Level Sharing (RLS) data set to hold data that is used by the service.

Figure 6-2 shows the CICS resource definitions that are automatically provisioned when a new service instance is requested. The VSAM data set `H1.H2.H3.UC01FILE` also is created.

```
EX G(UC01)
ENTER COMMANDS
 NAME       TYPE          GROUP
 UC01VSAM   FILE          UC01
 TCLUC01    TRANCLASS     UC01
 UC01       TRANSACTION   UC01
 UC01       URIMAP        UC01
```

Figure 6-2 Example CICS resource definitions

In this example, the service consumer is assigned the unique identifier **`UC01`** after requesting a new service to be provisioned. This unique identifier is used in all resource definitions, and in the associated, dynamically created z/OS resources.

By using this naming convention, the service consumer or a subset of service consumers can be identified when a problem occurs in a multi-tenant environment, which can aid in problem determination.

6.5.2 Standards

Throughout all Walmart services, another key tool in problem determination from the service provider and service consumer perspectives is strict adherence to expected standards.

The primary example of adherence to standards within Walmart is for services that are accessed through HTTP. The HTTP standards for return codes and reasons should be adhered to as the standard specifies. The use of proper error codes (for example, the 4xx and 5xx error codes) can help to pinpoint where a problem might be in a system or if a consumer is using a service incorrectly.

Although not all 4xx and 5xx error codes are perfectly suited for CICS cloud services, an attempt should be made to use them as closely as possible. Another technique Walmart adopted is assigning specific error messages to fully qualify the type of HTTP status code. These messages are returned in the HTTP status text. For example, for a CICS service that accesses a VSAM file (if the file is closed), the service returns an HTTP status code 507 and an HTTP status text indicating that the file being accessed is closed. Although there is no standard HTTP status code for a *file closed* status, the 507 code does indicate *insufficient storage*, and the HTTP status text supplies an error message that indicates the file condition, file name, reason code, and the name of the program that detected the error.

6.5.3 Communication with other teams

Throughout this chapter and the book, great emphasis is placed on enabling good communications throughout your whole organization, from service consumer, through provider and systems programmers.

You can enhance this communication, and in particular aid in problem determination, by documenting what your services do, how to use them, and from a systems programming perspective, what the application structure and logic flow looks like, which enables greater collaboration.

Among other tools, Walmart uses wiki pages, blogs, and documentation to aid in this collaboration across their teams. Each team can access the shared social resources and contributes to them.

6.6 Summary

This chapter highlights some of the operational considerations of hosting cloud services in CICS. From the monitoring, measurement, scalability, and problem determination perspectives, the Walmart example shows that true multi-tenant isolation that uses standard tools available in CICS can offer an excellent environment to host cloud services.

7

DevOps perspective

This chapter reviews how Walmart uses cloud services to address the challenge of variable speed IT.

To allow rapidly evolving service consuming applications to move from development to production while also evolving those services in response to consumer feedback required the platform engineers to take on the responsibility for release management.

This process, also known as DevOps, sits at the cross roads between development, product management, quality assurance, and other engineering efforts, including automation.

This chapter includes the following topics:

7.1 The challenge of variable speed IT

Walmart is very much a technology company. In fact, Walmart uses many innovative technologies in some capacity. Walmart's rich history of technology investment continues unabated through today and provides a good basis for a discussion about the concept of variable speed IT.

If you consider the requirements that are associated with current trends, such as e-commerce and mobile, you can see a recognizable example of variable speed IT. These requirements are highly variable and dynamic in functionality and demand. They also interact with core systems that support brick and mortar traditional systems, such as inventory and replenishment.

Figure 7-1 shows the different perspectives of the two worlds of new applications and traditional applications and some broadly generalized categorizations.

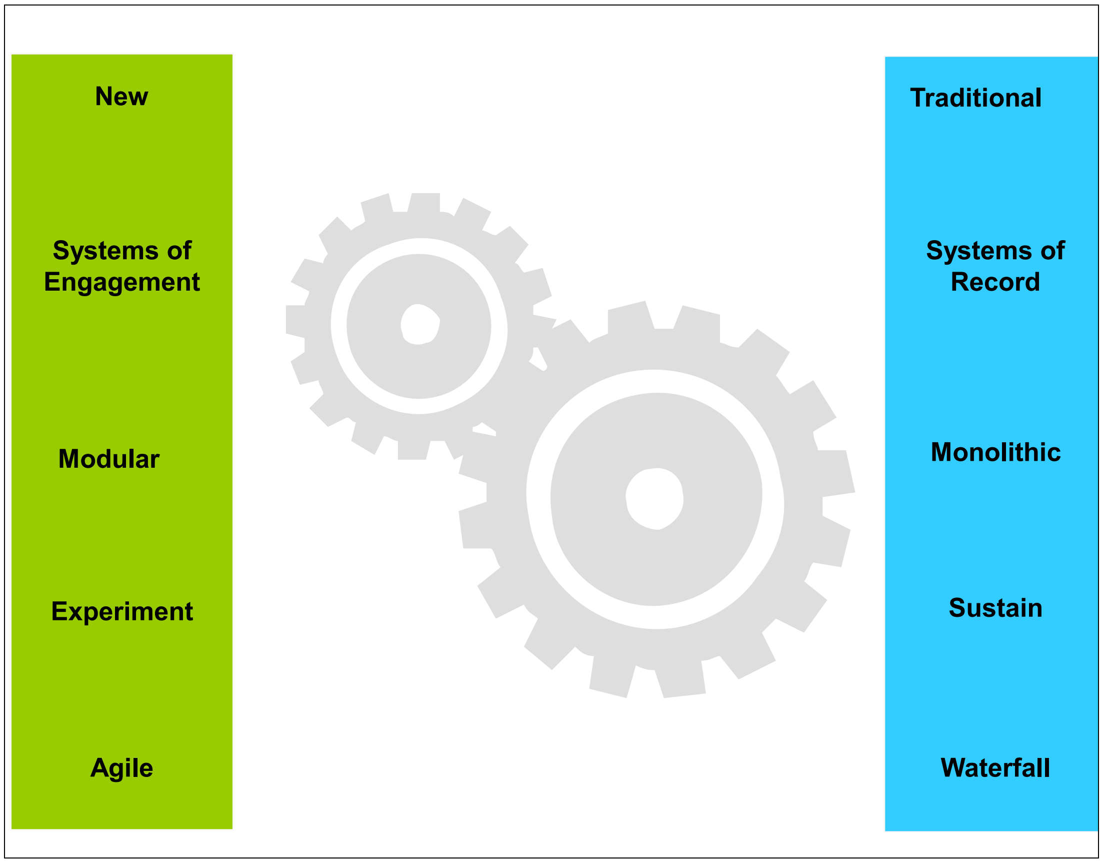


Figure 7-1 Variable speed IT: Two worlds?

The two worlds of variable speed IT exhibit the following contrasting characteristics:

- New applications versus traditional applications.
- Systems of engagement (directly accessible and tied to retail customer experience) versus systems of record (core applications, master data management, back office, human resources, and so on).
- New applications that tend to be more modularized versus traditional applications that tend to be more monolithic and intertwined with other core systems.

- The primary goal of a new application is to enable experimentation versus the goals of sustaining functionality and managing risk, which are predominant in a traditional application.
- Work management in a new application tends to be accomplished with agile methodologies versus waterfall methodologies in a traditional application.

Although these characteristics provide a nice and neat delineation between new applications and traditional applications, this delineation does not necessarily reflect the real world in the experience of the authors of this book. In reality, various enterprise systems fall somewhere on a spectrum between application or system types.

It is for this reason that the term *variable speed IT* is preferred over *bimodal* or *two-speed IT.* The term “variable speed IT” is a more accurate term, and it is descriptive of the approach that Walmart took to tackle these challenges from an IBM z/OS perspective. However, maintaining a simple, two-world view such as this one is beneficial for our purposes.

The challenge is dealing with the disparity in speed when involvement from both types of approaches is needed to achieve an objective. In a theoretical example, systems of engagement developers who are responding to a market demand might need to quickly add or enhance a mobile application feature that depends on certain systems of record components.

Aversion to operational risk, complexity, or highly stringent governance around the traditional system of record components hinders the ability to move with the necessary speed. This aversion is not good for the business and it is all the more troublesome that each of these IT speeds is valid and appropriate for what they represent. So, what can you do? Embracing DevOps provides a solution.

7.2 The role of DevOps

DevOps is not a technology challenge. Although technology plays a part, DevOps is largely about people, culture, and process. Walmart does not endorse any specific implementation or particular tool, but offers some points to consider based on their experiences.

Shifting to an organizational perspective, it is worth noting that the natural alignment between the speeds of IT and the groups that are associated with the development and operations teams is a logical and obvious one. Figure 7-2 on page 70 shows the priorities that are associated with each team.

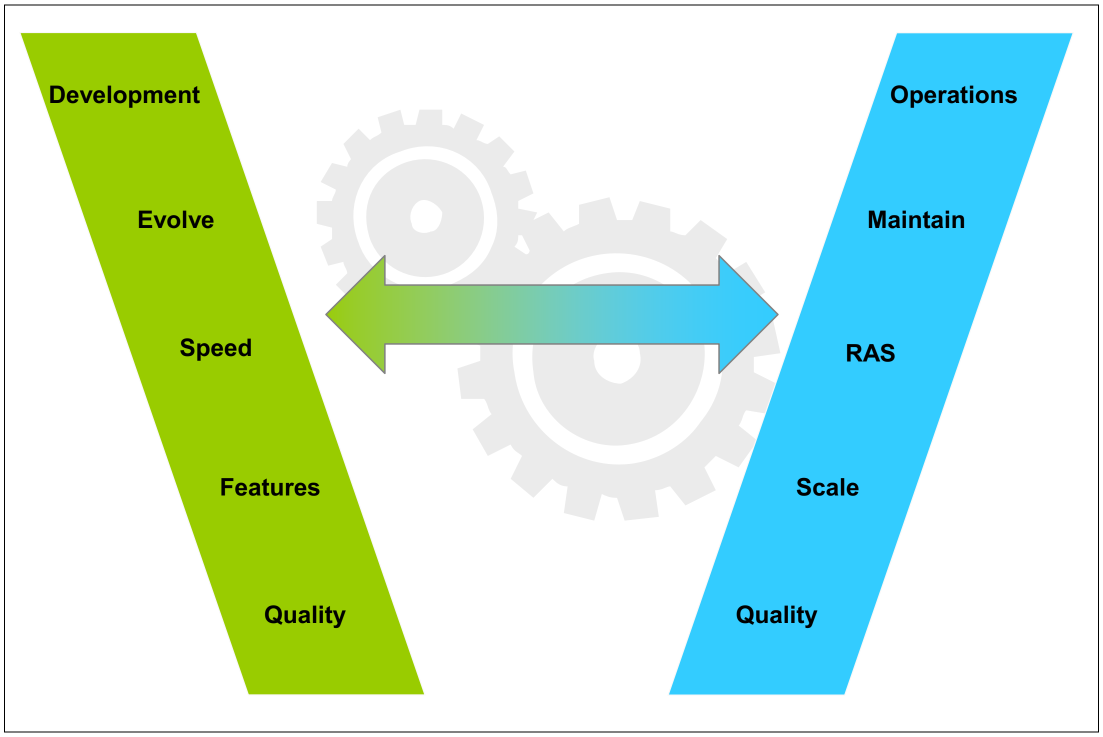


Figure 7-2 DevOps implications

The development and operations teams are concerned with a particular set of attributes that are relevant to their respective areas of responsibility. Consider the following points:

- The development team must progress and evolve applications, while the operations team needs to ensure that capabilities are maintained.
- Development must be agile and quick, while operations must ensure that reliability, availability, and serviceability (RAS) is not compromised.
- Development must provide new features, while operations must deliver scalability and stability.
- Development must ensure quality in new business services, while operations must ensure quality in capabilities.

Both teams are focused on the same attributes. These teams might approach the achievement of these characteristics in a different manner, but they both care about quality of service. In fact, both teams (and all other siloed factions within the organization) ultimately care about the same goal: to provide high quality, reliable, added value technology solutions that support and enhance their business. But, limitations on visibility and focuses of specialization result in a disconnectedness regarding the tasks at hand that can affect end-to-end quality. To improve this situation, these groups must be brought together and converge around their shared goals and perspectives.

DevOps idea now focuses on the philosophy built around enhancing communication and collaboration between development and operations areas. This philosophy is supported by automation and tools that enable tracking and feedback. The result is the regular delivery of high-quality, reliable, scalable technology solutions that enhance the business.

This is also where the z/OS cloud services team at Walmart focuses and where contributions become relevant. How do you support rapid development to meet new business objectives (while preserving the sanctity of core systems) given that many of the intellectual property assets within those systems remain highly valuable to existing business? How do you engage the personnel responsible for these systems when they are often exclusively operationally focused? Walmart chose to develop cloud services.

7.3 Cloud services

The approach at Walmart is to establish a layer that acts as a differential to accommodate the different speeds of new development versus the needs of traditional systems. As shown in Figure 7-3, this layer consists of the following primary components:

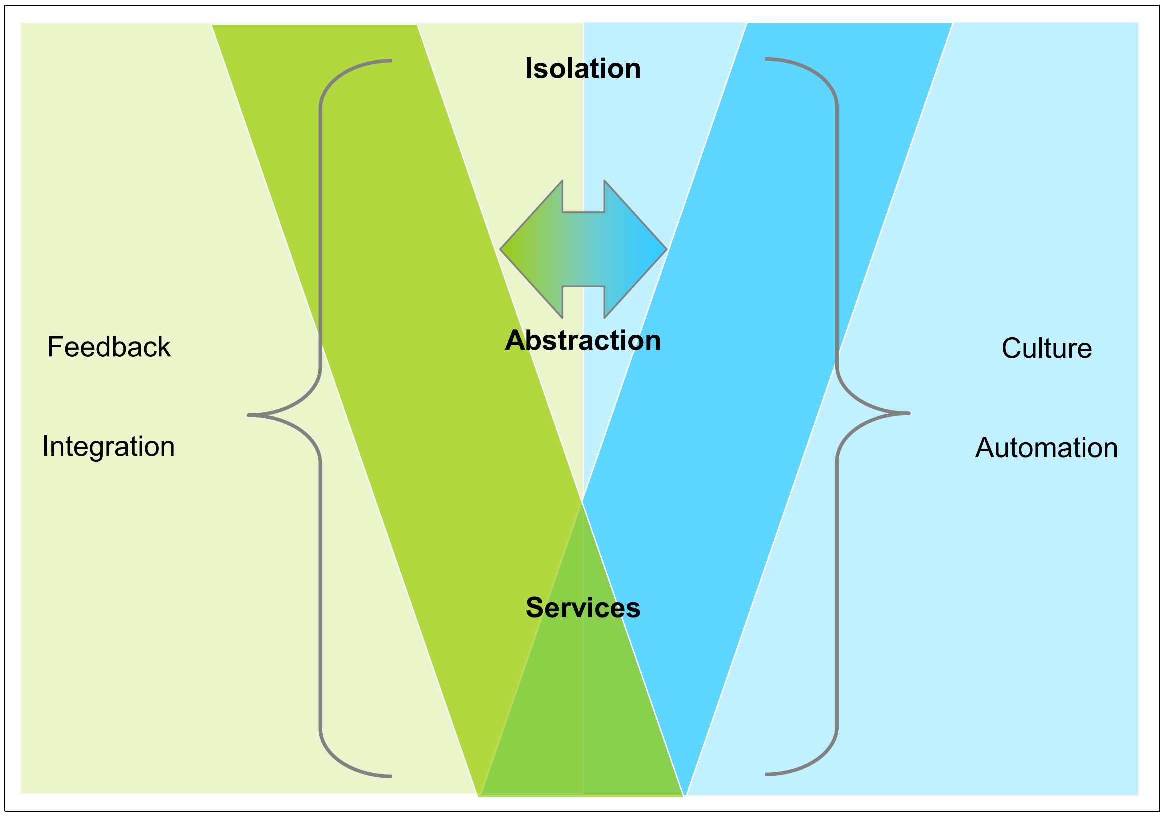


Figure 7-3 Supporting isolation, abstraction, and services with feedback, culture, integration, and automation

- Isolation

  Walmart used the advanced virtualization capabilities of the z/OS platform to establish hosting environments for new workloads that are isolated from traditional workloads, while maintaining proximity to valuable data assets in core systems.

  These isolation zones consist of separate sysplexes and hosting environments (such as dedicated region clusters) within sysplexes that are hosting traditional workloads. Virtualized isolation gives the operational staff the confidence that core systems are protected. At the same time, virtualized isolation provides locally optimized access to the related assets.

- Abstraction

  Walmart used various mechanisms to abstract the complexities of the core systems and the platform from the developer community to address skills gaps and promote integration.

  The primary mechanism is a RESTful API. Generally, developers do not need to know that the services are hosted on a mainframe. This approach enabled Walmart to move services from one data center to another with no effect on users. This configuration provides the freedom for consumers (the developers) and service providers to change their implementations as needed.

- Services

  Walmart established a suite of utility services that was designed and developed following the cloud computing service delivery model, as defined by National Institute of Standards and Technology (NIST) publication SP 800-145[1], and including on-demand self-service and rapid elasticity.

  All services are provisioned and made available by the use of RESTful APIs, which provide accessibility from any application platform with abstraction. Providing capabilities in this manner also promotes more loosely coupled application architectures that result in increased agility.

The approach of using z/OS and CICS to deliver the cloud services solution provides developers with quick, easy access to an array of services that give them more autonomy when interacting with core systems. Previous sections in this book described caching, object stores, and ID generators, but Walmart developed more services, including data access services and messaging. By providing these services by using the fully automated delivery of standardized components, the risk is reduced, while maintaining a level of operational control and confidence.

Some of the services provide lightweight entry points for data assets and can be used in place of heavier, more resource-intensive calls to the true back-end systems of record. Walmart noted a reduction in net CPU consumption by as much as 70% for some applications. Consuming these resources more efficiently leads to reduced operational costs, more stable environments, and frees up resources to satisfy other business needs.

As described in 7.2, “The role of DevOps” on page 69, DevOps is not only about technology; it also is about people, culture, and process, which can be addressed by the following factors:

- Feedback

  Feedback is important in various ways. The cloud services team spent many hours engaging directly with application developers to get their input and feedback about services and features.

  The application developers are guiding the development efforts of the cloud services team. The team also works closely with the systems administrators to ensure that their needs and concerns are addressed.

  The cloud services team, and some of the operations groups, developed dashboard services to provide immediate feedback about system and service activities. Communication and information were critical.

- Integration

  You must understand how services are to be consumed. Application developers might not be familiar with the mainframe, so choose open standards that are familiar to a wide audience. For more information, see Chapter 2, “The service consumer” on page 7.

[1] For more information about the NIST Definition of Cloud Computing, see this publication: http://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-145.pdf

- Culture (or attitude)

  This issue was likely the most challenging aspect of the work to develop cloud services, mostly in the operational groups due largely to all of the issues that were described in this chapter regarding the role of DevOps, such as maintaining the integrity of systems. Despite the challenges, Walmart made, and continues to make, progress.

- Automation

  Automation is vital. Self-service provisioning must work at the speed of IT. For more information, see Chapter 3, “The service provider” on page 13.

By providing on-demand, self-service access to services, the cloud services team allows developers to move faster. They have experience of new applications being deployed to production in as little as one week after acquiring resources in the development and quality assurance (QA) environments. This time frame is expected for all new development.

The ability to move fast enables companies to become agile. By using automation, the time and effort that are invested in acquiring a service instance are nearly eliminated, which leads to a degree of freedom that promotes experimentation and rapid, iterative development. Some of the services were used in ways that were never considered during design and creation.

The cloud services team at Walmart is small (only a few people) but with much talent and a good mix of skills. Through this approach and with the correct attitude, they made a substantial impact with relatively little investment. This benefit is realized downstream and through removal of waste from processes.

7.4 Promotion to production

As described in Chapter 2, “The service consumer” on page 7, application developers at Walmart use a web portal to provision an instance of a cloud service for development. At the same time, the automation behind the portal also provisions an instance of the service in a QA environment. This approach enables the application to be submitted for testing without any further interaction. When developers who requested that the service be ready to go live, they approach the cloud services team with an approved change control record.

Resources, such as CPU and storage that is consumed by a service instance in QA, are used as a model to provision a service instance in production quickly. The process is automated, but currently requires a member of the cloud services team to start it. The long-term plan is for the provisioning process to automatically forward the usage estimates that are provided by the developer (for example, transactions per second) to the capacity planning team.

After the service instance is available in production, the developer can promote the application. To access this production version, all a developer must do is change the host and port for the service. As described in Chapter 3, “The service provider” on page 13, the rest of the Uniform Resource Identifier (URI) stays the same.

The development and QA instances of cloud services are free in Walmart’s environment, a practice that is in line with most public cloud providers with which the on-premises Walmart services compete. This practice, along with the speed of provisioning, removes a critical barrier to adoption by developers.

At Walmart, all development service instances are hosted in the same pool of CICS regions. There is no security (authentication or SSL) in this environment, the configuration of which can be another potential barrier to adoption by developers. Full security is available as an option in QA for testing a configuration that is then carried forward into production. Although this configuration puts much responsibility on the developers, it enables them to move faster. In production, applications might be allocated to their own pool of CICS regions or in close proximity to their data source for improved performance. For more information, see Chapter 6, “Operational considerations” on page 59.

7.5 Process and governance

Before requesting a cloud service in production, there are some governance steps a developer must follow. At Walmart, developers must obtain approval from a senior developer and their management. They must also demonstrate that funds are available by providing a valid project ID. The process today uses change control procedures and is not fully automated. However, the provisioning process takes only seconds.

Walmart plans to streamline the process by generating change control records by using an auto-progressing workflow. This workflow will also notify resource owners (for example, storage and CPU owners) of the requirements for each new service instance.

Note: IBM UrbanCode™ Deploy provides automation for the process and governance of moving application changes from development through QA to production, including any required approvals by team members. CICS provides a plug-in for deploying all styles of application. For more information, see *CICS and DevOps: What You Need to Know*, SG24-8339.

7.6 Service versioning

With the two-week sprints that are used at Walmart, it is necessary to deliver capability in small, reliable pieces, which means regularly updating each service.

Changes are first pushed to development and then QA before production. All code is under version control, with access limited to the cloud services team. Today, all service instances and hence all users, receive the update at the same time. To date, these changes are incremental and compatible with an earlier version. In the future, it might be necessary to make a significant non-backwards compatible change to a service interface, such as a new parameter or data format, which requires developers to change their applications. So, how can changes be made without disrupting developers and users of the applications they provide?

The solution that is being considered at Walmart is semantic versioning. This solution includes version, release, and patch identification, which relates to the implementation of the provided service. For more information, see this Semantic Versioning technical white paper:

http://www.osgi.org/wp-content/uploads/SemanticVersioning1.pdf

At a minimum, the version and release levels will be included in new URIs going forward, as shown in the following example:

```
(Host:port)/root/type/version/org/tenant/resources
```

This approach would allow the developer to choose when to upgrade. It would also allow the cloud services team to introduce breaking changes. The ability to independently deploy services and the applications that use them is the most important principle of microservices architectures.

Note: Walmart is evaluating CICS application multi-versioning. This capability was introduced in CICS TS V5.2 and allows several different versions of an application or service to be installed onto the same set of CICS regions without changes to load module names. Users can access different versions of a service by using URI naming conventions, as described in this section. This use allows the introduction of breaking changes without disruption to developers or the use of more infrastructure. For more information, see *Cloud Enabling IBM CICS*, SG24-8114.

7.7 Summary

It might be said that Walmart adopted a DevOps-style approach to address the variable speed IT challenges. At a minimum, Walmart tried to address many of the same components that are regularly highlighted in the DevOps vernacular. Ultimately, Walmart is not too interested in the label. They are interested in the results, and they achieved some positive results by using this approach.

Related publications

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

IBM Redbooks

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

- *Creating IBM z/OS Cloud Services*, SG24-8324
- *CICS and DevOps: What You Need to Know*, SG24-8339
- *Cloud Enabling IBM CICS*, SG24-8114
- *Application Development for IBM CICS Web Services*, SG24-7126

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

ibm.com/redbooks

Other publications

The publication *Building Microservices,* by Sam Newman (ISBN:978-1-4919-5035-7) is also relevant as a further information source.

Online resources

The following websites are also relevant as further information sources:

- The Making of IBM Design Thinking:

  http://www.ibm.com/design/social.shtml
- The NIST Definition of Cloud Computing:

  http://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-145.pdf
- Semantic Versioning:

  http://www.osgi.org/wp-content/uploads/SemanticVersioning1.pdf

Help from IBM

IBM Support and downloads:

ibm.com/support

IBM Global Services:

ibm.com/services

Redbooks How Walmart Became a Cloud Services Provider with IBM CICS

(0.2”spine)
0.17”<->0.473”
90<->249 pages

SG24-8347-00

ISBN 0738441732