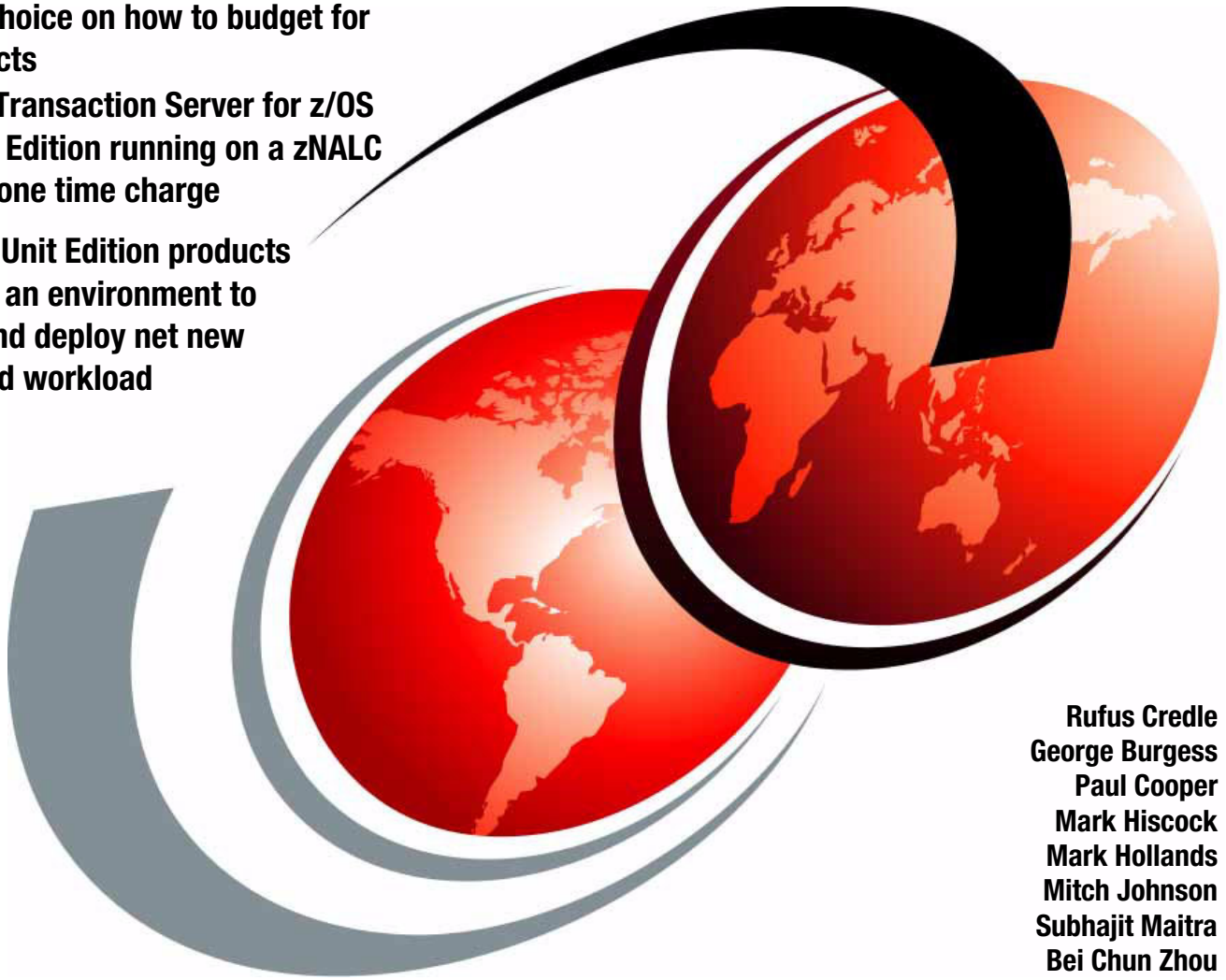
Draft Document for Review August 27, 2014 11:59 am SG24-8225-00

An Architect's Guide to New Java Workloads in CICS (Transaction Server)

- Value Unit Edition family of products allows a choice on how to budget for new projects
- IBM CICS Transaction Server for z/OS Value Unit Edition running on a zNALC LPAR is a one time charge
- Use Value Unit Edition products to provide an environment to develop and deploy net new Java based workload



Rufus Credle
George Burgess
Paul Cooper
Mark Hiscock
Mark Hollands
Mitch Johnson
Subhajit Maitra
Bei Chun Zhou

Redbooks

ibm.com/redbooks

International Technical Support Organization

An Architect's Guide to New Java Workloads in CICS

October 2014

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

First Edition (October 2014)

This edition applies to CICS Transaction Server for z/OS V5.2, WebSphere Application Server Liberty Profile, WebSphere Application Server for z/OS, Operational Decision Manager for z/OS, etc.
This document was created or updated on August 27, 2014.

Contents

Notices

This information was developed for products and services offered in the U.S.A.

IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not grant you any license to these patents. You can send license inquiries, in writing, to:
IBM Director of Licensing, IBM Corporation, North Castle Drive, Armonk, NY 10504-1785 U.S.A.

The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law: INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to non-IBM websites are provided for convenience only and do not in any manner serve as an endorsement of those websites. The materials at those websites are not part of the materials for this IBM product and use of those websites is at your own risk.

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

Any performance data contained herein was determined in a controlled environment. Therefore, the results obtained in other operating environments may vary significantly. Some measurements may have been made on development-level systems and there is no guarantee that these measurements will be the same on generally available systems. Furthermore, some measurements may have been estimated through extrapolation. Actual results may vary. Users of this document should verify the applicable data for their specific environment.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental.

COPYRIGHT LICENSE:

This information contains sample application programs in source language, which illustrate programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to IBM, for the purposes of developing, using, marketing or distributing application programs conforming to the application programming interface for the operating platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs.

Trademarks

Preface

This IBM® Redbooks® *publication* introduces the System z® New Application License Charge (zNALC) pricing structure and examples of zNALC workload scenarios which is the focus of this book. This book describes the products that can be run on a zNALC LPAR and reasons why one would consider such an implementation.

This book provides an introduction to the WebSphere Application Server Liberty Profile, its ability to host applications within a CICS environment, and how it will interact with CICS applications and resources. In addition, it describes the security technologies that are available to applications that are hosted within a WebSphere Application Server Liberty Profile in CICS.

This book describes how to implement the modernization of presentation in CICS with CICS Liberty and share scenarios to develop CICS Liberty applications to gain the benefit from IBM CICS Transaction Server for z/OS Value Unit Edition.

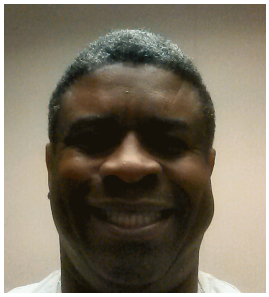
This book discusses some general considerations when using mobile devices to interact with CICS applications and explains specific CICS technologies for connecting mobile devices using IBM CICS Transaction Server for z/OS Value Unit Edition.

This book introduces the concept of decision management and describes how IBM Operational Decision Manager for z/OS can run inside the CICS Transaction Server for z/OS to provide decision management for CICS COBOL and PL/I applications.

This book discusses the CICS Transaction Server for z/OS (CICS TS) Feature Pack for Modern Batch. This capability can be installed into CICS TS V4.2 or higher and enables the WebSphere Batch Environment to schedule and manage batch applications in CICS.

Authors

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

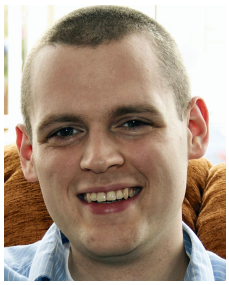


Rufus Credle is a Certified Consulting IT Specialist at the ITSO, Raleigh Center. In his role as project leader, and information developer, he conducts residencies and develops IBM Redbooks, Redpapers™, and Solution Guides. Subjects include network operating systems, enterprise resource planning (ERP) solutions, voice technology, high availability, clustering solutions, web application servers, pervasive computing, IBM and OEM e-business applications, WebSphere Commerce, IBM MQ, IBM CICS, IBM industry technology, System x®, and IBM BladeCenter®. Rufus' various positions during his IBM career include assignments in administration and asset management, systems engineering, sales and marketing, and IT services. He has a BS degree in Business Management from Saint Augustine's College. Rufus has been employed at IBM for 34 years.

George Burgess is a Software Engineer in CICS Development at IBM Hursley Park. Prior to this he spent three years as a CICS Transaction Server on z/OS Subject Matter Expert for the Peoples Republic of China. He has 29 years of experience as an Application Programmer, Systems Programmer, CICS Developer and OMEGAMON® XE for CICS Developer. His areas of expertise include Common Business Oriented Language (COBOL), CICS, DB2®, Java, IBM MQ, IMS DL/1, VSAM, JCL, z/OS, and OMEGAMON.

Paul Cooper has worked in the CICS Development organization at IBM Hursley Park for over 15 years. In that time he has helped develop the Java technology in CICS, Web Services in CICS, Mobile support for CICS, and the CICS Cloud infrastructure.

Mark Hiscock is the Operational Decision Management development team lead for z/OS based in Hursley, UK. He joined IBM in 1999 and holds a first class degree in Computer Science from the University of Portsmouth. He has over 10 years experience working in mainframe development on products such as: ODM, MQ, CICS, DB2, Message Broker and WebSphere Application Server.



Mark Hollands is a Software Engineer in the CICS Development organization. He has two years experience in developing CICS Explorer plug-ins for a number of the CICS Tools, including CICS Configuration Manager and CICS Deployment Assistant. After graduating with a degree in Computer Science, Mark joined IBM as a Software Developer working on WebSphere Voice Response, before transferring to CICS Tools Development.

Mitch Johnson is currently working as a Subject Matter Expert for MQ, ODM and CICS in Advance Technical Skills in the United States. Prior to this, Mitch was a consultant in IBM Software Services for WebSphere where he was a Subject Matter Expert for WebSphere Application Sever on z/OS. He has worked on five previous IBM Redbooks publications and Redpapers that primarily dealt with connectivity to z/OS resources from WebSphere Application Server and other resources. His areas of expertise include CICS, DB2, IMS, MQ, z/OS, Java, JEE Connectors and security.

Subhajit Maitra is an Senior IT Specialist and member of the IBM North America System Z WebSphere Technical Sales team. His expertise includes IBM Operational Decision Manager (ODM), IBM Integration Bus (IIB) and IBM MQ on System Z. Subhajit is also a IBM Global Technical Ambassador for Central and Eastern Europe (CEE), helping customers in that region implement business critical solutions on Z. His 21 year career in information technology includes roles as a developer, designer, and architect in the healthcare, financial services and government industries. Prior to this publication, Subhajit worked with the ITSO in building and delivering workshops worldwide and as an author of previously published Redbooks®. He holds a master's degree in computer science from Jadavpur University, in Kolkata, India and is an IBM zChampion.

Bei Chun Zhou is Software Engineer working within the IBM CICS Transaction Server for z/OS service team. He graduated from Tsinghua University and obtained a master degree from Chinese Academy of Sciences. Since then, he has worked at IBM for four years as CICS L3 support and is now the lead of China CICS L3 team. His expertise is mainly in CICS TS. He has rich experience to support CICS customers for problem solving, system upgrade, health check, performance tuning and new feature enablement.

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

Tamikia Barrow, Debbie Willmschen
International Technical Support Organization, Raleigh Center

Amanda Ballard-Stuart, Staff Administration Assistant
IBM Hursley

Matthew Wilson, Ivan Hargreaves, Daniel Millwood, Michael Wang, Ian Mitchell, - IBM Development
IBM Hursley

Andy Bates, Adrian Kyte, Geoffrey Pirie - IBM Marketing
IBM Hursley

Now you can become a published author, too!

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

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

ibm.com/redbooks/residencies.html

Comments welcome

Your comments are important to us!

We want our 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

Part 1

Considering new workloads on the mainframe

In this part we discuss mainframe workload pricing and help you to understand what will qualify for System z® New Application License Charge and Value Unit Edition.

1

Mainframe workload pricing

This chapter introduces the System z® New Application License Charge pricing structure and examples of zNALC workload scenarios which is the focus of this book.

Further information about the available pricing options can be found at:

http://www-03.ibm.com/systems/z/resources/swprice

We discuss the following topics in this chapter:

1.1 Introduction

System z New Application Licence Charges(zNALC) gives a customer the opportunity to run a new workload on System z at a fixed cost provided that the workload qualifies as net new application workload. For example, Java language business applications running in IBM CICS Transaction Server for z/OS Value Unit Edition or WebSphere® Application Server.

Currently, the other IBM software products that are eligible to run on zNALC enabled LPAR are DB2® for z/OS®, IMS™ Database, IMS™ Transaction Manager and WebSphere® MQ for z/OS®.

IBM CICS Transaction Server for z/OS Value Unit Edition requires that it run on a zNALC enabled Logical Partition (LPAR).

Running CICS Transaction Server for z/OS V5.2 in a non-zNALC installation using the Monthly Licence Charge (MLC) pricing option is normally considered an Operating Expenditure (OpEx). This choice is made because the expenses are charged on a monthly basis and are directly related to the CPU usage for the month.

Buying IBM CICS Transaction Server for z/OS Value Unit Edition and running it on a zNALC LPAR is a one time charge and is categorized as Capital Expenditure (CapEx). The same principle applies to other software required to support the application such as DB2® for z/OS® Value Unit Edition, IMS™ Database Value Unit Edition or WebSphere® MQ for z/OS® Value Unit Edition.

1.2 Available facilities

If the customer already has a fully licensed environment running CICS, DB2, IMS or WebSphere MQ, the new Java applications can use any or all of the following facilities:

- CICS®
  - Distributed Program Link to link to the existing applications in running MLC environments
  - Function ship requests for
    - remote VSAM files
    - remote STARTs
    - remote transient data queues
    - remote IMS™ Databases
- DB2 for z/OS
  - Access existing DB2 databases
- IMS
  - Access existing IMS databases
- IBM WebSphere MQ for z/OS
  - Access local and remote MQ queues
  - Accept incoming work requests and trigger CICS transactions

For full details about the zNALC option, visit the following link:

http://www-03.ibm.com/systems/z/resources/swprice/mlc/znalc.htm

1.3 Did you know?

When using the zNALC pricing option, it is not necessary to have a zIIP processor available to the Logical Partition as offloading the Java workload to the zIIP processor will not affect the overall cost. This also means that the workload running on non zNALC LPARS needing to use the zIIP processor will not be affected by any work requiring a zIIP processor on the zNALC LPAR.

1.4 Business value

The Value Unit Edition (VUE) family of products provides a choice for how to budget for new projects. Previously, all new projects would primarily result in an increase in operational expenditure. Deploying into a VUE environment on a zNALC LPAR enables a proportion of the project cost to be a capital expenditure. In organizations where strict budgetary controls are in place, having the flexibility to choose between these two budgets can help to get new projects financed and delivered.

1.5 Solution overview

Commission a System z New Application Licence Charge Logical Partition on System z, install IBM CICS Transaction Server for z/OS Value Unit Edition V5 and suitable database software to support the workload that will execute on the LPAR.

1.6 Usage scenarios

Use the family of Value Unit Edition products to provide an environment or environments where a net new Java based workload can be developed and deployed to:

1. provide support for a mobile banking workload
2. create a modern interface to existing business applications
3. consolidate multiple distributed systems onto a single System z platform

1.7 Related information

List relevant material from IBM Redbooks or other IBM documents, as well as useful product/solution web pages

For more information, see the following documents:

- IBM Redbooks: New Workloads in CICS, SG24-8225

  http://www.redbooks.ibm.com/abstracts/sg248225html
- CICS Transaction Server for z/OS Value Unit Edition product page

  http://www-03.ibm.com/software/products/en/cics-ts-vue
- IBM Offering Information page (announcement letters and sales manuals):

  http://www.ibm.com/common/ssi/index.wss?request_locale=en

2

Understanding what will qualify for zNALC and VUE

This chapter describes the products that can be run on a zNALC LPAR and reasons why one would consider such an implementation.

In general the basic qualifying requirements are:

- Must be net new and Java based hence the focus on Liberty and JVM server applications
- Qualifying application packages. These are primarily vendor packages which are available on distributed platforms as well as System z. Each package needs to be validated separately.

In most cases a customer wishing to rehost a Java language application from a distributed platform to an existing System z platform would be eligible for zNALC enabled products, again this is subject to validation.

Some reasons for rehosting are:

- Cost management which can take many forms for example:
  - Reduce architectural complexity
  - Network simplification
  - Skills reuse especially in those areas where skills are becoming more difficult / expensive to find or retain
  - Reuse of resources such as customer data residing on System z be it in VSAM files, DB2 or IMS databases
  - Reduce operational expense of distributed servers, for example:
    - air conditioning and cooling
    - electricity
    - water
    - software licences
    - hardware maintenance
    - machine and network maintenance
    - staff salaries
- Co-location of application and data will deliver a reduced network latency
- Easier problem determination
- Consolidation of applications onto a single platform

Other reasons for rehosting are:

- Project separation, new Java development can be developed and delivered using a potentially more cost effective solution and reuse existing resources within the organization
- VUE allows you to deploy new Java workload without affecting existing zIIP resource allocations and associated workloads
- The cost of the zNALC LPAR and related software is fixed price
- Leveraging of existing resources to support the new workload

The following products are eligible to run on a zNALC enabled LPAR:

- IBM CICS Transaction Server for z/OS Value Unit Edition

  Eligible Workload is defined as net new Java workload that executes within the IBM CICS Transaction Server for z/OS Value Unit Edition Java Virtual Machine (JVM) server environment, on condition that the workload is qualified and approved through the zNALC qualification process.

- IBM DB2 Value Unit Edition

  The terms of DB2 Value Unit Edition limit its use to support net new applications or workloads on IBM System z servers that meet zNALC LPAR qualifications. This software would be used to support the workload running in CICS.

- IBM IMS Database Value Unit Edition

  Supports a net new workload that includes IMS Database Value Unit Edition, where IMS Database Value Unit Edition has been qualified and approved through the System z New Application License Charge (zNALC) qualification process as eligible to run in a zNALC LPAR(s). This software would be used to support the workload running in CICS.

- IBM MQ VUE

  Support for new workloads that run on qualified System z New Application License Charge (zNALC) logical partitions (LPARs). This software would be used to support the workload running in CICS.

Part 2

Liberty and CICS

This part introduces the concepts and features of the WebSphere Application Server Liberty Profile within a CICS Transaction Server for z/OS environment.

Discussed in detail are two key benefits of the CICS Liberty environment, starting with the modernization of the presentation layer of CICS applications then explaining how consolidation and co-location of presentation and data can be beneficial in a CICS infrastructure.

The following chapters are included:

3

Overview of WebSphere Application Server Liberty Profile in CICS

This chapter provides an introduction to the WebSphere Application Server Liberty Profile, its ability to host applications within a CICS environment, and how it will interact with CICS applications and resources. The chapter will then describe the security technologies that are available to applications that are hosted within a WebSphere Application Server Liberty Profile in CICS.

This chapter covers the following topics:

3.1 Evolving application servers

As software development methodologies develop over time to deliver more functionality to customers quickly and reliably, being able to incorporate new technologies into existing environments has never been more important. The ability for an application server like CICS to support these continuously changing demands is essential to deliver these new services. The challenges on application servers include:

- The ability to rapidly deploy (and re-deploy) application artifacts as part of a DevOps continuous integration process.
- Software should be modular and easily assembled. This allows applications to be rapidly composed from existing modules, and easily deployed into the runtime environment.
- Modern application patterns, such as RESTful web services and responsive UI are rapidly becoming more popular. Application servers must be able to adopt these new programming models.

The WebSphere Application Server Liberty Profile (*Liberty*) is lightweight, easy to install, and use to develop and deploy applications. It therefore provides a convenient and capable platform for developing and testing your web and OSGi applications. Liberty is built using OSGi technology and concepts. The fit-for-purpose nature of the runtime relies on the dynamic behavior inherent in the OSGi framework and service registry. As applications (bundles) are installed or uninstalled from the framework, their services are automatically added or removed from the service registry. The result is a dynamic, composable run time that can be provisioned with only what your application requires and responds dynamically to configuration changes as your application evolves.

CICS Transaction Server for z/OS (*CICS*) V5.1 added support for Liberty to run within a JVM server in CICS, and this has been extended in CICS Transaction Server for z/OS V5.2 adding more supported features available for use in Liberty applications running within CICS.

3.2 Overview

Liberty is a simple, lightweight development and application runtime environment that offers these benefits:

- *Simple to configure*: Configuration is read from a single XML file (`server.xml`). CICS extends this simplicity further by adding the ability to automatically generate this XML file.
- *Dynamic and flexible*: The Liberty profile server runtime loads only what your application needs and constructs the run time in response to configuration changes.
- *Extensible:* The Liberty profile server provides support for user and product extensions, which can make use of system programming interfaces (SPIs) to extend the runtime. An example of this is the additional features that CICS provide in CICS Liberty (discussed in 3.4, “Liberty in CICS” on page 14).

3.3 Strengths

Liberty offers great advantages when used as both a development runtime and production runtime within CICS. Liberty is both lightweight and capable, particularly when considering the ability of third parties to extend and enhance the available features. A good example of this is the CICS and z/OS features that have been added in CICS Liberty.

3.3.1 Simple configuration

The server configuration, from the server administrator's perspective, is only a single `server.xml` file that contains all needed information. The WebSphere Application Server Liberty Profile for z/OS V8.5.5 (version included in CICS TS V5) configuration file has many optional parameters used for specific scenarios. You can find them in the IBM Knowledge Center:

http://www-01.ibm.com/support/knowledgecenter/SSD28V_8.5.5/com.ibm.websphere.wlp.core.doc/autodita/rwlp_metatype_core.html

3.3.2 Runtime composition with features and services

The composable nature of Liberty is based on the concept of features. A *feature* is a unit of functionality. Features can overlap, and they can include other features.

CICS Liberty consists of a JVM server that hosts the Liberty kernel, and any number of optional features. The feature code and most of the kernel code runs as OSGi bundles within an OSGi framework. Features provide the programming models and services required by applications. You can choose which optional features should be enabled according to your application requirements.

Note: It is only possible for one CICS Liberty to run per CICS region with security enabled. If multiple Liberty profile servers need to run within a single CICS region, security must be disabled within the JVMProfile for the JVM server. By default security is enabled so if the required flag is not set within the JVMProfile only one CICS Liberty instance will start.

3.3.3 Developer focus

With Liberty you can do rapid development and deployment to meet with modern development trends. Liberty offers the following advantages for developers:

- **Fast and no-cost download for developer's workstation**

  Liberty profile server is fast and no cost for developer workstation use. It can be downloaded and installed from Eclipse.org or WASdev.net.

- **Rapid development and deployment**

  You deploy an application in Liberty profile server by either dropping the application into server's `dropins` directory, or by adding an application entry to the server configuration (`server.xml`) file.

  In addition to this, within CICS Liberty you can package an application, for example an Enterprise Bundle Archive (EBA) in a CICS Bundle and deploy this bundle within CICS. This is the suggested deployment method for applications in CICS Liberty.

- **Easy extensibility for custom features and third-party components**

  Liberty supports direct extension of the runtime using product extensions. A product extension allows custom content to be added to a Liberty installation in a way that avoids conflicts with the base content of the product and with other product extensions.

3.4 Liberty in CICS

As previously mentioned, CICS runs Liberty within a CICS JVM server. Figure 3-1 shows the basic architecture of how a Liberty profile server is hosted within a CICS region:

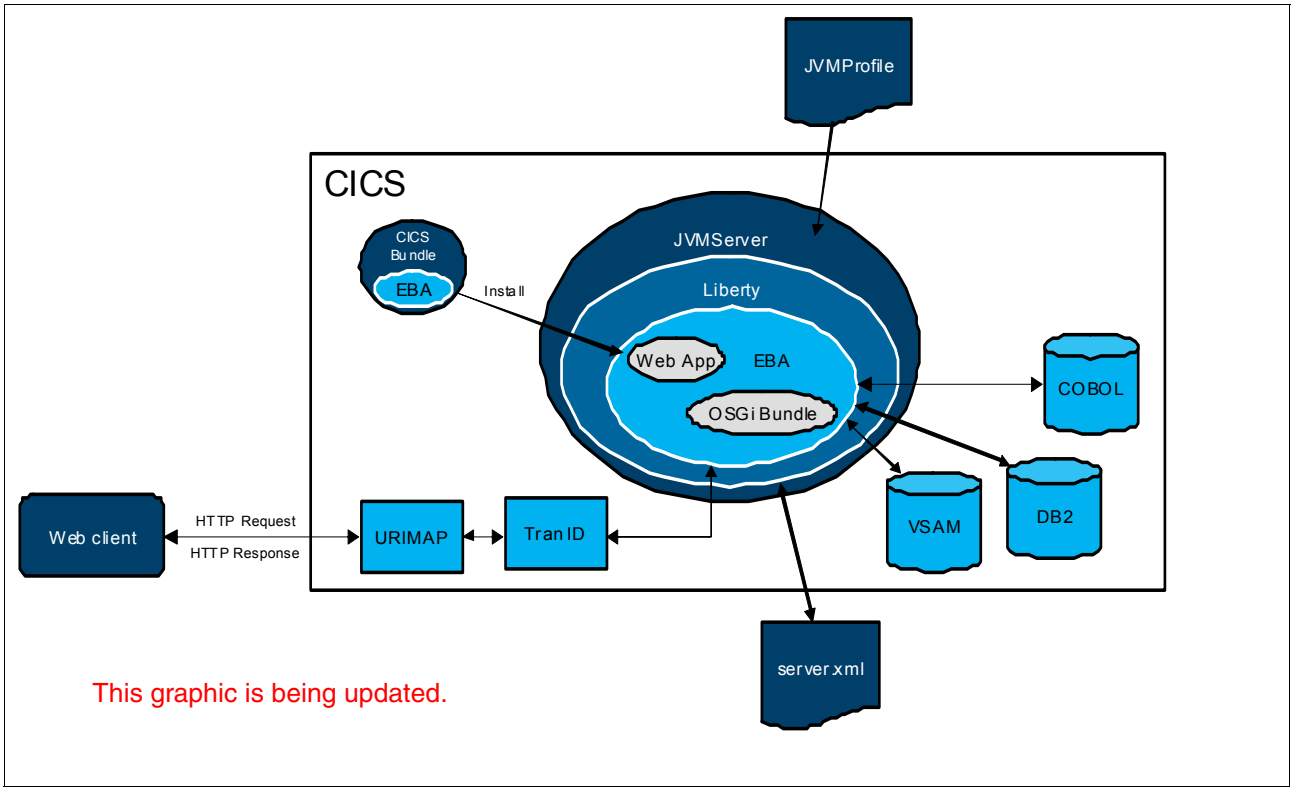


Figure 3-1 Architecture of Liberty in a CICS JVM server

As shown in Figure 3-1, the Liberty profile server is hosted in a CICS JVM server which is defined in a JVMProfile. This example shows how a CICS bundle, containing an Enterprise Bundle Archive (EBA) is placed inside CICS and then installed into the Liberty environment. This EBA, containing a Web application is then able to access CICS resources, for example DB2 or VSAM data sets, or other CICS COBOL applications.

This example demonstrates using a URIMAP as an entry point into CICS Liberty from a web client allowing context switching of Transcation ID or User ID. Liberty itself listens on a port and handles HTTP traffic, with the ability to define transport security (discussed in 3.5.2, “Security overview” on page 16). If context switching is not required, a web client could simply connect directly to Liberty.

Figure 3-1 also shows that the `server.xml` file defining the configuration of the Liberty profile server is also available, although this can be automatically generated by CICS. Security aspects of this architecture will be discussed further later in this chapter.

CICS TS V5.2 supports a subset of the total features of the full Liberty profile. These supported features include, but are not limited to the following:

- JavaServer Faces (JSF) 2.0
- JavaServer Pages (JSP) 2.2
- Java Servlet 3.0
- Java API for RESTful Web Services (JAX-RS) 1.1
- Java API for XML Web Services (JAX-WS) 2.2

- JavaScript Object Notation (JSON4J) 1.0
- Secure Socket Layer (SSL) 1.0
- Web Application Bundles (WAB) 1.0
- Java Database Connectivity (JDBC) 4.0

A complete list of supported Liberty features can be found in the IBM CICS Knowledge Center:

http://www.ibm.com/support/knowledgecenter/SSGMCP_5.2.0/com.ibm.cics.ts.java.doc/topics/liberty_features.html

CICS Liberty does include all features of the full Liberty profile (in CICS TS V5.2 this is the WebSphere Application Server Liberty Profile for z/OS V8.5.5.1), however the features that appear in the IBM Knowledge Center are the subset that are currently supported. You may chose to include other features in your applications, but this is not supported.

In addition to the Liberty features supported in CICS, the following are also provided:

- CICS Core (*cicsts:core-1.0*) -

  Provides core CICS features and Java Transaction API (JTA) 1.0

- CICS JDBC (*cicsts:jdbc-1.0*) -

  Provides support for applications to access a local CICS DB2 database using JDBC.

- CICS Liberty security (*cicsts:security-1.0*) -

  Provides integration of Liberty security with CICS security, including propagation of thread identity. This feature includes the *zosSecurity-1.0* feature.

3.4.1 Integration with CICS Transaction Server

The two subsequent chapters in this part will outline the advantages of running Liberty applications in a CICS environment. The focus is to allow new applications, whether they be new presentation layers to interact with current CICS applications, or applications that will use resources managed by CICS, to benefit from co-location with the CICS environment. Both use cases focus on the ease of creating new applications that can integrate with CICS.

The two main ways to interact with CICS are to call an existing application, for example a CICS COBOL application running on an existing non-zNALC LPAR, or new applications, to take advantage of the JCICS API to call CICS functionality. There are advantages to each approach which mainly depend on what you are trying to achieve. If the main objective is to modernize your application presentation layer, for example implementing a Java API for RESTful Web Services (JAX-RS) interface allowing interaction with mobile applications, the appropriate choice maybe to link to existing applications. Similarly, if a new application is being developed from scratch, you can implement all functionality from Java within Liberty. This new application, as well as calling the JCICS API directly to access resources could also call existing applications hosted on another LPAR.

Before these options are explored further in the rest of this part, the rest of this chapter will outline what security options are available within CICS Liberty and some security considerations that are important to highlight.

Note: While it is possible for an application hosted in CICS Liberty to link to a CICS COBOL, or other type of application in CICS, the ability to link to a CICS Liberty application is not supported. If this is required, a CICS COBOL application could make a HTTP call to a CICS Liberty application.

3.5 Security

This section will provide an introduction to the security features that are available for CICS Liberty applications.

The security features mentioned below may, and probably will, form part of a wider security infrastructure. It is likely that further levels of security may be required for external users to access applications hosted in CICS Liberty, for example integrating with IBM Worklight. This chapter will only cover the security features specific to CICS Liberty applications, however IBM Worklight will be discussed in 6.5, “IBM Worklight and CICS” on page 50.

3.5.1 Introduction to Security with Liberty in CICS

CICS Liberty includes *appSecurity-2.0* enabling security for web applications when the *servlet-3.0* feature is present. This compliments SSL support (through the *ssl-1.0* feature), which enables SSL (including TLS) connections using HTTPS (discussed in 3.5.2, “Security overview” on page 16).

For CICS specific security options, CICS has an additional feature, *cicsts:security-1.0*, which has integrated *zosSecurity-1.0* features with CICS security options, for example propagation of user identity. To use the *cicsts:security-1.0* feature, the angel process, available in CICS TS V5.2, must be running. This is used for accessing z/OS authorized services and can utilize SAF security frameworks. The angel process is the suggested authentication and authorization method for CICS Liberty applications. See section 3.5.3, “The Liberty server angel process” on page 18 for more details on the angel process.

Furthermore, CICS Transaction Server for z/OS V5.2 includes improved performance through exploiting the Liberty authentication cache. When a user is authenticated a new *Subject* object is created storing all authorization information, including roles. This *Subject* object is stored in the Liberty authentication cache, with a configurable cache expiry time, preventing multiple authentication and authorization requests for the same user.

3.5.2 Security overview

Before discussing the security options that are available in Liberty in CICS, the following are some key terms that are essential when understanding security:

- **Transport / Communication**

  Transport, or communication, refers to the mechanism that is used for data to travel between two places, for example from a client (for example a web browser on a computer) to a server (for example an application in Liberty running within CICS). When referred to in this chapter we are concerned with how to secure the data that is sent between two sources.

  Communications in CICS Liberty are secured with the Secure Sockets Layer (SSL) protocol. The SSL protocol provides transport layer security including authenticity, data signing, and data encryption to ensure a secure connection between a client and server. SSL in CICS Liberty includes TLS v1.2 support required for some security standards and the protocol used can be configured in `server.xml`, within the `<ssl>` element using the `sslProtocol` attribute.

  You can configure a CICS Liberty JVM server to use SSL for data encryption, and optionally authenticate with the server using a client certificate. Client certificates can be stored in a Java key store or in a SAF keyring.

Note: Application Transparent TLS (AT-TLS) is not supported within CICS Liberty

- **Authentication**

  Authentication confirms that an entity (for example a user) that is attempting to access a resource (for example an application hosted in Liberty) is a valid entity. Typically, this entity will provide a username and password when attempting to gain access. This username and password, or possibly a client certificate, is used to authenticate the entity.

  CICS Liberty includes many different options to aid in authentication. For an introduction on Liberty authentication visit this website:

  http://www.ibm.com/support/knowledgecenter/SSD28V_8.5.5/com.ibm.websphere.wlp.core.doc/ae/cwlp_authentication.html

  CICS Liberty supports a number of the authentication options described on that web page, with some additions. The basic Liberty authentication features supported in CICS Liberty are:

  – SSL client authentication
  – Form logon
  – Lightweight Third-Party Authentication
  – Custom user registry
  – Trust Association Interceptor

  In addition, the angel process discussed in 3.5.3, “The Liberty server angel process” on page 18 allows authentication using z/OS security services (SAF) which, along with using the angel process for authorization, is required for applications that interact with other CICS processes and are contained within CICS bundles.

- **Authorization**

  Authorization determines whether a given entity has been granted the correct privileges in order to access a resource. This can be used, for example, in protecting certain areas of a website that may only be available to certain authorized users.

  For an introduction on Liberty authorization, visit this website:

  http://www.ibm.com/support/knowledgecenter/SSD28V_8.5.5/com.ibm.websphere.wlp.core.doc/ae/cwlp_authorization.html

  In addition to the <application-bnd> configuration options available in Liberty, CICS Liberty includes the following addition authorization options:

  – Roles (defined as SAF EJBROLE) - discussed in 3.5.4, “SAF roles” on page 19
  – The angel process authorization - discussed in 3.5.3, “The Liberty server angel process” on page 18
  – ThreadIdentityService (can push Subject credential onto CICS task)

- **Role**

  Typically, authorization is aided by the use of roles. An entity can be assigned one or more roles and then a resource can be authorized to be used by a role. If the entity is a member of a role, and a role is authorized to access a resource, access is granted to the entity.

- **Subject**

  A subject is a representation of a given entity, for example, as mentioned before, a user. When a user is authenticated, their authorization information (for example Role membership) is retrieved and stored in a Subject object within Liberty authentication cache. Any subsequent authentications will result in this Subject object being retrieved from the cache, improving performance. The Liberty authentication cache has a configurable expiry time for the Subject object, with a default of ten minutes.

3.5.3 The Liberty server angel process

As discussed in 3.5.1, “Introduction to Security with Liberty in CICS” on page 16, CICS Transaction Server for z/OS V5.2 added support for the angel authentication and authorization process to CICS Liberty, and is now the default solution for authentication and authorization in CICS Liberty.

The angel process is lightweight, doing very little CPU-consuming work after establishing some control blocks. Only one angel process is required per z/OS operating system image (LogicalPARtition, *LPAR*), and this process has no configuration files and uses no TCP/IP ports.

The angel process will use the SAF user registry for all authentication requests by default if the CICS Liberty security feature is included. This will then allow a Liberty based application hosted in CICS, and other CICS processes that are linked to from CICS Liberty, to use the same user identity. Figure 3-2 shows how authentication and authorization requests are routed through the angel process if the CICS TS Security feature is present:

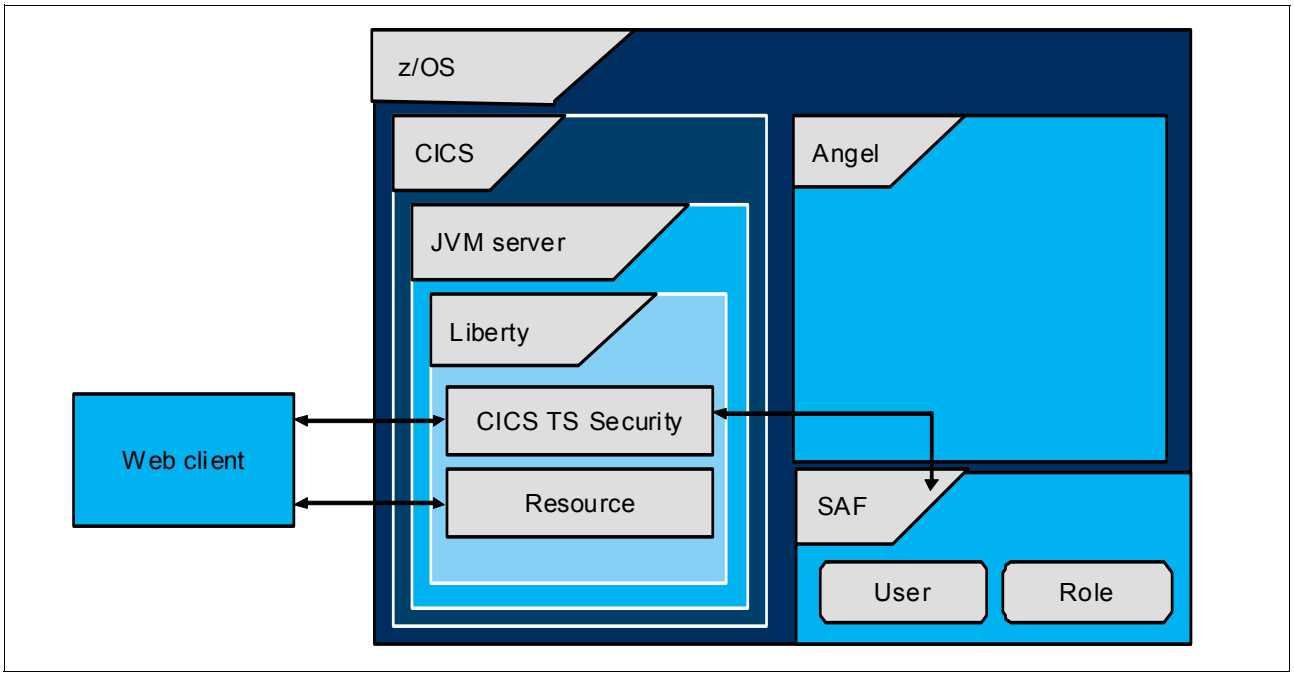


Figure 3-2 Process diagram for authentication and authorization with the angel process

Figure 3-2 shows an example of a web client connecting to a Liberty profile server running within CICS and requesting access to a resource. The CICS TS Security feature (*cicsts:secuity-1.0*) has been configured in the Liberty profile server. When the web client requests access to the resource the request is routed through the angel process onto SAF where the user credentials are authenticated and checked against the requested resource. The resource access could be controlled using roles and the user would have to be part of the role to be granted access. Once the user is authenticated and authorized the response to the authorization request is passed back to Liberty and if successful the resource is returned to the web client.

For more information on how to configure CICS Liberty with the angel process visit the IBM Knowledge Center:

http://www.ibm.com/support/knowledgecenter/SSGMCP_5.2.0/com.ibm.cics.ts.java.doc/topics/security_angel.html

3.5.4 SAF roles

When using the CICS Liberty security feature you can include a configuration element `<safAuthorization>` in `server.xml` to enable the use of SAF roles (EJBROLE). If this is present any roles defined in `server.xml` are ignored and role membership is defined and granted using SAF roles. An EJBROLE can be defined using SAF and then membership of that role is granted to users defined in the SAF registry, allowing access and permissions to SAF authorized resources for example CICS bundles containing CICS Liberty applications.

For more information on how to configure roles within CICS Liberty see the IBM Knowledge Center:

http://www.ibm.com/support/knowledgecenter/SSGMCP_5.2.0/com.ibm.cics.ts.java.doc/topics/jee_app_role.html

3.5.5 Scenarios

The following chapters will detail the main scenarios for creating new or migrating existing applications to CICS Liberty, explaining the advantages and possibilities that are available.

4

Modernization of presentation - BZ

This chapter describes how to implement the modernization of presentation in CICS with CICS Liberty. We look at scenarios to develop CICS Liberty applications to gain the benefit of IBM CICS Transaction Server for z/OS Value Unit Edition, and the features that CICS Liberty provides to run the presentation logic. We also introduce a toolkit to migrate an existing Java application from other platforms into CICS Liberty.

This chapter covers the following topics:

4.1 CICS Liberty scenarios

Before CICS Liberty, CICS supported the 3270 screen and web as the presentation interfaces with CICS Basic Mapping Support (BMS), CICS Web Support (CWS) or CICS Dynamic Scripting Feature Pack.

The 3270 screen is a traditional mainframe interface which is not as modern as web interface or mobile interface. There are many requirements from CICS customers to replace 3270 screen with web browser and RESTful clients.

CICS Web Support can support web application, but the application programmers need to use CICS Web APIs to analyze the HTTP data and to assemble the HTTP response. Web applications are not typically developed in this way any more.

The Dynamic Scripting feature pack originated from CICS Transaction Server for z/OS Version 4 release 1 supports the use of PHP and Groovy to develop situational applications. In CICS Transaction Server for z/OS Version 5 release 1, the Dynamic Scripting feature pack runs in CICS Liberty and it only supports PHP. We will introduce more in section 4.2, “CICS Liberty features for the presentation layer” on page 24.

There are other third party presentation technologies used to connect to CICS. Some CICS customers are eager to modernize these client connectors with Web Front End (servlets).

CICS Liberty is the preferred way to develop web applications in CICS or to handle RESTful request. All Java workloads deployed using CICS Liberty are candidates for approval for IBM CICS Transaction Server for z/OS Value Unit Edition.

There are two typical scenarios for the customers to use CICS Liberty.

4.1.1 Scenario one

The first scenario is that a customer wants to develop a pure Java solution either for a new business requirement or as a replacement of existing solution. The Java solution includes presentation logic, business logic and data access logic. In this case, everything can be deployed and run in a zNALC enabled LPAR.

In Figure 4-1 on page 23, we show a general architecture for this scenario. We have a CICS region running in zNALC enabled LPAR. In the region, we set up a JVM server for CICS Liberty. We develop an application with presentation, business logic and data access. In the presentation layer, we can use the features that CICS Liberty provides. For more information, please refer to section 4.2, “CICS Liberty features for the presentation layer” on page 24.

To develop and deploy a CICS application, the CICS Explorer® SDK is a preferred tool which can be freely downloaded from the IBM website and be installed in an Eclipse Integrated Development Environment (IDE). The development process is similar as the process to develop a Liberty project for distributed platform. To access the data, CICS provides full JCICS API support.

After the development work is done, the Java web application, either in the form of WAR files or an EBA file, can be deployed as one or more CICS bundles in zFS. Customers can keep multi-version of the applications in CICS Liberty. If there are common Java classes to be shared by several different Java web applications, customers can deploy these classes as common OSGi bundles to zFS directories and refer to these directories from a bundleRepository elements of the server.xml of the Java web applications.

For more information on how to develop Java applications in CICS Liberty, please refer to Chapter 5, “Consolidating applications in CICS Liberty” on page 29.

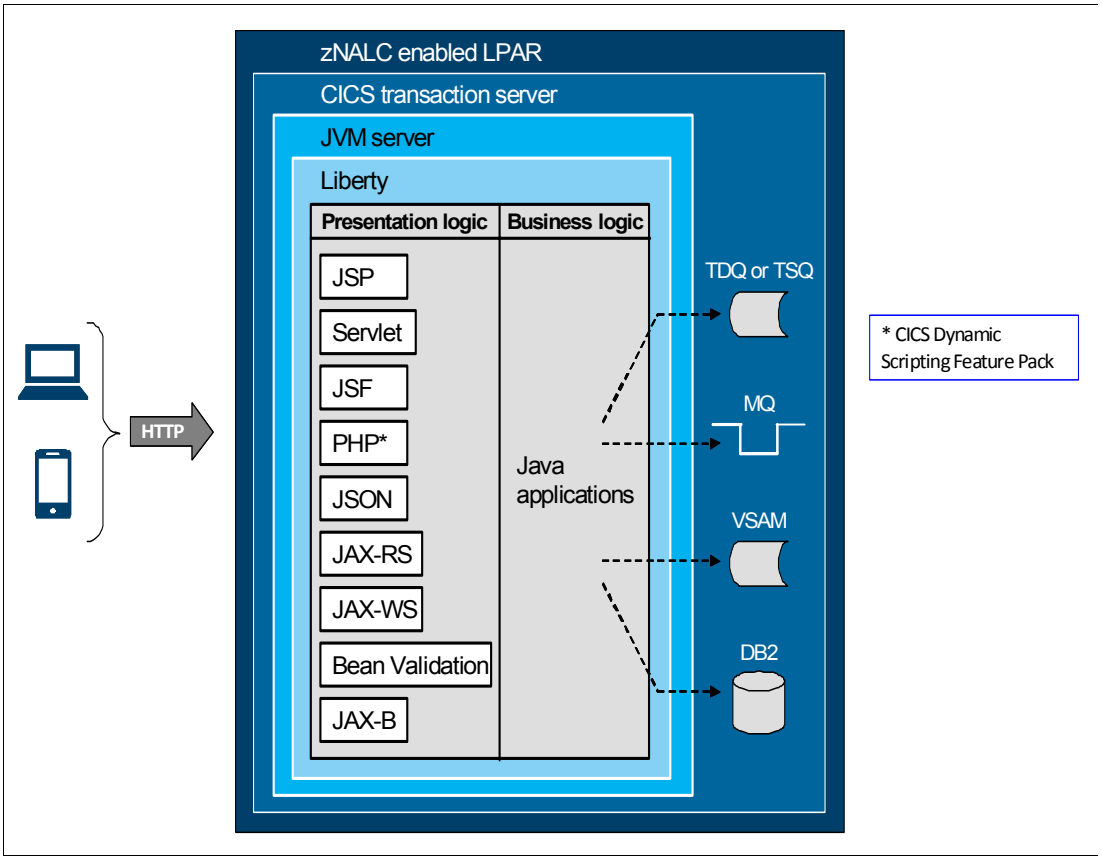


Figure 4-1 Presentation, business logic and data access all in CICS Liberty

4.1.2 Scenario two

Another scenario is that there is a need to replace old less modern presentation layer (for example, BMS) into more modern presentation layer like web or mobile. In this case, existing business logic and data access can be re-used. The customer can develop and deploy the new presentation layer in CICS Liberty which runs in zNALC enabled LPAR. The presentation layer can then link to the existing business logic which is still in standard z/OS LPAR. Figure 4-2 on page 24 shows the architecture for this scenario.

The development and deployment in this scenario is all the same as the previous scenario. The difference is that we just focus on the presentation layer. To communicate with an existing business logic in another CICS region, Dynamic Program Link (DPL) is a preferred way. CHANNEL/CONTAINER and COMMAREA are both supported to carry the data.

Many existing CICS applications are centered around the use of structured data records, which are typically stored as sequential files in the Virtual Storage Access Method (VSAM) data set or as DB2 tables. To communicate with these existing CICS application, it is frequently desirable to reuse the copybooks, which describe these existing record structures. To simplify the interactions from the Java application with the structured data required by the existing CICS applications, IBM provides utilities like JZOS and J2C to make the task easier. For more information on how to use JZOS and J2C, please refer to IBM Redbooks publication, IBM CICS and the JVM server: Developing and Deploying Java Applications,

SG24-8038-00. See Chapter 8 which you can download here:
http://www.redbooks.ibm.com/abstracts/sg248038.html?Open

.

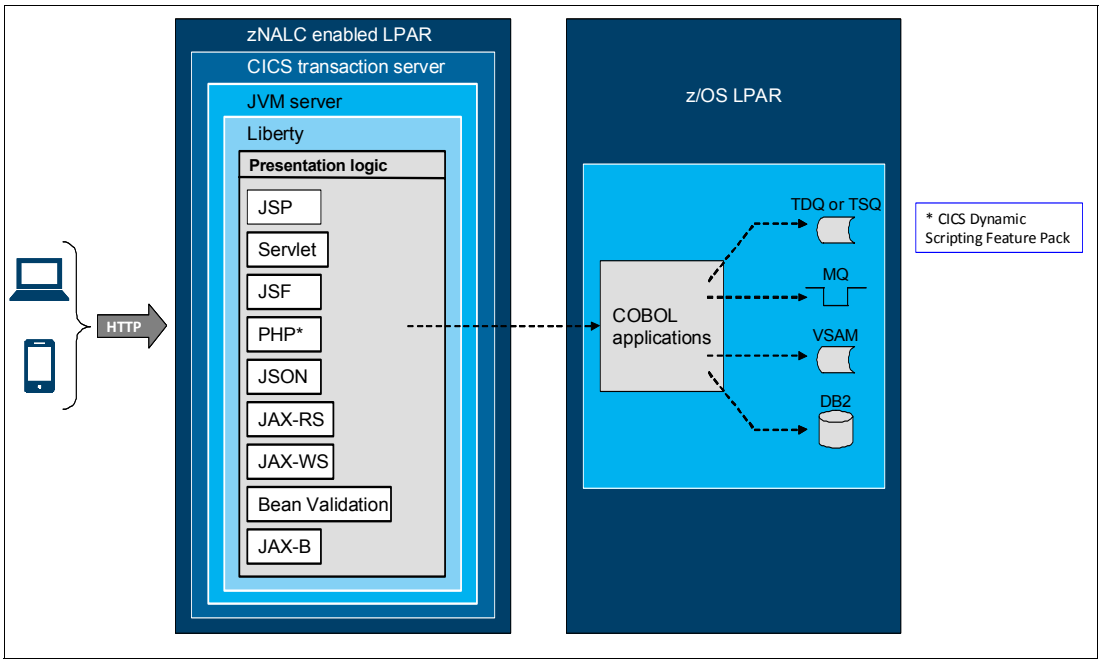


Figure 4-2 Presentation in CICS Liberty and link to existing business logic

4.2 CICS Liberty features for the presentation layer

CICS uses some of the features in the WebSphere Application Server Liberty Profile to run web applications in a JVM server. For a list of features CICS supports in version 5 release 2, please refer to the IBM Knowledge Center:

http://www-01.ibm.com/support/knowledgecenter/SSGMCP_5.2.0/com.ibm.cics.ts.java.doc/topics/liberty_features.html?lang=en

In this section, we briefly introduce the Liberty features that CICS support for the presentation layer. As mentioned, these features are from WebSphere Application Server Liberty Profile. For more information of how to use these features you can refer to IBM Redbooks publication, WebSphere Application Server Liberty Profile Guide for Developers, SG24-8076-01 which you can download here:

http://www.redbooks.ibm.com/abstracts/sg248076.html?Open

4.2.1 JavaServer Pages (JSP) 2.2

JavaServer Pages (JSP) is a widely used technology to easily create dynamic web pages based on HTML, XML, or other document types. JavaServer Pages enable the separation of the Hypertext Markup Language (HTML) code from the business logic in web pages so that HTML programmers and Java programmers can more easily collaborate in creating and maintaining pages.

CICS Liberty supports the JavaServer Pages 2.2 specification. For more information about JSP and how to develop JSP in the Liberty profile, see the following website:

http://www-01.ibm.com/support/knowledgecenter/SSAW57_8.5.5/com.ibm.websphere.nd.multiplatform.doc/ae/welc6top_jsp_files_container.html?lang=en

4.2.2 JavaServer Faces (JSF) 2.0

JavaServer Faces (JSF) simplifies the development of user interfaces for web applications by providing the following features:

- Templates to define layout
- Composite components that turn a page into a JSF UI component
- Custom logic tags
- Expression functions and validation
- Component libraries
- XHTML page development

For more information about JSF and the features it provides, see the following website:

http://www-01.ibm.com/support/knowledgecenter/SSAW57_8.5.5/com.ibm.websphere.nd.multiplatform.doc/ae/cweb_javaserver_faces.html?lang=en

4.2.3 Java Servlet 3.0

Java servlets is a widely used technology for building dynamic content for web-based applications. The servlet Java classes are used to extend the capabilities of a server, more commonly of a web server.

CICS Liberty provides support for HTTP Servlets written to the Java Servlet 3.0 specification. For more information on developing servlets, see the following website:

http://www-01.ibm.com/support/knowledgecenter/SS7K4U_8.5.5/com.ibm.websphere.zseries.doc/ae/container_dev_servlets.html?lang=en

4.2.4 JavaScript Object Notation (JSON4J) 1.0

The JavaScript Object Notation (JSON4J) library is an implementation of a set of JavaScript Object Notation (JSON) handling classes for use within Java environments.

The JSON4J library provides the following functions:

- A simple Java model for constructing and manipulating data to be rendered as the JSON implementation.
- A fast transform for XML to JSON conversion, for situations where conversion from an XML reply from a web service into a JSON structure is wanted for easy use in Asynchronous JavaScript and XML (Ajax) applications.
- A JSON string and stream parser that can generate the corresponding JSONObject, which represents that JSON structure in Java. You can then change that JSONObject and serialize the changes back to the JSON implementation.

CICS Liberty provides access to the JSON4J library that provides a set of JSON handling classes for Java environments. For more information about JSON4J, see the following website:

http://www-01.ibm.com/support/knowledgecenter/SSAW57_8.5.5/com.ibm.websphere.wdt.doc/topics/cjson4j.htm?cp=SSAW57_8.5.5&lang=en

JSON4J can be used to connect mobile devices into CICS. Please refer to Chapter 6, “Overview of connecting mobile devices to CICS” on page 43.

4.2.5 Java API for RESTful Web Services (JAX-RS)

Java API for RESTful Web Services (JAX-RS) is a technology to develop services that follow Representational State Transfer (REST) principles. RESTful services are based on manipulating resources. Resources can contain static or dynamically updated data. By identifying the resources in your application, you can make the service more useful and easier to develop.

CICS Liberty provides support for the Java API for RESTful Web Services on the Liberty profile. For more information about JAX-RS, see the following website:

http://www-01.ibm.com/support/knowledgecenter/SSAW57_8.5.5/com.ibm.websphere.nd.doc/ae/welc6tech_wbs_rest_dev.html?lang=en

JAX-RS can also be used to connect the mobile into CICS. Please refer to Chapter 6, “Overview of connecting mobile devices to CICS” on page 43.

4.2.6 Java API for XML Web Services (JAX-WS) 2.2

Java API for XML-Based Web Services (JAX-WS) is the next generation web services programming model. Using JAX-WS, development of web services and clients is simplified with more platform independence for Java applications by the use of dynamic proxies and Java annotations.

CICS Liberty provides support for JAX-WS 2.2 based SOAP web services. For more information about JAX-WS, see the following website:

http://www-01.ibm.com/support/knowledgecenter/SSAW57_8.5.5/com.ibm.websphere.nd.doc/ae/cwbs_jaxws.html?lang=en

JAX-WS can also be used to connect the mobile into CICS. Please refer to Chapter 6, “Overview of connecting mobile devices to CICS” on page 43

4.2.7 Java Architecture for XML Binding (JAXB) 2.2

Java Architecture for XML Binding (JAXB) is a Java technology that provides an easy and convenient way to map Java classes and XML schema for simplified development of web services. JAXB leverages the flexibility of platform-neutral XML data in Java applications to bind XML schema to Java applications without requiring extensive knowledge of XML programming. JAXB provides the xjc schema compiler tool and the schemagen schema generator tool to transform between XML schema and Java classes.

CICS provides JAXB support to map between Java classes and XML representations. For more information about JAXB, see the following website:

http://www-01.ibm.com/support/knowledgecenter/SSAW57_8.5.5/com.ibm.websphere.nd.multiplatform.doc/ae/cwbs_jaxb.html?cp=SSAW57_8.5.5&lang=en

JAXB can also be used to connect the mobile into CICS. Please refer to Chapter 6, “Overview of connecting mobile devices to CICS” on page 43.

4.2.8 Bean Validation 1.0

It is always quite important to validate input received from the user to maintain data integrity in application logic. For example, it is necessary to validate an input of an e-mail address before sending an e-mail. Bean Validation is a new validation model available as part of the Java

Enterprise Edition 6 platform. The model is supported by constraints in the form of annotations placed on a field, method, or class of a JavaBeans component.

The Bean Validation API is introduced as a standard mechanism to validate Enterprise JavaBeans in all layers of an application, including, presentation, business and data access. Before the Bean Validation specification, the JavaBeans were validated in each layer. To prevent the re-implementation of validations at each layer, developers bundled validations directly into their classes or copied validation code, which was often cluttered. Having one implementation that is common to all layers of the application simplifies the developers work and saves time.

With Bean Validation, CICS Liberty provides validations for JavaBeans at each layer of an application. For more information about Bean Validation, see the following website:

http://www-01.ibm.com/support/knowledgecenter/SSAW57_8.5.5/com.ibm.websphere.nd.multiplatform.doc/ae/welc6tech_bv.html?lang=en

4.2.9 PHP support by Dynamic Scripting Feature Pack

PHP support is not a feature provided by WebSphere Application Server Liberty Profile, but a feature provided by CICS Transaction Server Feature Pack for Dynamic Scripting V2.0. The feature pack provides an agile web application platform for developing and running modern web applications. You can use the CICS Transaction Server Feature Pack for Dynamic Scripting V2.0 to create and run PHP applications that meet your specific needs or the needs of your clients. For more information, see the following website:

http://www-01.ibm.com/support/knowledgecenter/SSGMCP_5.2.0/com.ibm.cics.ts.dynamicscripting.doc/WelcomePage/WelcomePage.html?lang=en

4.3 Migrate existing Java presentation logic into CICS Liberty

Migrating a Java application from one platform to another is easier than the migrating applications developed in other languages. For the customers who have existing Java presentation logic outside CICS and want to migrate the logic into CICS Liberty, IBM WebSphere Application Server Migration Toolkit can make the migration easier. The customers need to make sure the features used in existing Java applications are supported by CICS Liberty.

The IBM WebSphere Application Server Migration Toolkit is a suite of tools and collections of knowledge that enables your organization to quickly and cost-effectively migrate to WebSphere Application Server V7.0 through V8.5.5, whether from a previous version of WebSphere Application Server or competitive application servers including Apache Tomcat Server, JBoss Application Server, Oracle Application Server, and Oracle WebLogic Server.

For more information, please refer to:

http://www.ibm.com/developerworks/websphere/downloads/migtoolkit/

The Migration Toolkit can help to complete part of the migration, but not all. You still need to tailor your application to make it work inside CICS. For example, the data access in mainframe is quite different from that in distributed platform. You may need to consider to move some recoverable application data into VSAM or DB2, and then use JCICS API or JDBC to access to the data. For more information, please refer to Chapter 5, “Consolidating applications in CICS Liberty” on page 29.

5

Consolidating applications in CICS Liberty

This chapter describes the considerations for porting existing Java Enterprise Edition (JEE) and other existing Java applications to CICS Liberty. Also the considerations for developing Java applications that exploit features unique to CICS Liberty as well as other Liberty features are described.

In this chapter, the following topics are covered:

5.1 Porting Java application to CICS Liberty

The features in a release of CICS Liberty are a subset of the features supported by the corresponding release of Liberty provided in *WebSphere Application Server for z/OS.* Existing JEE and other Java applications can be ported or migrated to a CICS Liberty as long as the applications only use those features that are supported by the currently installed release level of CICS Liberty.

The list of supported CICS Liberty features for a release is determined by which features have been verified by the CICS development organization. For a list of the currently supported CICS Liberty features, go to the following web page:

http://www.ibm.com/support/knowledgecenter/SSGMCP_5.2.0/com.ibm.cics.ts.java.doc/topics/liberty_features.html

Other enhancements may be available and should work with certain caveats or restrictions in CICS Liberty. For information about these enhancements to CICS Liberty, go to the CICS Developers community at the following web page:

https://www.ibm.com/developerworks/community/blogs/cicsdev

One example of such an enhancement is the addition of support for MQ which is described in the article downloadable at the following web page:

https://www.ibm.com/developerworks/community/blogs/cicsdev/entry/using_websphere_mq_java_bindings_with_a_liberty_jvm_server

<< This entire section should be worded so that it conforms to the official policy at the time of publication. Also the signature string below should not be changed>>

Note: To determine which features which may be available in a particular instance of CICS Liberty, locate the product service signature string in the Liberty startup messages (see an example below)

product = CICS Transaction Server for z/OS 5.2.0, CICS Liberty NOTUSAGE, WebSphere Application Server 8.5.5.1, WAS FOR Z/OS 8.5.5.1 (wlp-1.0.4.cl50120140502-1451)

In this example, the features shipped with release 8.5.5.1 of the WebSphere Application Server for z/OS Liberty profile are available. A list of exactly which features that may be available for a particular release of *WebSphere Application Server for z/OS* can be obtained by going to URL.

http://www.ibm.com/support/knowledgecenter/SSAW57_8.5.5/com.ibm.websphere.wlp.nd.iseries.doc/ae/rwlp_feat.html?cp=SSAW57_8.5.5%2F2-0-1-2-0

5.1.1 Which Java applications should be migrated to CICS?

Not every Java application that meets the above criteria should be ported to CICS Liberty. Consider porting only those Java applications when there is a logical reason or a need for the application to run in a CICS JVM server.

Examples of why a Java application should be migrated to CICS Liberty are:

- If you have an Java application that interacts with a CICS application using CICS Transaction Gateway, CICS MQ Bridge or a JEE Connector (J2C) connection factory.

Consider moving this Java application to CICS Liberty in order to take advantage of being colocated with the CICS application currently being accessed by the user interface.

- If, in CICS you want to start exploiting some of the features unique to CICS Liberty. For example the JDBC features that allow dynamic binding of a data source at runtime or the feature that coordinates the commits or rollbacks of updates made to remote data sources (JDBC Type 4) in the same logical unit of work as other CICS managed resource.
- Or, if you want to take advantage of using CICS bundles for your applications. Using bundles means that application artifacts can be packaged with the corresponding CICS resource definitions (that is definitions for TRANSACTIONs, PROGRAMs, URIMAPs and so on) so that both the application and its resource definitions can be managed and/or deployed as a single administrative entity.

Since CICS Liberty runs in a CICS JVM server, all of the inherent advantages provided to Java applications by the CICS JVM server are available to CICS Liberty applications.

- Instead of having multiple Java virtual machines (JVM) running a CICS task, there is a single JVM in a JVM server which can spawn up to 256 threads with each thread executing a CICS task. This provides vertical scaling as workload increases. A CICS Liberty JVM server can also be configured with only the features required by an application to execute. Eliminating the features not required in a CICS Liberty JVM server avoids loading unnecessary functions and reduces sever startup time and storage utilization.
- Multiple JVM severs can coexist is a single CICS region with each JVM server configured independently and running different applications. Each JVM server (CICS Liberty and non Liberty) running different applications can provide horizontal scaling of application workload within a CICS region. Currently there can be only one CICS Liberty JVM server in a CICS region that interacts with a Liberty angel process. using z/OS security features. For horizontal scaling of CICS Liberty applications that required z/OS security, the CICS Liberty JVM server will need to be replicated in another CICS region.

For more details about running Java in CICS JVM servers and OSGI, download the white paper at URL:

https://www.ibm.com/developerworks/community/blogs/cicsdev/entry/white_paper_running_java_workloads_with_jvm_servers_and_osgi2

5.1.2 Exploiting the OSGi Framework

Another consideration for consolidating applications in CICS Liberty is to use the OSGi (Open Service Gateway initiative) framework for deploying and administering Java applications. The OSGi framework restructures the components of an application as individual bundles of components or packages that are loosely coupled but when combined constitute an application. This contrasts with packaging solutions where all components are packaged and administered in a single monolithic Java archive file (JAR) which then must be included in the CLASSPATH. This separation of the application into bundles allows for the independent deployment and management of the different bundles that compose an application and potentially the reuse of selected bundles by other applications.

This flexibility means that:

- Different versions or levels of the same package can coexist in CICS Liberty concurrently.
- Since the Java CLASSPATH is not used to load the Java classes, changes can be implemented without the need to stop and restart the JVM.

There are various development tools available (such as CICS Explorer or Eclipse Integrated development environment (IDE) tools, such as Luna or Kepler) to aid in the refactoring of existing Java artifacts like JARs into OSGi bundles for deployment to bundle repositories and the packaging of applications into CICS bundles for deployment to a CICS Liberty runtime.

For more information on OSGi packaging in CICS Liberty, go to the following web page:

http://www.ibm.com/support/knowledgecenter/SSGMCP_5.2.0/com.ibm.cics.ts.java.doc/topics/osgi_overview.html

For more information about Liberty application development in general and to download Eclipse and Liberty development plugins, go to the following web pages:

https://developer.ibm.com/wasdev

and/or

http://www.eclipse.org/downloads/packages/

To download CICS Explorer go to URL:

http://www.ibm.com/cics/explorer

Note: Both the Eclipse tooling and CICS Explorer work with common source code management products like Rational Team Concept (RTC).

5.2 Developing new application using JCICS classes

CICS provides Java interfaces to most of the same standard CICS application programmer interfaces (APIs) found in traditional CICS application programs. New Java applications can be developed and/or existing Java applications can be modified to access almost the full set of CICS APIs. Use of these APIs gives Java applications running in a CICS JVM server full access to VSAM files, the ability to link to COBOL and other CICS application programs passing common data areas and containers and the ability to access transient data and temporary storage queues and so on.

Some examples of Java code showing the use of some the JCICS classes and methods that could be used in a CICS Liberty application are shown in Example 5-1 and Example 5-2.

Example 5-1 JCICS Example showing accessing key sequenced VSAM file

```
// Instantiate an instance of a JCICS record holder
com.ibm.cics.server.RecordHolder record = new
   com.ibm.cics.server.RecordHolder();
// Instantiate an key sequence VSAM file
com.ibm.cics.server.KSDS file = new com.ibm.cics.server.KSDS();
// Set the file name
file.setName("FILEA");
// Retrieve and delete a record in a Try/Catch block
try {
   file.read(key.getBytes(),record);
   file.delete(key.getBytes());
catch (Exception e) {
```

Example 5-2 JCICS examples showing the use of channel and containers in linking to a CICS program

```
// Instantiate an instances of a JCICS channel and a JCICS container
Channel channel = task.createChannel("MINICICS-Channel");
Container requestContainer = channel.createContainer("MINICICS");
// Copy contents of the CICS commarea area to the JCICS request container
requestContainer.put(commarea.getByteBuffer());
// Invoke program and pass the channel with the request container
progName.link(channel);
// Retrieve the response container from the channel
Container responseContainer = channel.getContainer("MINICICS");
// Copy contents of the response container to the CICS commarea area
commarea = new com.ibm.ats.odm.COMMAREA(responseContainer.get());
```

The same development tools (CICS Explorer and Eclipse) can be used for the development of Java applications that use JCICS. Accessing CICS resources using JCICS classes should be more understandable to Java programmers than the EXEC CICS equivalents used in other languages..

Note: Some JCICS classes and methods should not be used in CICS Liberty. For example any API that deal with a terminal (that is methods like clear, converse, erase and so on) would not be used since these require a principle facility or terminal and there would be no terminal associated with a task running in CICS Liberty.

5.2.1 Java access to records and their fields

Eventually, most if not all applications running in CICS will need to work with a record and its fields when accessing a VSAM file, a DB2 row and so. Most Java development on other platforms work with stream-oriented data rather than work with record-oriented data as typical applications in CICS. Because of this, there are options for generating a Java class that represents the layout of a records and the corresponding *setters* and *getters* methods for accessing the individual fields in the record. See examples of setter and getter methods being used in Example 5-3.

Example 5-3 Example of the using of GETTERs and SETTER methods with a COMMAREA Class

```
commarea.setName(request.getParameter("custName"));
   commarea.setEffectdate(request.getParameter("effectDate"));
   commarea.setAmount(Long.parseLong(request.getParameter("amount")));
   commarea.setAge(Long.parseLong(request.getParameter("age")));

   // Invoke a CICS COBOL program passing a common area
   commarea = MiniCICS.invoke(commarea);

   request.setAttribute("custName",commarea.getName());
   request.setAttribute("age", commarea.getAge());
   request.setAttribute("amount", commarea.getAmount());
   request.setAttribute("effectDate", commarea.getEffectdate());
```

Some of the options for generating Java classes from record oriented layouts are described below:

- The J2C component of the Java EE Connectors feature of IBM Rational® Application Developer (RAD) includes a wizard that generates a Java data bean with the getter and setter methods that can be used to access individual fields within a record. Supported

languages include COBOL, C and PL/I, see Figure 5-1. Also the Java methods generated by the RAD wizard include support for code page conversion and conversion between Big and Little Endian data types, see Figure 5-2 on page 34.

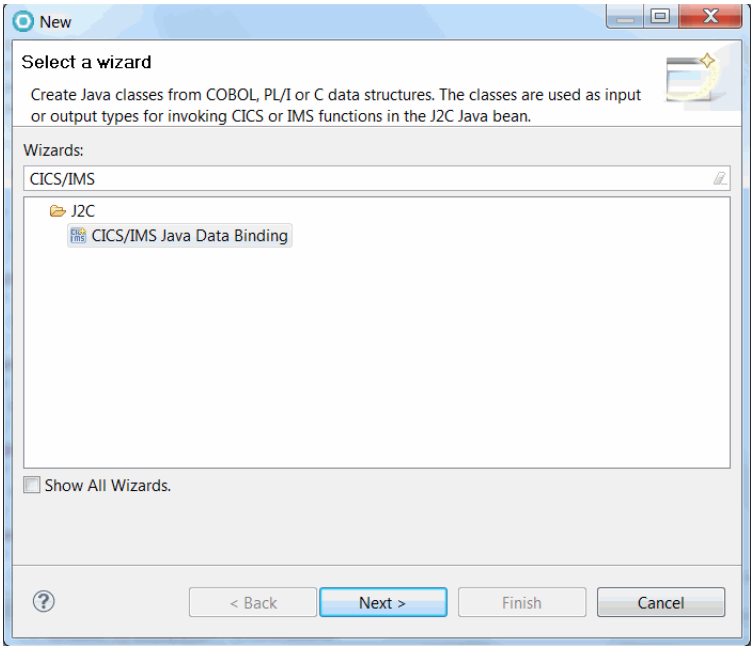


Figure 5-1 Rational Application Developer J2C wizard

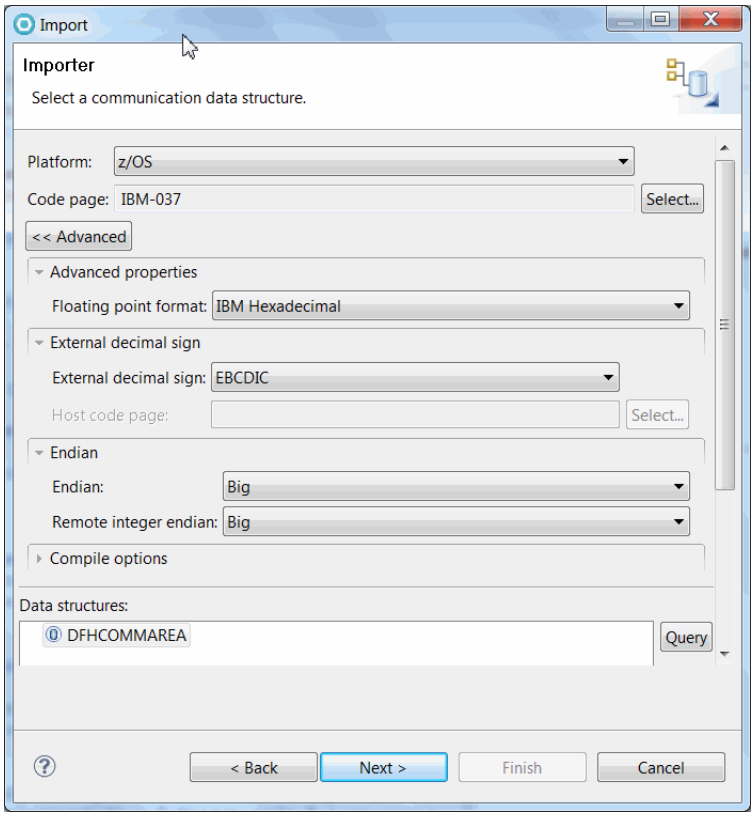


Figure 5-2 RAD J2C Wizard Import Options

- Another option for generating a Java class with *getters* and *setters* for fields from an existing record layout is the JZOS Assembler/COBOL Record Generator utility included as part of the IBM Experimental JZOS Batch Toolkit for z/OS SDKs (see Example 5-4). This utility runs on z/OS and can be used to generate Java classes from COBOL copy books and Assembler DSECTs

Example 5-4 Snippet of JCL executing the RecordClassGenerator Utility

```
//* Generate a .java file for each copy book
//JAVA EXEC PROC=EXJZOSVM,VERSION='50'
//MAINARGS DD *
com.ibm.jzos.recordgen.cobol.RecordClassGenerator
  bufoffset=false
  package=com.ibm.ats.cobol.records
  outputDir=~/cobgen
//SYSADATA DD DSN=&&ADATA,DISP=(OLD,DELETE)
```

For more information about *IBM Experimental JZOS Batch Toolkit for z/OS SDKs* go to URL

https://www.ibm.com/developerworks/community/groups/service/html/communityview?communityUuid=2acdb076-7582-45b5-93a5-781f90169bd3

Note: Notice the Java statement in Example 5-2 on page 33 where the responseContainer was 'moved' to the commarea using a constructor method. This was required because this Java class was generated by the JZOS Record Generator utility which did not generate a setter method that accepts a byte array for updating the entire record. Only setter methods for individual fields are generated. The RAD J2C wizard does generate all of required methods.

More details on both of these techniques can be found in Chapter 8 of Redbook *IBM CICS and the JVM server: Developing and Deploying Java Applications*, SG24-8038.

Note: Bean Validation 1.0 is one the features supported by CICS Liberty but neither of these two methods would include any bean validation annotations. So if you want to include bean validation support, the annotations would have to be manually added.

5.2.2 Debugging Java in CICS Liberty

Both the CICS Explorer and Eclipse IDEs provide means to debug remote Java applications by stepping thought lines of code, setting break points, inspecting the contents of variables and so on. This technique can be used to also debug Java applications running in CICS Liberty. Figure 5-3 on page 36 shows remotely debugging a CICS Liberty application in CICS Explorer.

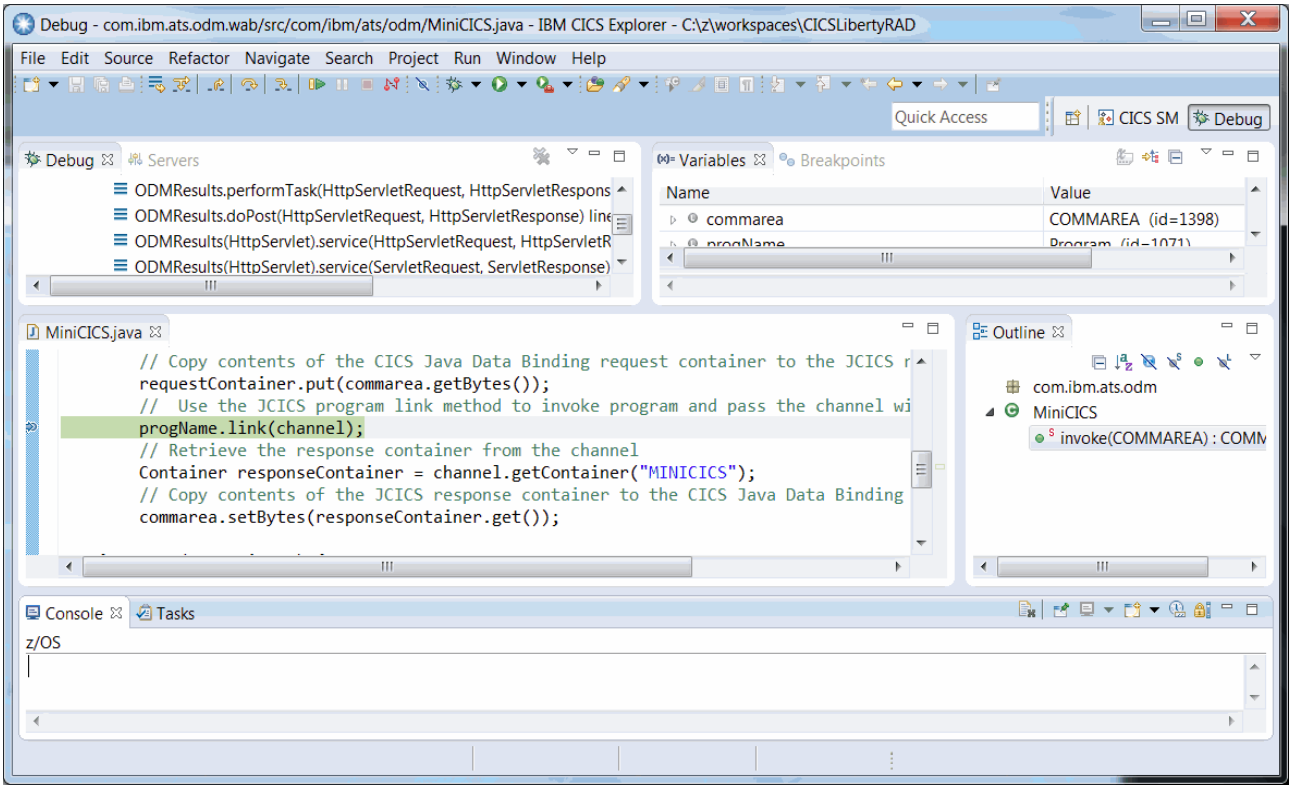


Figure 5-3 Debug Perspective of a Java Applications running in CICS Liberty

Set up for a remote debugging session for CICS Java from a developer's IDE is the same as when debugging on any other remote platform. Remote debugging can be very useful when migrating an existing application to CICS Liberty or the development of a new application for CICS Liberty.

Note: To enable remote debugging of a CICS Liberty server the following statement must be included in the JVM profile of the CICS Liberty JVM server resource definition.

-agentlib:jdwp=transport=dt_socket,server=y,address=<port>

5.3 Developing new applications using other Liberty features

There are Liberty features that are unique to CICS or provide functions that are exploited by CICS Liberty for better integration with CICS Transaction Server for z/OS and IBM CICS Transaction Server for z/OS Value Unit Edition.

5.3.1 CICS Liberty Java Database Connectivity (JDBC) options

CICS Liberty supports two different features for accessing JDBC data sources. The first JDBC feature (cicsts:jdbc-1.0) is unique to CICS Liberty, and is tightly integrated with CICS for exploiting the existing CICS DB2 connection resource (that is the DB2CONN resource). This feature only supports a DB2 local connection (JDBC Type 2). The second JDBC feature

(jdbc-4.0) is supplied by WebSphere Application Server for z/OS Liberty Profile and supports remote connections (JDBC Type 4).

Note: Use of the JDBC feature provided in the base Liberty product is discouraged for local or JDBC Type 2 connections in CICS Liberty.

Local Connections (JDBC Type 2)

The CICS JDBC connection feature (cicsts:jdbc-1.0) is used when an application accesses a local DB2. The access is automatically included in a logical unit work managed by the CICS transaction manager. Commits and rollbacks of updates made to DB2 are automatically coordinated with commits and rollbacks of other CICS resources with no additional coding required. Security for these applications are also managed by the CICS DB2 connection definition.

Remote Connections (JDBC Type 4)

CICS Liberty uses the Liberty JDBC feature (jdbc-4.0) for accessing remote DB2 subsystems (JDBC Type 4). Java applications that use this feature may use Java Transaction API (JTA) to integrate commits and rollbacks for remote JDBC system with CICS transaction management. This allows a remote DB2 connection to participate in the same logical unit of work as other CICS resource such as access to VSAM files, transient data queues, temporary storage queues and so on.

Note: The information displayed by a -DISPLAY DDF command provides the information required to configure JDBC Type 2 and Type 4 connections. The value of the Location provides the location name, and DOMAIN and TCPPORT values provides the server name and port number (see below):

```
DSNL080I  #DI2C DSNLTDDF DISPLAY DDF REPORT FOLLOWS:
DSNL081I STATUS=STARTD
DSNL082I LOCATION           LUNAME            GENERICLU
DSNL083I DSNV10P2           GBIBMIYA.IYCWZDB0 -NONE
DSNL084I TCPPORT=40100 SECPORT=30100 RESPORT=50101 IPNAME=-NONE
DSNL085I IPADDR=::9.20.5.0
DSNL086I SQL    DOMAIN=winmvs2c.hursley.ibm.com
DSNL086I RESYNC DOMAIN=winmvs2c.hursley.ibm.com
DSNL089I MEMBER IPADDR=::9.20.5.0
DSNL105I CURRENT DDF OPTIONS ARE:
DSNL106I PKGREL = COMMIT
DSNL099I DSNLTDDF DISPLAY DDF REPORT COMPLETE
```

5.3.2 JDBC connection options

There are two ways to identify the DB2 system to which the application is to connect.

They are:

- Using a JDBC provided DriverManager by providing a Uniform Resource Locator (URL) to provide connection properties
- Using a DataSource by doing a Java Naming and Directory Interface (JNDI) lookup of the datasource name to obtain connection properties at execution time

DriverManager JDBC Connections

The driver manager technique uses one of two types of URL to identify the target DB2 subsystem and optional connection properties.

- A default URL (see Example 5-5) which does not provide an explicit DB2 location name and no properties and defaults to the DB2 system specified by the CICS DB2CONN resource.

Example 5-5 Example of connecting to the default DB2 subsystem using a Default UR

```
Connection connection = DriverManager.getConnection("jdbc:default:connection");
```

- The other URL format provides a DB2 location name in the URL (see Example 5-6). The location name can be the location name of the local DB2 subsystem or the location name of a remote DB2 subsystem if DB2 Distributed Data Facilities (DDF) has been configured between the local and remote DB2 subsystem.

Example 5-6 Example of requesting an explicit connection to DB2 subsystem by location name

```
Connection connection = DriverManager.getConnection("jdbc:db2os390:location");
```

Note: In a Liberty JVM server, DriverManager connections requires the enablement of the CICS JDBC (cicsts:jdbc-1.0) Liberty feature.

DataSource JDBC Connections

The DataSource method allows an application to use the Java Naming and Directory Interface (JNDI) to perform a lookup of the data sources defined in the Liberty server.xml file. A special JNDI name of jdbc/defaultCICSDataSource (see Example 5-7) is used by the CICS JDBC feature for CICS JDBC connections. Connections to remote DB2 subsystems use the JNDI name of the datasource defined in the server.xml file (see Example 5-8).

Example 5-7 JNDI Lookup for the default CICS JDBC Data source

```
context = new InitialContext();
dataSource = (DataSource) context.lookup(jdbc/defaultCICSDataSource);
Connection connection = dataSource.getConnection();
```

Example 5-8 JNDI Lookup for a remote JDBC Data Source

```
context = new InitialContext();
dataSource = (DataSource) context.lookup(jdbc/myDataSource);
Connection connection = dataSource.getConnection();
```

Example 5-9 is a snippet of the server.xml file showing the JDBC JNDI configuration required for these examples.

Example 5-9 Contents of server.xml related to JDBC

```
<featureManager>
            .....
       <feature>cicsts:jdbc-1.0</feature>
       <feature>jdbc-4.0</feature>
             .....
   </featureManager>
.......
<dataSource jndiName="jdbc/MyDataSource">
     <jdbcDriver libraryRef="defaultCICSDb2Library"/>
```

```
     <properties.db2.jcc databaseName="DSNV10P2" driverType="4"
         password="{xor}Lz4sLCgwLTs=" portNumber="40100"
         serverName="winmvs2c.hursley.ibm.com" user="DB2USER”/>
</dataSource>

<cicsts_dataSource id="defaultCICSDataSource"
jndiName="jdbc/defaultCICSDataSource"/>

<cicsts_jdbcDriver id="defaultCICSJdbcDriver" libraryRef="defaultCICSDb2Library"/>

<library id="defaultCICSDb2Library">
          fileset dir="/usr/lpp/db2/jdbc/classes" includes="db2jcc4.jar
            DB2jcc_license.jar”/>
         <fileset dir="/usr/lpp/db2/jdbc/lib" includes="libdb2jcct2zos4_64.so"/>
</library>
```

Note: The cics_ts_jdbcDriver and library elements are required for both DriverManager and DataSource connections. They identify the libraries from where the JDBC driver components reside and the Java JAR files and shared object library to be used.

Part 3

Mobile

This part introduces the ability to expose existing CICS programs to mobile devices, as part of a new workload. It investigates two key mechanisms by which to do this, both of which involve hosting Java based transformation services in CICS.

These two key mechanisms involve the use of CICS Liberty, CICS Java, and the JSON and XML data serialisation formats. CICS Liberty allows standard Java components to be deployed to CICS, CICS Java allows CICS Web Services to run in a Java-based pipeline.

The following chapters are included:

- Chapter 6, “Overview of connecting mobile devices to CICS” on page 43
- Chapter 7, “Mobile devices and CICS Liberty” on page 55
- Chapter 8, “Mobile devices and CICS Java” on page 61

6

Overview of connecting mobile devices to CICS

This chapter discusses some general considerations when using mobile devices to interact with CICS applications. The subsequent two chapters will investigate some specific CICS technologies for connecting mobile devices using IBM CICS Transaction Server for z/OS Value Unit Edition:

- Chapter 7, “Mobile devices and CICS Liberty” on page 55
- Chapter 8, “Mobile devices and CICS Java” on page 61

In this chapter we discuss the following topics:

6.1 Mobile devices and IBM CICS Transaction Server for z/OS Value Unit Edition

For many years CICS has been capable of hosting programs that can be called from outside CICS. Technologies such as the CICS Transaction Gateway, CICS Web Support and CICS Web Services have enabled integration of CICS assets in a heterogeneous computing environment. The clients of these services could be dumb terminals, web browsers, peer servers, or even mobile devices.

Exposing existing CICS assets to new types of clients can be an example of a New Workload, particularly if the client is a mobile device, or other System of Engagement. There are many technologies that can be used to connect a mobile device to CICS, some of which involve hosting transformational services in CICS. The transformational service will be a candidate for approval to be hosted in IBM CICS Transaction Server for z/OS Value Unit Edition if it:

- enables a new type of client (that is, a new workload) to call a program in CICS;
- and is hosted in a JVM server in CICS;
- and links to a target application program hosted in a regular CICS region.

Examples of transformational services that enable connectivity include:

- transforming data in JSON format to and from CICS application data format
- transforming data in XML format to and from CICS application data format.

Figure 6-1 on page 44 illustrates a mobile device being used to access CICS. A transformational service is hosted in IBM CICS Transaction Server for z/OS Value Unit Edition, and it provides access to applications from another CICS region.

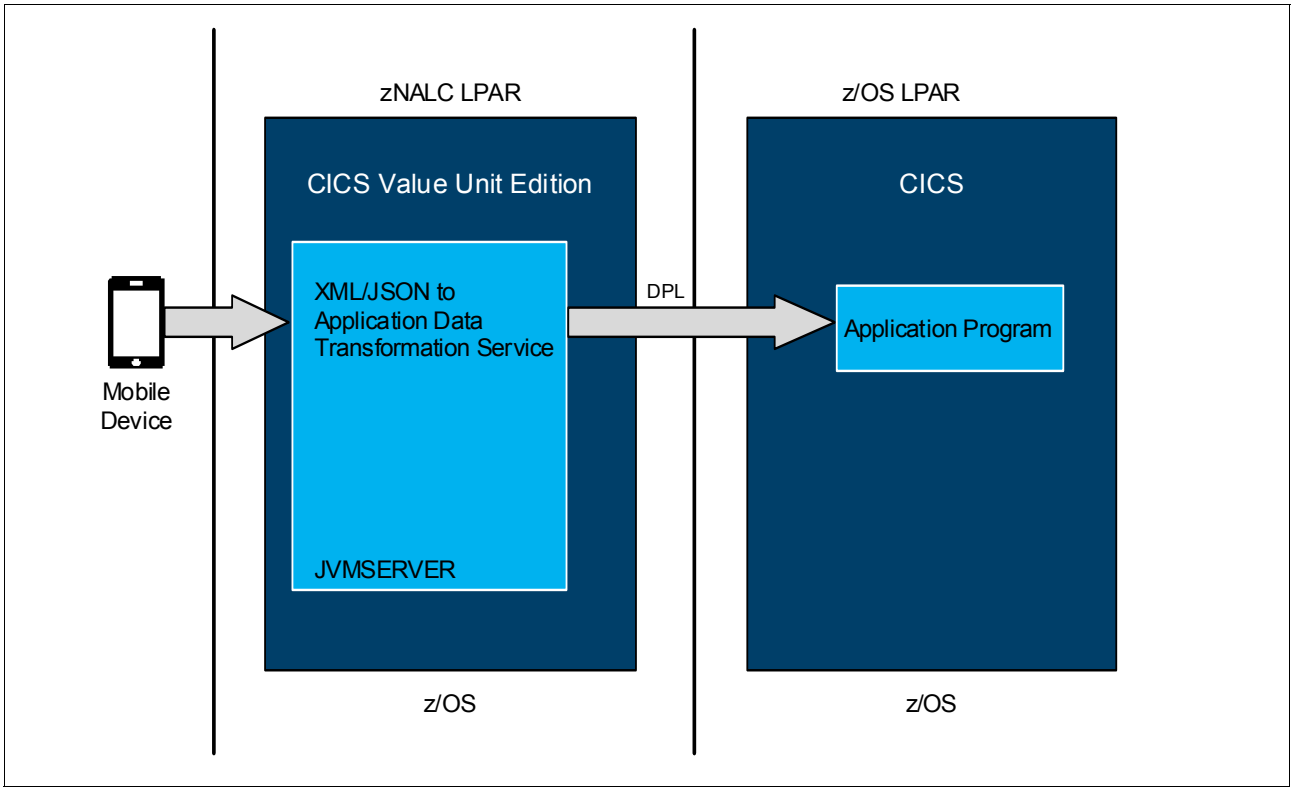


Figure 6-1 Hosting a transformational service in IBM CICS Transaction Server for z/OS Value Unit Edition that links to an existing CICS program

Two major connectivity options exist that can be used to satisfy the above criteria:

1. Hosting transformational services in a WebSphere Application Server Liberty Profile environment in CICS. This can involve using standard Java data manipulation technologies such as JAX-RS and JAX-WS. The WebSphere Application Server Liberty Profile is hosted within a JVM server in CICS, and is therefore a candidate for approval to be deployed to IBM CICS Transaction Server for z/OS Value Unit Edition.
2. Hosting CICS' integrated transformational services in a JVM server. This involves hosting a WSBind file in a Java-based pipeline in CICS. The Java-based pipeline is hosted in a JVM server in CICS, and is therefore a candidate for approval to be deployed to IBM CICS Transaction Server for z/OS Value Unit Edition.

These two techniques are investigated further in the following two chapters: Chapter 7, "Mobile devices and CICS Liberty" on page 55, and Chapter 8, "Mobile devices and CICS Java" on page 61. The remainder of this chapter reviews issues that are common across both scenarios.

Note: Other techniques do exist to connect a mobile device to CICS, but the associated workload is less likely to be approved to be hosted in IBM CICS Transaction Server for z/OS Value Unit Edition. The techniques presented in this book are candidates for approval for use with IBM CICS Transaction Server for z/OS Value Unit Edition.

If you're interested in alternative connectivity options, you could consider using:

- IBM WebSphere Liberty z/OS Connect
- CICS Web Support and the 3270 Web Bridge
- CICS' native Web Services implementation
- the CICS Transaction Gateway
- connectivity through IBM MQ
- or any other mechanism by which work can be sent to CICS.

Mobile devices, and the intermediate servers they communicate with, are general purpose computing devices, and with sufficient effort can be made to communicate using any protocol supported by CICS.

6.2 The use of mobile devices with CICS

The revolution in mobile computation is a significant opportunity for CICS based organizations. By extending existing enterprise applications onto a mobile platform, a business can capitalize on its existing investment without the need to develop an entirely new solution to support mobile services. In addition, a line of business can provide service to users who increasingly expect to be able to interact with an organization using their mobile phone.

As a platform, the primary benefits offered by CICS in support of mobile devices are noted in the following list:

- Provide reuse of existing enterprise services.

  Existing assets and investment can be reused as part of a mobile application. Any existing investment in Web Services can be leveraged on behalf of mobile clients.
- Provide simplified consumption of enterprise data.

CICS supports access to data using two popular data serialization formats: XML (Extensible Markup Language) and JSON (JavaScript Object Notation). JSON is a particularly popular data format for use from mobile devices.

- CICS already operates at the heart of the enterprise.

  Hosting mobile applications within CICS brings them closer to the enterprise data that they are accessing, minimizing application path lengths and keeping response times down.

- Adopt a RESTful architectural style for service delivery.

  A RESTful architectural style is one where the target resource, and the operation to be performed against it, are defined by a combination of a well-structured Uniform Resource Identifier (URI) and one of the four Hypertext Transfer Protocol (HTTP) methods (GET, POST, PUT, and DELETE). This architectural approach is favoured amongst developers of applications for mobile devices.

- Provide capacity to manage mobile workload.

  Customers around the world use CICS TS to host hundreds of millions, and in some cases billions, of transactions per day. CICS Workload Management provides a robust and scalable platform suitable for supporting the heaviest of mobile workloads.

6.3 Accessing services using XML and JSON

There are two common standards based technologies for cross-platform data serialization, Extensible Markup Language (XML) and JavaScript Object Notation (JSON). CICS supports them both.

It is likely that workload initiated from a mobile device will use one or other of these data serialization formats.

The two formats share much in common: they both provide a way to describe structured data in an interoperable text based form. But their differences do make them more or less suitable for particular usage scenarios. Either can be used to serialize data when contacting CICS from a mobile device, either directly, or indirectly via an intermediate gateway, mediation service, or server.

6.3.1 Extensible Markup Language (XML)

XML is readily recognizable by the presence of angle-brackets in the serialized data. XML tags wrapper each atomic data unit. The size of these tags can be a significant proportion of the total size of the serialized data, which is the main cause of XML's reputation as a verbose (and therefore inefficient) data format. In Example 6-1, 160 characters of markup data are used, compared to 85 characters of application data.

XML based Web Services have been supported in CICS Transaction Server since Version 3. They involve a Web Service Description Language (WSDL) document that describes the data format for a Service in an interoperable form. Client programs can be written or generated to call the XML based Web Service using the information from the WSDL document.

The XML representation of the data is encapsulated in a SOAP message. SOAP is the messaging protocol. A SOAP message can optionally contain an extensible list of Headers in addition to the XML application data. A family of related specifications have evolved around SOAP and WSDL to allow for sophisticated data exchange models, including such Qualities

of Service as Transactionality, Identity Propagation, and dynamically re-configurable Addressing.

SOAP implements its own error format, the SOAPFault. Applications pass error information to each other using SOAPFault messages.

XML based Web Services are widely adopted in many industries, and in many programming environments. They remain a favorite technology for interoperability scenarios.

Example 6-1 Example XML Data, 160 characters of markup, 85 characters of application data

```
<contact xmlns="http://example.org/contacts">
  <name>International Business Machines Corp.</name>
  <address>
    <street>1 New Orchard Road</street>
    <city>Armonk</city>
    <state>NY</state>
    <code>10504-1722</code>
  </address>
  <phone>800-426-4968</phone>
</contact>
```

6.3.2 JavaScript Object Notation (JSON)

JSON is a relatively young technology compared to XML. It offers an alternative method for serializing data, characterized by the presence of curly-braces in the data. It generally serializes data to a shorter form than the equivalent XML, leading to its reputation as an efficient data format. In Example 6-2 on page 48, 81characters of markup data are used for the same application data as was used in Example 6-1.

JSON is particularly associated with the JavaScript programming language, which is popular as a language for mobile devices. It is often the data format of preference amongst developers of mobile applications. CICS Transaction Server has supported JSON based Web Services since version 4 release 2, through use of the CICS TS Feature Pack for Mobile Extensions.

JSON does not have a direct equivalent to the WSDL document, so it can be difficult to document the characteristics of a JSON interface in a formal fashion. A JSON-Schema language does exist for defining the syntax of JSON data, but it is not widely implemented throughout the industry.

JSON does not have the wealth of supporting specifications that are available for XML Web Services, this can limit it is usefulness for some tasks, but makes it more efficient for others.

JSON enables (but does not require) a Representational State Transfer (RESTful) interface to data. Typical CICS programs implement a remote procedure call; they accept input, perform an action, and return output. This pattern is referred to as Request-Response. A RESTful interface is quite different. It is data driven, where each data fragment is referenced through it's own URI, and the only actions that can be performed on the data are Create, Read, Update and Delete. These actions map to the underlying HTTP methods of POST, GET, PUT and DELETE. This concept is a little exotic under CICS, but is widely used amongst JSON service providers. The association between JSON and RESTful Services is sufficiently strong that the terms are sometimes used interchangeably.

Example 6-2 Example JSON Data, 81 characters of markup, 85 characters of application data

```
{
  "name": "International Business Machines Corp.",
  "address": {
     "street": "1 New Orchard Road",
     "city": "Armonk",
     "state": "NY",
     "code": "10504-1722"
   }
   "phone": "800-426-4968"
}
```

6.3.3 Key differences between XML and JSON

Here are a list of key differences between the two formats, as implemented in CICS:

- The content of a SOAP message is XML data, whereas a JSON message contains JSON data. JSON and XML are different encoding mechanisms for describing structured data.
- JSON tends to be a more efficient encoding mechanism, so a typical JSON message is smaller than the equivalent XML message.
- JSON is easy to integrate in JavaScript applications, but XML is not. This difference makes JSON a preferred data format for many mobile application developers.
- SOAP provides a mechanism to add Headers to a message, and a family of specifications for qualities of service. For example, WS-Security defines sophisticated rules for securing SOAP messages, and WS-AtomicTransactions defines a distributed two-phase commit protocol. JSON does not have this mechanism, it relies on the services of the underlying HTTP network protocol. This reliance results in fewer options for securing and configuring a workload.
- SOAP web services are described by using WSDL documents. JSON web services are structured less formally; they tend to be loosely coupled and prefer informal documentation often including examples of serialized data.
- SOAP has a larger ecosystem of related tools that can help with application development.
- SOAP web services have an explicit error format that involves using SOAP Fault messages. There is no equivalent for JSON.
- SOAP web services support use of both HTTP and IBM MQ based messaging, JSON requires HTTP.
- JSON web services support both a RESTful and a Request-Response driven interface, SOAP only supports the Request-Response interface.
- The comparative performance characteristics of the two technologies are difficult to predict. The smaller payload of a JSON message does not necessarily result in a cheaper or more efficient Web Service. The business cost is further complicated by the deployment technologies; this involves the use of IBM CICS Transaction Server for z/OS Value Unit Edition, IBM Mobile Workload Pricing for z/OS, and System z Application Assist Processors.

Business requirements are likely to dictate which data serialization protocol is used in a project. As a general rule, SOAP tends to be more suitable for server to server communication (for Qualities of Service), and JSON is more suitable for mobile device to server communication (for ease of client side development). In some deployment scenarios JSON may be used between the Mobile Device and an intermediate server, and SOAP is used for further server to server communication.

6.4 CICS Web Service development strategies

Enabling a CICS program to be called from a mobile device often involves exposing it as a remotely callable Web Service.

There are two main service enablement techniques used to do this. The first is called *Bottom-Up*, and involves exposing an existing asset to new callers. The second is called *Top-Down*, it involves implementing a new service that conforms to the requirements of an external specification. A third hybrid scenario also exists, it is called *meet-in-the-middle*.

In each case a Transformational Service is used to convert the application data to and from the format used for data interchange. The data interchange format can be either JSON or XML, the application data will be structured in the format used by the application (e.g. as described by a COBOL copy-book).

6.4.1 Bottom-Up Service Enablement

In this scenario, a Web Service (either JSON or XML based) is generated from an existing program. The programmatic interface to the existing program (such as a COBOL copy-book) is processed using tools, and transformational meta-data or programs are produced. The specifics of how this works, and the artefacts produced, varies by scenario. But the general concept is shared: you have an existing asset that needs exposing as a Web Service, and the tools make this possible without changing the existing application.

In practice this is not always achievable. There can be characteristics of an existing application that make it unsuitable for use as a Web Service. For example, it might have a 3270 interface, rely on the existence of a CICS Terminal, or use shared-memory for cross program communication. A program's interface may rely on data types that are not supported by the tooling, such as pointers, or redefined data structures.

In most cases an existing program can be exposed as a Web Service, either without changes, or with minimal additional effort.

6.4.2 Top-Down Service Enablement

In this scenario a Web Service (either JSON or XML based) is generated from a specification document. This document is typically either a WSDL document, or a JSON-Schema. Tooling is used to generate new application data structures from that specification.

A new application is then written that uses the generated data structures, and interacts with the existing programs.

This scenario generally results in a better Web Service than in a bottom-up scenario. The interface is defined according to the requirements of the Service, not the existing implementation. Services can be designed to support version control, and to evolve with minimal disruption for existing callers.

An important variation of a top-down scenario involves creation of a *requester* or client program in CICS. This program can invoke a remote Web Service, hosted outside of CICS. Requester mode scenarios can also be considered as part of a New Workload, if the transformational capability is hosted in a JVM server. One approach that can be exploited is to have a CICS program link to a stub program hosted in IBM CICS Transaction Server for z/OS Value Unit Edition, and the stub can invoke the remote Web Service. The stub must either be hosted in a JVM server, or drive a transformation service that is hosted in a JVM

server, in order to be a candidate for approval for IBM CICS Transaction Server for z/OS Value Unit Edition.

6.4.3 Meet-In-The-Middle Service Enablement

This is a hybrid scenario that can combine the best features of both bottom-up and top-down service enablement. It involves mapping an existing program to a defined interface.

A typical meet-in-the-middle scenario involves generating a Web Service interface to an existing application using bottom-up enablement techniques. This is followed by a manual modification of the generated service description documents. The modifications might include renaming fields so that they will make sense to an external developer, removing unnecessary content, adding additional validation rules, adding versioning information, etc.

Once a perfected interface is established, the existing program must be integrated with the new interface. This might involve the use of tooling to map between the two formats, or the generation of new application structures using top-down techniques. The application may need to be changed to implement the new interface. Subsequent maintenance to the Web Service is performed top-down from the service description document.

This technique tends to involve significant additional effort over a pure bottom-up enablement strategy, so it is not popular for that reason. But it results in a better quality of Web Service.

6.5 IBM Worklight and CICS

The recommended strategies for integrating mobile devices into a secure and high performance CICS environment are discussed in the IBM Redbooks publication *SG24-8161-00 Implementing IBM* CICS *JSON Web Services for Mobile Applications*. It is available here:

http://www.redbooks.ibm.com/abstracts/sg248161.html

A key consideration includes the use of IBM Worklight®. IBM Worklight provides a comprehensive platform in which to build, test, run and manage mobile web applications. It can help to reduce both application development and maintenance costs, improve time-to-market and enhance mobile application governance and security.

Figure 6-2 illustrates an IBM Worklight Server being used to mediate between the mobile devices, and CICS.

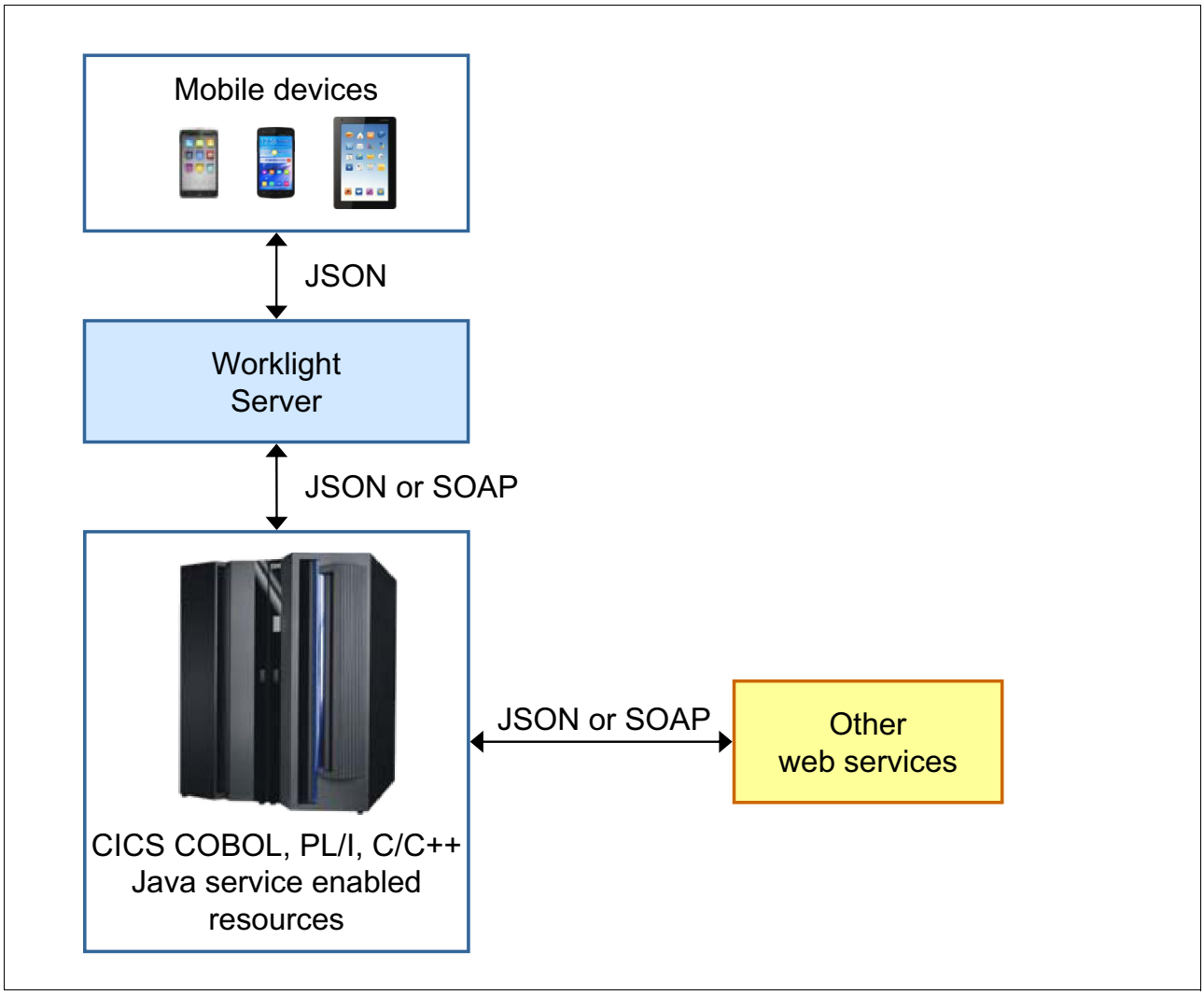


Figure 6-2 IBM Worklight Server as an intermediate node between mobile devices and CICS

The Worklight Server acts as an aggregation point, allowing many devices to share a single connection to CICS. It also provides device abstraction services for the application developer, helps ensure the physical security of the devices, and can provide protocol conversion services for CICS.

The mobile devices can communicate with the Worklight Server using JSON format data, and the Worklight Server can communicate with CICS using whatever mechanism is most convenient, including both JSON and SOAP based Web Services.

The capabilities of IBM Worklight in reference to securing mobile applications are investigated in detail the IBM Redbooks publication *SG24-8179-00 Securing Your Mobile Business with IBM Worklight*. It is available here:

http://www.redbooks.ibm.com/abstracts/sg248179.html

6.6 IBM DataPower and CICS

An IBM DataPower® Appliance is often deployed alongside CICS to secure Web Services workloads.

DataPower can secure mobile traffic before it arrives at CICS, and validate the JSON and XML data at high speeds, acting as a form of firewall. It can also be used in combination with IBM Worklight to provide authentication services for mobile devices.

The combination of IBM DataPower and IBM Worklight to secure mobile workloads is investigated in detail the IBM Redbooks publication *SG24-8179-00 Securing Your Mobile Business with IBM Worklight*. It is available here:

http://www.redbooks.ibm.com/abstracts/sg248179.html

Figure 6-3 illustrates IBM DataPower being used to authenticate users and secure communication between mobile devices and the Worklight Server.

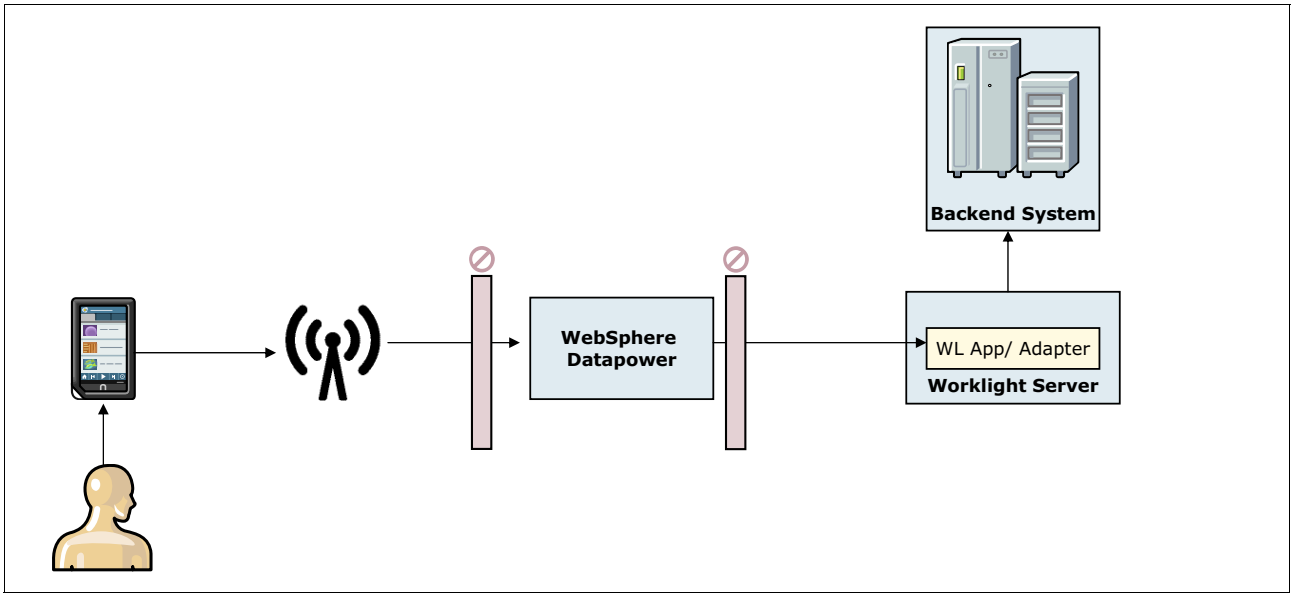


Figure 6-3 IBM DataPower Appliances authenticate the user and protect communication to the Worklight Server

The capabilities of IBM DataPower in reference to integrating and securing Web Services for CICS are investigated in detail in IBM Redbooks publication *SG24-7988-00 Set Up Security and Integration with the DataPower XI50z for zEnterprise*. It is available here:

http://www.redbooks.ibm.com/abstracts/sg247988.html

6.7 Configuration for High Availability

Configuring a CICS infrastructure for high availability involves two primary techniques:

- using TCP/IP load balancing to distribute work across a number of routing regions
- using a Distributed Program Link to route work to a suitable Application Owning Region.

Figure 6-4 illustrates workload being balanced across a series of CICS Routing regions, each of which hosts a transformation service for new work load, and routes the work to a group of Application Owning Regions (AORs) in another LPAR.

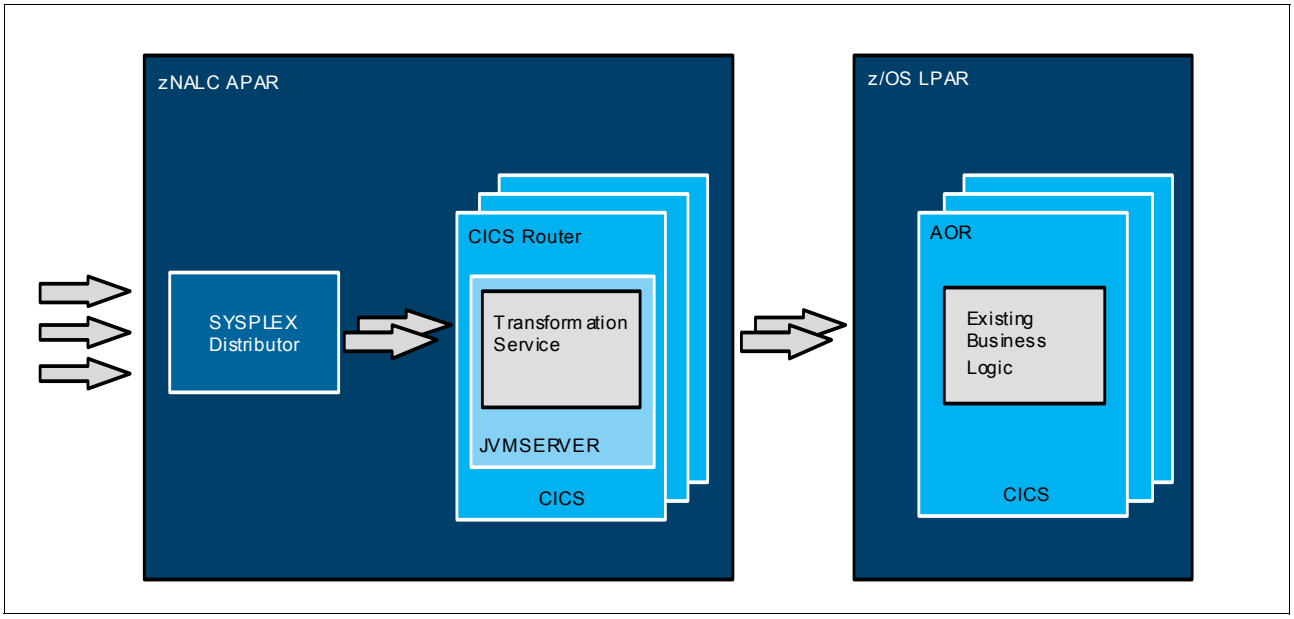


Figure 6-4 Workload balancing across multiple regions

These techniques are suitable for use in both of the implementation scenarios considered in Chapter 7, “Mobile devices and CICS Liberty” on page 55 and Chapter 8, “Mobile devices and CICS Java” on page 61.

IBM Redbooks publication *SG24-7144-01 CICS Web Services Workload Management and Availability* explores the CICS architectures for supporting high availability of Web Services. It is available here:

http://www.redbooks.ibm.com/abstracts/SG247144.html

The following two chapters investigate two different technologies that are suitable for hosting Java based transformation services:

- Chapter 7, “Mobile devices and CICS Liberty” on page 55
- Chapter 8, “Mobile devices and CICS Java” on page 61

7

Mobile devices and CICS Liberty

This chapter discusses hosting Java based Web Services in CICS using CICS Liberty. These Web Services can support either SOAP or JSON data encoding formats, and exploit the JAX-WS and JAX-RS technologies.

It builds upon the general information from Chapter 6, “Overview of connecting mobile devices to CICS” on page 43.

In this chapter we discuss the following topics:

7.1 Hosting Transformational Services in CICS Liberty

At the time of writing, there are two main options for hosting a transformational service in CICS Liberty within CICS. The first exploits a technology called the Java API for XML Web Services (JAX-WS), and the second uses the Java API for RESTful Web Services (JAX-RS). These technologies implement SOAP and JSON for Java applications.

Figure 7-1illustrates a transformational wrapper hosted in CICS Liberty. It is used to expose an existing application to a new workload initiated from mobile devices.

In both scenarios a generated Java interface is hosted in a CICS Liberty environment, and custom control logic is written to interface a Java program with the existing CICS programs.

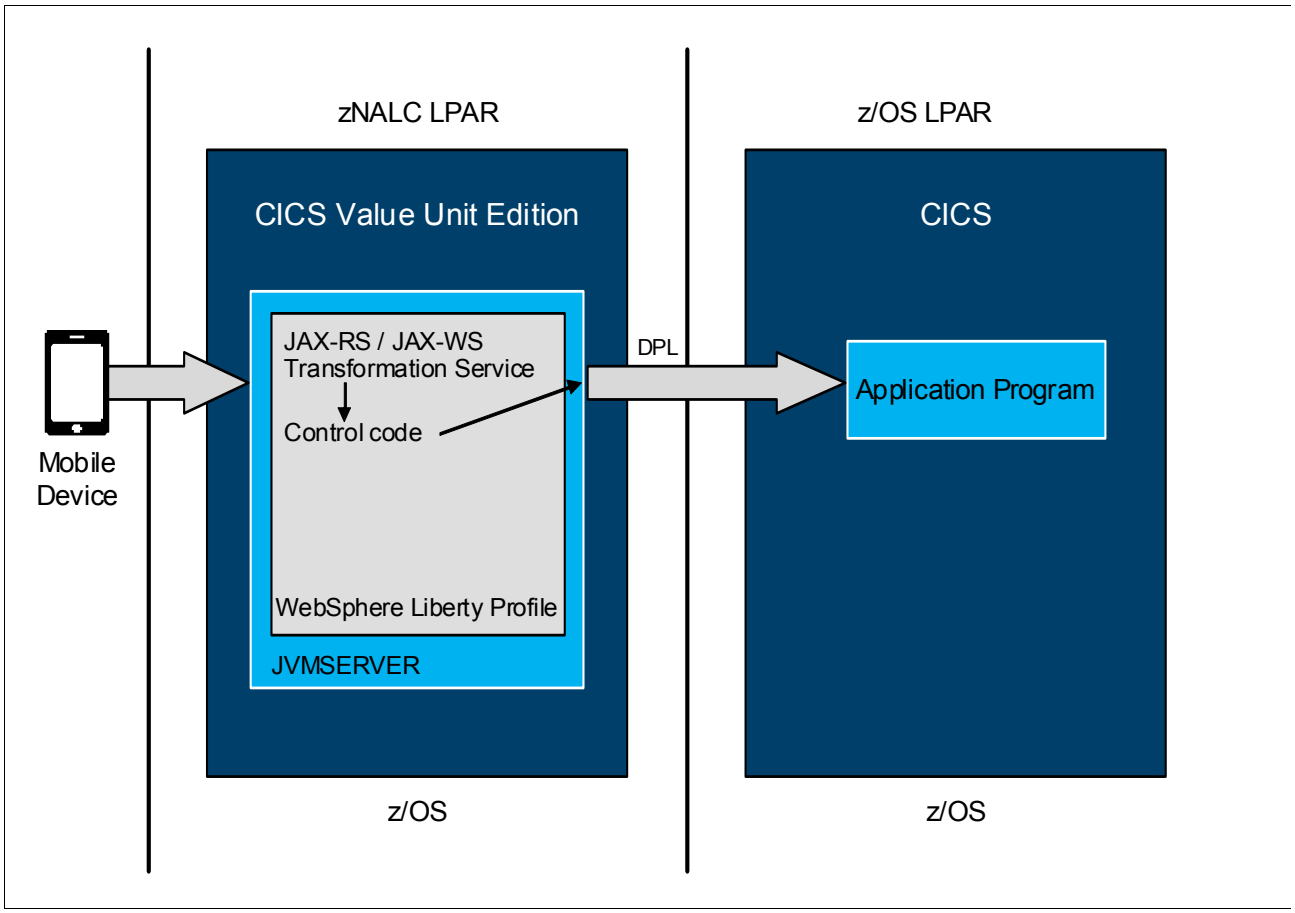


Figure 7-1 Hosting JAX-WS or JAX-RS services in a CICS Liberty environment

We will now consider these two technologies in more detail.

7.1.1 Java API for XML Web Services (JAX-WS)

The Java API for XML Web Services is a standard component of Java. It is available on all platforms supported by Java, and it allows Java applications to implement SOAP based Web Services.

It supports both bottom-up service enablement patterns (where a WSDL interface is generated from an existing Java class), and top-down enablement patterns (where Java

classes are generated from an existing WSDL document). It also supports requester mode scenarios, where a CICS Java program is used to invoke a remote SOAP Web Service.

JAX-WS provides the tools required to bind Java programs to WSDL, and the runtime transformational service required to perform the data transformations. This transformational service is implemented in Java, so is a candidate for approval for IBM CICS Transaction Server for z/OS Value Unit Edition.

Provider mode JAX-WS applications are a fully supported component of the CICS Liberty environment in CICS. Requester mode applications are also supported under CICS Liberty, but are of less value due to the inability to LINK to a Liberty application from a traditional CICS program. Requester mode applications can be hosted in a regular non-Liberty JVM server in CICS.

Characteristics of a JAX-WS application in CICS Liberty

A JAX-WS application is, by definition, a Java application. This in turn implies that a JAX-WS application cannot use bottom-up development techniques to expose an existing non-Java CICS application as a SOAP Web Service. There will always be some form of meet-in-the-middle technique required to wrapper an existing non-Java program with a JAX-WS interface.

This will require additional application development effort compared to some other service enablement strategies, but it can be worth considering. There are several significant benefits:

1. The JAX-WS service is a candidate for approval to be hosted in IBM CICS Transaction Server for z/OS Value Unit Edition.

   This can be relevant for business reasons, independent of the technical considerations.

2. JAX-WS provides a complete and concise mapping of WSDL to Java.

   If you've struggled with mapping complex WSDL documents and XML Schema into COBOL (or other native programming languages), JAX-WS can provide an alternative path to Service Enablement.

   Some XML constructs are inherently difficult to map into COBOL style data. Examples include inheritance hierarchies, recursive data structures, and cross references. Java is more naturally suited to representing such information.

3. JAX-WS is a cross-platform technology

   JAX-WS is a standard part of Java, and is supported by many vendors. The skills required to create JAX-WS services are not CICS specific.

An example JAX-WS application deployed using *CICS Liberty* is available here: *https://www.ibm.com/developerworks/community/blogs/cicsdev/entry/jax_ws_and_jaxb_support_in_cics_ts_v5_2_open_beta_liberty_profile?lang=en*.

Example 7-1 on page 57 illustrates a simple JAX-WS application.

Example 7-1 An Example JAX-WS Hello World Application

```
package org.example;
import javax.jws.*;

@WebService
public class TextService
{
  @WebMethod public String decorateText(String text)
  {
```

```
    return "You said: '" + text + "'";
  }
}
```

7.1.2 Java API for RESTful Web Services (JAX-RS)

The Java API for RESTful Web Services is also a standard component of Java, but not distributed as part of the Java Software Development Kit (SDK). A reference implementation is available on all platforms that are supported by Java. It allows Java applications to implement JSON and XML based Web Services, both in RESTful and Request-Response usage patterns. In this document we will consider JAX-RS for enabling JSON connectivity.

It supports the bottom-up service enablement patterns, where a JSON interface is generated from an existing Java class. It also supports requester mode scenarios, where a CICS Java program is used to invoke a remote JSON Web Service.

JAX-RS provides the runtime transformational service required to perform the data transformations. This transformational service is implemented in Java, and is a candidate for approval to be hosted in IBM CICS Transaction Server for z/OS Value Unit Edition.

Provider mode JAX-RS applications are a fully supported component of the CICS Liberty environment in CICS. Requester mode applications are also supported under CICS Liberty, but are of less value due to the inability to LINK to a Liberty application from a traditional CICS program. Requester mode applications can be hosted in a regular non-Liberty JVM server in CICS.

Characteristics of a JAX-RS application in CICS Liberty

JAX-RS applications are very similar to JAX-WS applications, they share many of the same characteristics. For example, a JAX-RS application must be written in Java. This means (as with JAX-WS) that additional application code must be written in Java to make a JAX-RS application communicate with any non-Java CICS programs. A traditional non-Java CICS program cannot be exposed as a JSON Web Service using JAX-RS in a pure Bottom-up fashion.

The benefits of JAX-RS include:

1. The JAX-RS service is a candidate to be hosted in IBM CICS Transaction Server for z/OS Value Unit Edition

   This can be relevant for business reasons, independent of the technical considerations.

2. JAX-RS provides a simple mechanism to expose Java programs as JSON Web Services.

   Customized mapping information can be stored within the Java application using a library of Java Annotations.

3. JAX-RS is a cross-platform technology

   JAX-RS is a standard part of Java, and is supported by many vendors. The skills required to create JAX-RS services are not CICS specific.

An example RESTful JAX-RS application deployed using *CICS Liberty* is available here: *https://www.ibm.com/developerworks/community/blogs/cicsdev/entry/writing_restful_web_services_using_cics_liberty_server_part_2?lang=en*.

Example 7-2 on page 59 illustrates a simple JAX-RS application.

Example 7-2 An Example JAX-RS Hello World Application

```
package org.example;
import javax.ws.rs.*;
import javax.ws.rs.core.*;

@Path("/json/example")
public class TextService2
{
  @POST
  @Path("/post")
  @Consumes(MediaType.APPLICATION_JSON)
  @Produces(MediaType.APPLICATION_JSON)
  public String decorateText(String text)
  {
    return "You said: '" + text + "'";
  }
}
```

7.2 z/OS Connect and CICS Liberty

IBM WebSphere Liberty z/OS Connect is software that enables z/OS systems such as CICS and IMS to participate in a Mobile computing environment. It runs inside the WebSphere Application Server Liberty Profile and provides a JSON based interface between mobile and cloud devices and back-end systems.

z/OS Connect provides an alternative mechanism for wrappering CICS assets with a JSON interface. However, at time of writing it cannot be hosted in CICS Liberty. This limits its relevance to an IBM CICS Transaction Server for z/OS Value Unit Edition environment.

For further information on z/OS Connect, see:
http://www.ibm.com/developerworks/downloads/ws/waszosliberty/

IBM has released a statement of direction for z/OS Connect, as part of the CICS Transaction Server for z/OS V5.2 announcement letter:
http://www-01.ibm.com/common/ssi/cgi-bin/ssialias?infotype=an&subtype=ca&supplier=897&letternum=ENUS214-107.

That statement reads:

> *IBM intends to deliver IBM WebSphere Liberty z/OS Connect (z/OS Connect) as a common program component of WebSphere Application Server for z/OS, IMS Enterprise Suite for z/OS, CICS Transaction Server for z/OS, and CICS Transaction Gateway. z/OS Connect is intended to provide a simplified, secure, and scalable gateway functionality to route web, cloud, and mobile application traffic that accesses applications provided by the aforementioned z/OS products, as well as z/OS Batch and z/OS UNIX System Services applications. z/OS Connect intends to offer: (i) a fast on-ramp interface to z/OS applications by providing a common access mechanism based on RESTful services; (ii) tooling to allow a cloud or mobile developer to define secure enterprise connectivity without the need for extensive code development or knowledge of System z.*

No further information is available at the time of writing, but z/OS Connect might become relevant to IBM CICS Transaction Server for z/OS Value Unit Edition, at some point after the publication of this book.

7.3 Connectivity from Java to CICS

In both of the above scenarios, the JAX-RS or JAX-WS application will interact with the services and programs in CICS. They do this using the JCICS application programming interface. Application code will be written in Java that converts between Java's data format, and the format used by other CICS programs. For example, the Java application might need to construct a commarea to pass to a COBOL program, on interpret the contents of a CICS Container.

The techniques for interacting with structured records from Java are discussed in 5.2.1, "Java access to records and their fields" on page 33.

7.4 Security considerations

Security of Web Services (both JSON and SOAP) is a broad topic, encompassing many considerations.

Mobile devices are usually connected to CICS using an intermediate proxy. The security configuration might be different between the mobile device and the proxy, and between the proxy and CICS. Mobile devices can benefit from the device security characteristics of IBM Worklight (see 6.5, "IBM Worklight and CICS" on page 50), and the authentication and firewall capabilities of IBM DataPower (see 6.6, "IBM DataPower and CICS" on page 52).

CICS Liberty supports several mechanisms by which a proxy can be authenticated to CICS. These are discussed in 3.5, "Security" on page 16.

At the time of writing, CICS Liberty does not support the use of the WS-Security specification for SOAP based Web Services. Nor does it support OAuth based authentication for mobile devices. But most of the benefits of these protocols can be experienced through the combination of IBM Worklight and/or IBM DataPower with CICS.

7.5 Other considerations

There are many ways to call an existing CICS program from a mobile device. Hosting a transformational service in CICS Liberty offers the advantage that the associated transformational work is a candidate for approval for use with IBM CICS Transaction Server for z/OS Value Unit Edition. However, other options are available, including the use of a Java Pipeline as discussed in Chapter 8, "Mobile devices and CICS Java" on page 61.

One of the limitations of the CICS Liberty approach is that the transformational work is not as tightly integrated with CICS as with some of the other options. For example, the CICS Liberty environment will listen on its own TCP/IP socket, rather than using CICS' own native sockets listener. This will affect the techniques that must be used to investigate problems.

The CICS Liberty environment does not make use of a CICS PIPELINE resource, and does not share the same characteristics that might be familiar from that environment. For example, the CICS supplied Handler programs that implement the various SOAP Web Services specifications such as WS-AtomicTransactions and WS-Security are not available for use. Another difference is that the CICS SOAPFault API is not available for native CICS applications to use. But the pattern of wrappering existing CICS assets with a CICS Liberty hosted transformation service can be very useful.

8

Mobile devices and CICS Java

This chapter discusses hosting Java based Web Services in a CICS JVM server without using CICS Liberty. These Web Services run in a Java-based pipeline in CICS, and use the CICS supplied data transformation service to interact with CICS programs.

This chapter builds upon the general information from Chapter 6, “Overview of connecting mobile devices to CICS” on page 43.

In this chapter we discuss the following topics:

8.1 Hosting Transformational Services in CICS Java

CICS has had the ability to automatically transform XML into application data as part of a Web Service since CICS Transaction Server V3.1. The ability to host this transformation service within a JVMSERVER resource was introduced in CICS Transaction Server V4.2. This Java based transformation service was extended to support JSON as part of the CICS Transaction Server Feature Pack for Mobile Extensions V1.0. The Feature Pack is available for CICS Transaction Server V4.2 and V5.1. The capability is integrated into CICS itself from CICS Transaction Server V5.2.

The JVMSERVER based environment for CICS Web Services is referred to as a *Java-based pipeline*. A Java-based pipeline that exposes an existing program to mobile devices is a candidate for approval for hosting in IBM CICS Transaction Server for z/OS Value Unit Edition.

Figure 8-1 illustrates a Java-based pipeline being used to expose an existing CICS program as an XML or JSON Web Service.

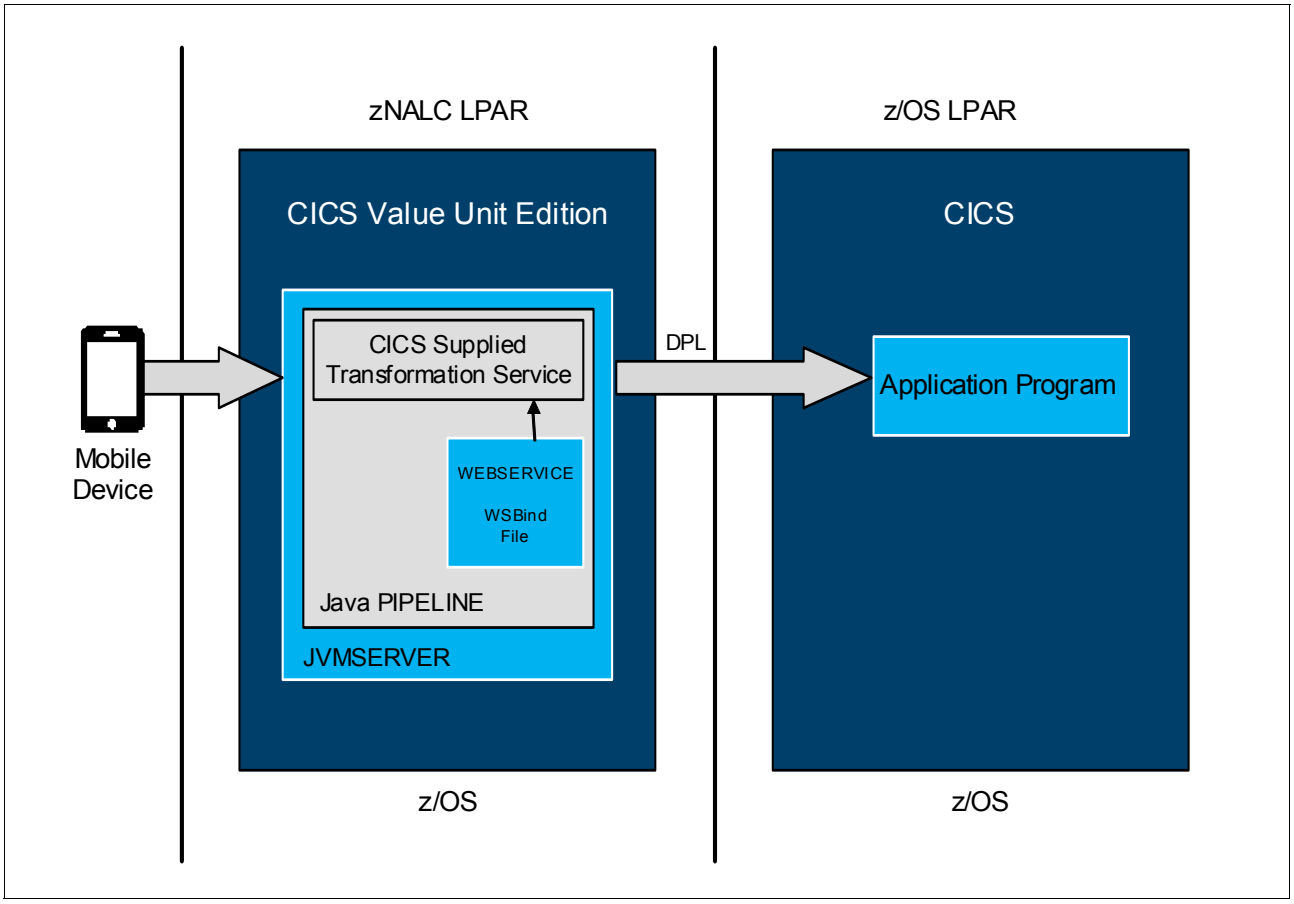


Figure 8-1 Hosting the CICS supplied transformation service in a Java pipeline

8.2 Characteristics of CICS data transformation

The Java based CICS data transformation service facilitates the conversion of XML or JSON into structured application data, and back again. It does this using pre-generated meta-data

(a WSBind file) that instructs CICS in how to transform the data. The meta-data is prepared in advance using tools.

The application development aspects of how this works are discussed further in IBM Redbooks publication *SG24-7126-00 Application Development for CICS Web Services*. It is available here:

http://www.redbooks.ibm.com/abstracts/sg247126.html

The JSON specific characteristics of this process are discussed in IBM Redbooks publication *SG24-8161-00 Implementing IBM CICS JSON Web Services for Mobile Applications*. It is available here:

http://www.redbooks.ibm.com/abstracts/sg248161.html

The CICS supplied data mapping service is popular as most existing CICS programs can be exposed as Web Services (either JSON or XML) without application changes, and without the need for a new wrapper program to be written. The technology involved has also been adapted for other IBM products, including Rational Developer for System Z (which provides application development support for CICS Web Services), CICS Transaction Gateway (which supports hosting CICS WSBind files for JSON from version 9.1), and WebSphere Application Server Liberty Profile (which supports hosting WSBind files for JSON).

The CICS data mapping technology can be used for both *bottom-up* development scenarios, and *top-down* scenarios. And it can be used in both *Provider* mode, and *Requester* mode. For further description of these development scenarios see Chapter 6.4, “CICS Web Service development strategies” on page 49.

The CICS data mapping tools and the regular non-Java data transformation service has been successfully proven in many CICS Web Services customer deployments, since CICS Transaction Server V3.1.

8.3 The Java-based pipeline

A Java-based pipeline in CICS is very similar to a regular CICS pipeline resource (of the type that has been available for hosting Web Services from CICS Transaction Server V3.1). Its main differentiating characteristic is that it is implemented in Java, and is therefore eligible to be considered for approval for IBM CICS Transaction Server for z/OS Value Unit Edition.

Converting a traditional CICS pipeline to a Java-based pipeline involves:

- the configuration of a suitable JVM server in CICS
- minor changes to the Pipeline configuration file
- (optionally) changing any pipeline handler programs to use Java

This can be a convenient mechanism for adapting an existing CICS Web Services infrastructure, though it is unlikely to perform as well as the native CICS implementation. If a JSON interface is required, consider using the capabilities of IBM z/OS Connect, see Chapter 7.2, “z/OS Connect and CICS Liberty” on page 59.

The Java-based pipeline takes XML or JSON data, and automatically converts it to structured application data, using instructions found within the WSBind file. It then LINKs to the target application program passing the transformed data as input, and finally converts the response back into XML or JSON to return to the remote caller. This can provide a significant amount of value for relatively little effort.

8.4 Security considerations

Security of Web Services (both JSON and SOAP) is a broad topic, encompassing many considerations. As with the CICS Liberty scenario in Chapter 7.4, “Security considerations” on page 60, both IBM Worklight and IBM DataPower can be used to provide additional capabilities beyond those natively available through CICS.

CICS supports a wide range of Security protocols for XML based Web Services, including WS-Security, and WS-Trust, and in CICS Transaction Server Version 5.2 this is extended to include aspects of the SAML and Kerberos protocols. CICS also supports the z/OS Identity Propagation protocol.

General security considerations are discussed further in IBM Redbooks publication *SG24-7658-00 Securing CICS Web Services*. It is available here:

http://www.redbooks.ibm.com/abstracts/sg247658.html

The Identity Propagation protocol is discussed in IBM Redbooks publication *SG24-7850-00 z/OS Identity Propagation*. It is available here:

http://www.redbooks.ibm.com/abstracts/sg247850.html

Many deployments of CICS Web Services include the use of IBM DataPower. The DataPower appliance is used for authentication purposes and as an XML firewall. Atrust relationship is configured between CICS and the DataPower appliance using a client certified transport connection, and the userid associated with the DataPower appliance is granted surrogate authority to assert the identity of the end-user. The identity of this user is passed to CICS from the IBM DataPower appliance using WS-Security. This is probably the single most common configuration for authentication with XML based Web Services for CICS: DataPower is responsible for Authentication, and CICS is configured to trust DataPower to supply the identity of the end-user.

8.5 Other considerations

There are many ways to call an existing CICS program from a mobile device. Hosting a transformational service in a Java-based pipeline offers the advantage that the associated transformational work is a candidate for approval for use with IBM CICS Transaction Server for z/OS Value Unit Edition. However, other options are available, including the use of CICS Liberty as described in Chapter 7, “Mobile devices and CICS Liberty” on page 55.

One of the limitations of the Java-based pipeline is that the associated JVM server will not be suitable for hosting OSGi applications or CICS Liberty. It requires a special JVM server of its own.

The transformation service for JSON Web Services may be better hosted in the IBM CICS Transaction Gateway v9.1, or in WebSphere Application Server Liberty Profile.

Part 4

IBM Operational Decision Manager

This part introduces the benefits of decision management to the enterprise. It explains how IBM Operational Decision Manager for z/OS allows existing CICS COBOL and PL/I applications can exploit decision management as a new workload in CICS.

The following chapters are included:

- Chapter 9, “Understanding Decision Management in CICS” on page 67
- Chapter 10, “Implementing Decision Management in CICS” on page 77

9

Understanding Decision Management in CICS

This chapter introduces the concept of Decision Management and describes how IBM Operational Decision Manager for z/OS is a candidate for approval for running inside IBM CICS Transaction Server for z/OS Value Unit Edition to provide Decision Management for CICS COBOL and PL/I applications.

The following topics are covered in this chapter:

9.1 Introduction to Decision Management

Making decisions has always been at the center of business. For instance, banks must decide who they will lend money to, insurance companies must decide who they will insure, retailers must decide when promotions will occur and who will be eligible for them. Every business must make decisions to be successful and today's businesses must make a startling number of decisions every day.

A customer's demand for a flawless, seamless, instant experience means businesses are making more decisions, they have less time to make them, and they need to get those decisions right the first time. This requires decisions that are consistent with business policy and can be made at machine speed, without manual processes and human involvement.

9.1.1 Common business decisions which require managing

There are, generally speaking, three types of decisions found in most businesses:

- Decisions that help increase revenue:
  - This type includes decisions that are used by marketing and sales to make targeted offers based on customer profiles, demographics, and analytical models.
  - For example: Is a customer eligible for a certain promotion or a cross-sell or up-sell opportunity? Should a store discounting the price of a product at the end of the day?
- Decisions around consistency and compliance with regulations:
  - This type of decision can be found in all industries, such as financial, insurance, and government sectors.
  - For Example: Are there prohibitions against a customer buying a certain quantity of a product? Is a customer eligible to make a certain purchase based on where she is located?
- Decisions that reduce and mitigate risk:
  - The third type of decision includes those decisions that help reduce and mitigate risk.
  - For Example: Does the customer who just filled out a loan application online meet the criteria to be approved?

Businesses must make one or all of these types of decisions, many times a day, ensuring they are made correctly and consistently according to their business policies.

9.1.2 Where are most decisions made today?

The traditional approach to decision making requires a business analyst to understand the business policies and create a requirements document which defines the decisions to be made. Then, one or more software developers take the requirements and code or embed the decisions into the various application programs which support the business. The development is then followed by a lengthy testing process before the new decisions become live in production.

Unfortunately, the decisions are now hidden in the code of one or more programs, and over time as additional changes are added to the business policy, the code becomes more complex, making it difficult to change, hard to visualize and nearly impossible to manage. The decisions may change frequently or rarely and changing a program to change the decision, testing it, and getting it into production is not fast enough in today's business environment.

This scenario can be avoided by implementing a Decision Management solution which takes the decisions out of code and places them in a central repository. This makes the decisions more flexible, visible, auditable and manageable. This is illustrated in Figure 9-1:

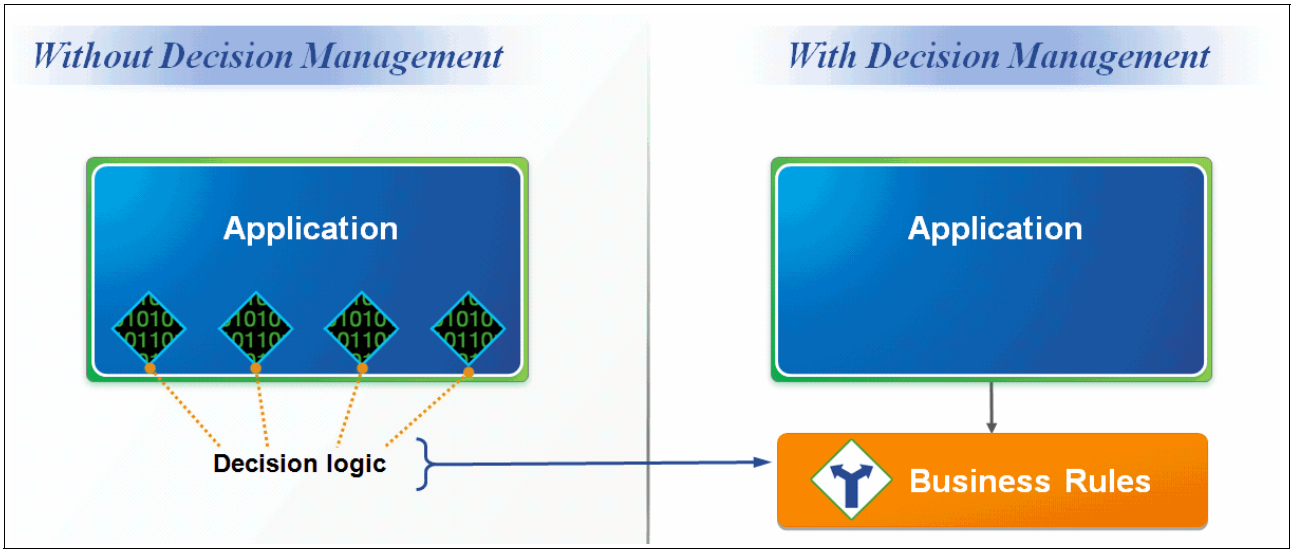


Figure 9-1 The decision logic is moved out of code and into a Decision Manager

Not all decisions are equal and some are more applicable for Decision Management than others. Those decisions which are best suited are:

- Those decisions which must be changed frequently to support the business
  - Decision Management avoids costly application code changes
- Those decisions which are duplicated in multiple applications running on multiple platforms
  - Decision Management implements the decision once and stores it centrally to be called from multiple applications
- Those decisions which must be visible for business purposes
  - Decision Management allows decisions to be shared easily with lines of business or regulatory auditors

The next section introduces IBM Operational Decision Manager for z/OS and explains how it can be used to implement a Decision Management solution on z/OS and specifically in CICS.

9.2 IBM Operational Decision Manager for z/OS

Operational Decision Manager for z/OS provides a smarter way of dealing with business decisions. It is designed to help organizations gain more control over the business decisions that take place in their enterprise applications. Businesses that use Operational Decision Manager for z/OS will simplify their ability to make decision changes to enterprise applications. This enables them to cut costs and cycle times, improve their agility and time to market, and enhance their visibility into business decisions as well as the governance of those decisions.

Operational Decision Manager for z/OS can deliver these advantages to enterprise users because it enables separation of the decision logic from business applications and processes.

9.2.1 Operational Decision Manager components

Operational Decision Manager for z/OS includes four main components which work together to provide a full Decision Management experience:

- Rule Designer is used as the starting point to create the model on which the business decisions are authored. Rule Designer is an Eclipse-based development toolkit for business decisions. It is installed on a Windows or Linux workstation.
- Decision Center is used as a team repository to govern the business decisions, and to author them through a web interface. Decision Center runs on WebSphere Application Server on z/OS, z/Linux, or a distributed environment.
- Decision Server is where business decisions are made. API calls are issued from COBOL and PL/I programs to the Decision Server to make decisions. The Decision Server has multiple deployment options depending on the environment the application program is running in. These environments are discussed further in section 9.2.4, “Execute decisions using Decision Server” on page 74.
- Rule Execution Server console is a web-based graphical interface to monitor Decision Servers and to manage the deployed artefacts such as RuleApps, ruleset archives required libraries. It runs in either a stand alone address space or within WebSphere Application Server for z/OS.

Figure 9-2 shows the interaction between these components: Rule Designer is able to create and synchronize rule projects to Decision Center. Then Decision Center allows business users to modify, test and simulate business decisions. Finally, the decisions are published to the Decision Server using either Rule Designer or Decision Center.

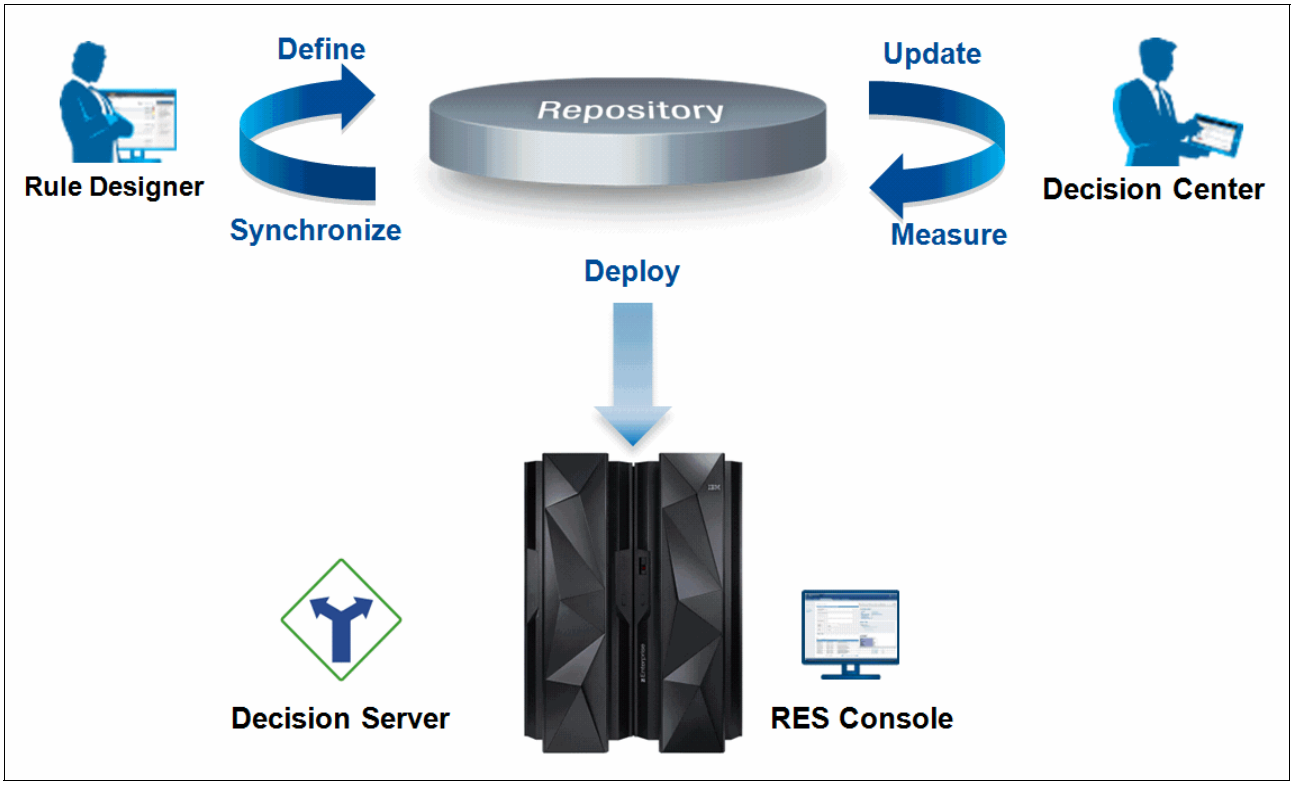


Figure 9-2 Operational Decision Manager components working together

Rule Designer must run on a Windows or Linux workstation but Decision Center is able to run on multiple operating systems which allows for flexible deployment patterns as shown in Figure 9-3.

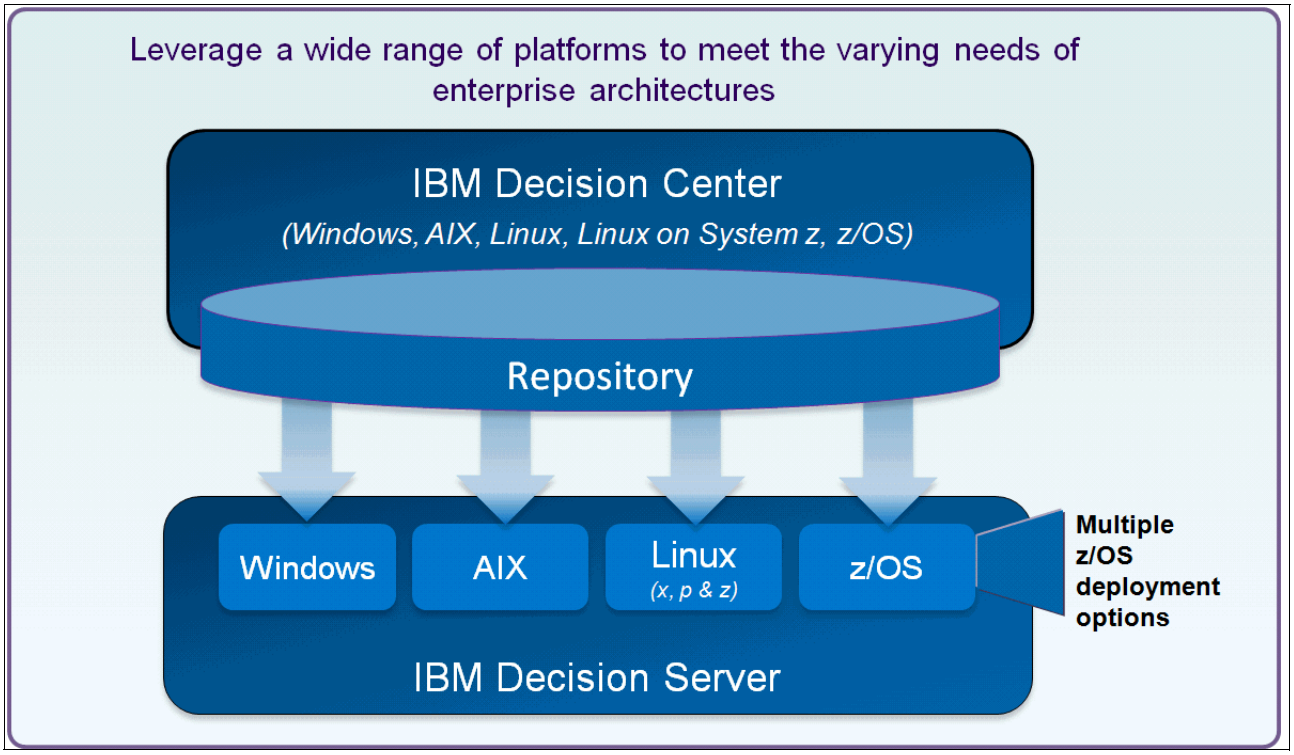


Figure 9-3 Decision Center and Decision Server can run on various operating systems

Note: A recommended approach when using Operational Decision Manager for z/OS is to run Decision Center on a distributed or z/Linux platform and deploy remotely to Decision Server running on z/OS.

In the next section, the key concepts of the Rule Designer, Decision Center and Decision Server are explained in more detail.

9.2.2 Create decisions using Rule Designer

Rule Designer enables the creation of a rule project which contains the Execution Object Model (XOM) and the Business Object Model (BOM). It allows a COBOL copybook or PL/I include file to be used as the starting point for these models as shown in Figure 9-4 on page 72. These models are required to allow the authoring of business decisions based on the business data which is currently being used in the COBOL or PL/I applications.

The Rule Designer can also be used to take an existing rule project based on a Java XOM and enable it for COBOL or PL/I execution. This is achieved by generating the COBOL copybook or PL/I include file necessary for the program to execute the existing rule sets.

Note: Existing rule projects which use a XOM generated from an XSD cannot be enabled for COBOL or PL/I execution.

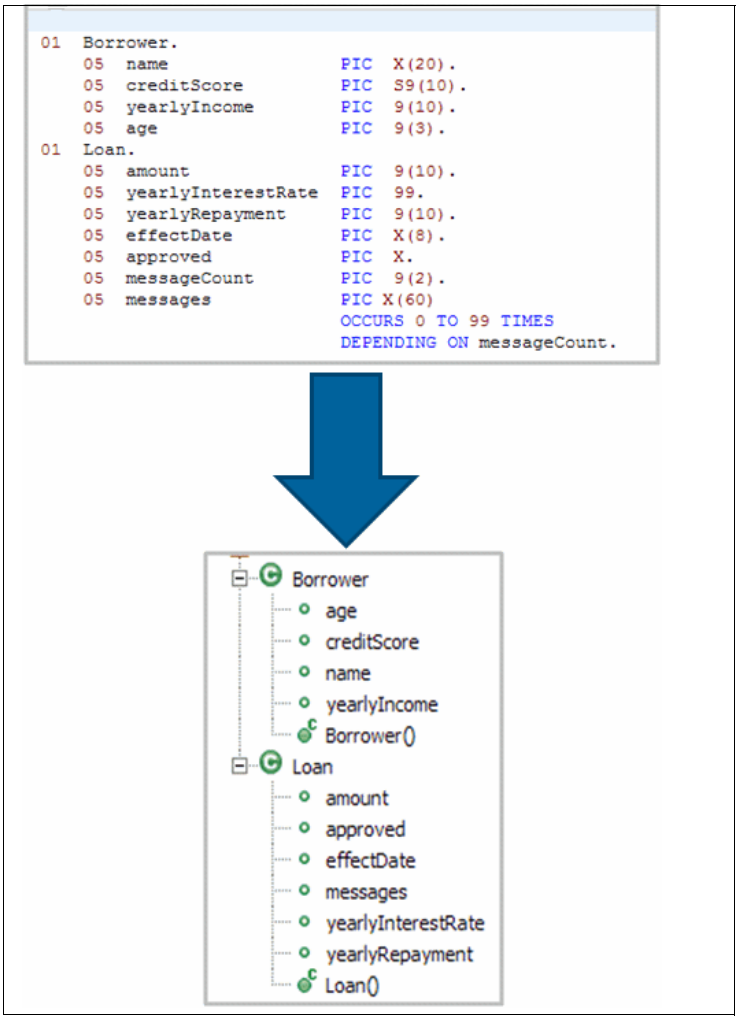


Figure 9-4 Cobol copybook or PL/I include files are used to create the business and execution models

Once the rule project, XOM and BOM are created, the Rule Designer can be used to start authoring the initial decisions. The decisions can be expressed using the Business Action Language (BAL) as shown in Figure 9-5.

```
if
  the amount of 'the loan' is more than 1000000
then
  add "The loan cannot exceed 1000000" to the messages of 'the loan' ;
  reject 'the loan';
```

Figure 9-5 A business decision expressed in Business Action Language

The BAL provides a simple if-then syntax that is used with a vocabulary to write business decisions. The BAL defines the syntax and provides constructs for expressing conditions and actions, and the vocabulary defines the terms that are used in the business decisions.

Decisions can also be entered using decision tables which provide a graphical way to author decisions, as illustrated in Figure 9-6 on page 73.

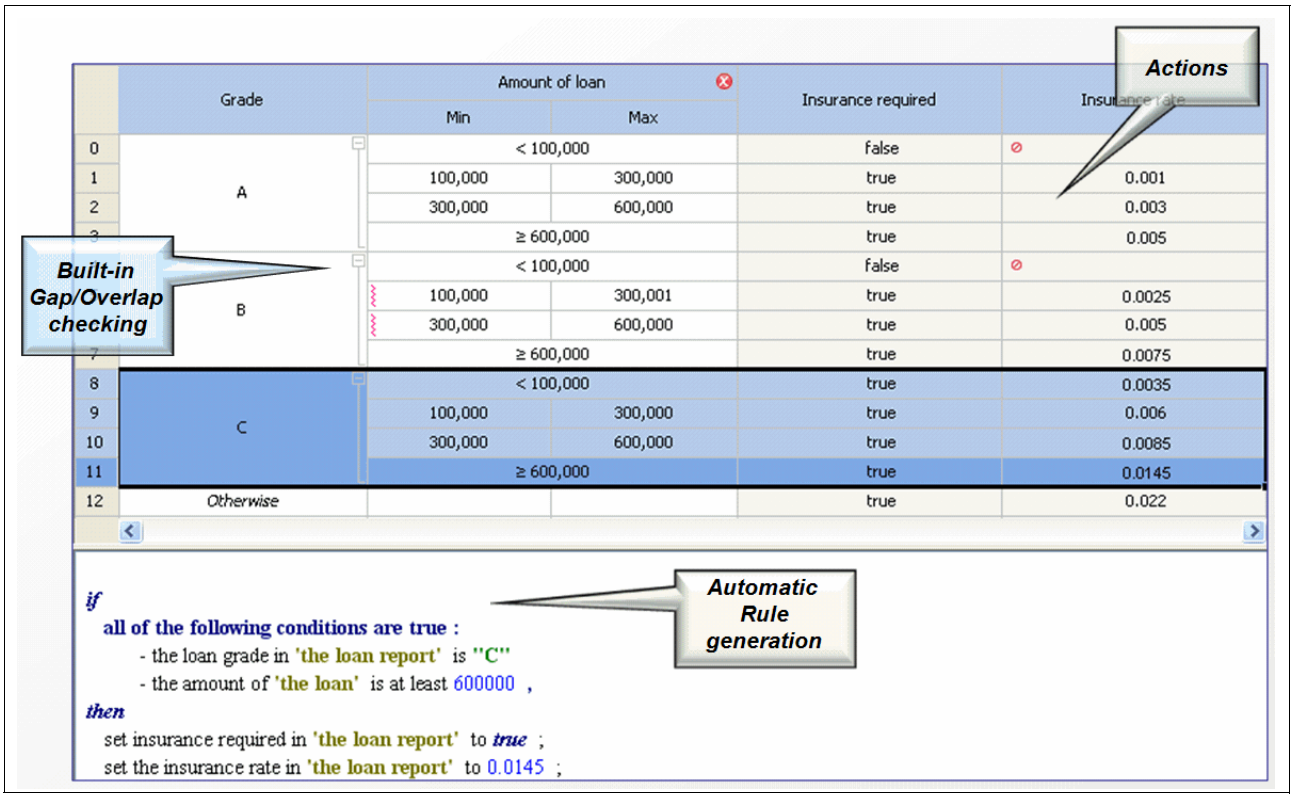


Figure 9-6 Decision tables are a graphical way to author decisions

Rule Designer checks for overlaps and gaps in decision table conditions as values are entered into the cells, and indicates problems using visual cues. As shown in Figure 9-6, when there is a problem, the column header displays a warning symbol and the affected cells are highlighted.

Once the BOM, XOM and business decisions have been created, the rule project can be synchronized to Decision Center for further decision authoring, testing and simulation.

9.2.3 Centrally manage decisions using Decision Center

Decision Center is the central hub that coordinates the decision lifecycle across the business and IT parts of the organization. It provides the ability to author decisions, publish reports on existing decisions, validate decisions using testing and simulation and deploy decisions to the Decision Server.

Decision Center provides web based graphical interfaces which allow business users to achieve these tasks with limited dependence on the IT department. The two graphical environments provided are:

- Business console - The preferred environment for business users to take advantage of social interaction features and change management.
- Enterprise console - The standard environment for administrators, where deployments can be performed and administrative features like project security and permission management are available.

Once decisions have been edited, tested and simulated in Decision Center, and are ready for execution, they must be deployed to a Decision Server.

9.2.4 Execute decisions using Decision Server

Operational Decision Manager for z/OS provides several execution environments in which the Decision Server can run. They are listed here and illustrated in Figure 9-7.

- zRule Execution Server allows COBOL and PL/I applications to execute decisions in a standalone server address space or embedded within the client application's address space.
- zRule Execution Server inside the CICS JVM server allows CICS COBOL and PL/I applications to execute decisions in the same CICS region or a remote Rule Owning Region (ROR).
- Rule Execution Server on WebSphere Application Server for z/OS allows COBOL and PL/I applications to execute decisions via the WebSphere Local Adapter (WOLA) interface.

These execution options are illustrated in Figure 9-7:

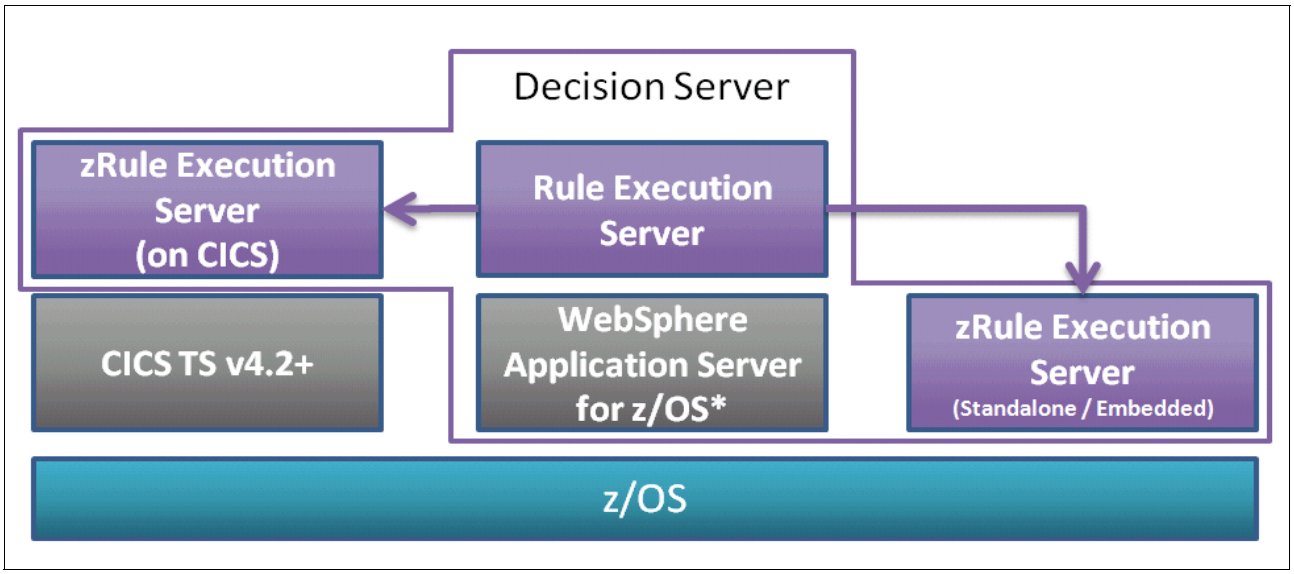


Figure 9-7 Decision Server on z/OS execution environment option

Note: WebSphere Application Server for z/OS is provided with Operational Decision Manager for z/OS as a limited licence product for Decision Center and Decision Server.

The benefits of each execution environment are discussed more thoroughly in the Red Book "Flexible Decision Automation for Your zEnterprise with Business Rules and Events" which can be found here:

http://www.redbooks.ibm.com/abstracts/sg248014.html

The execution environment which can be considered as a qualifying workload for execution within IBM CICS Transaction Server for z/OS Value Unit Edition is the zRule Execution Server running inside the CICS JVM server.

The next section discusses how existing COBOL and PL/I programs running in a CICS Application Owning Region (AOR) can execute decisions in a Rule Owning Region (ROR) running inside the JVM server of a IBM CICS Transaction Server for z/OS Value Unit Edition.

9.3 CICS Rule Owning Region architecture

A CICS Rule Owning Region is responsible for hosting Operational Decision Manager for z/OS inside its JVM server. It is then possible for other CICS regions running on separate LPARs to execute decisions inside the ROR. This architecture is illustrated in Figure 9-8.

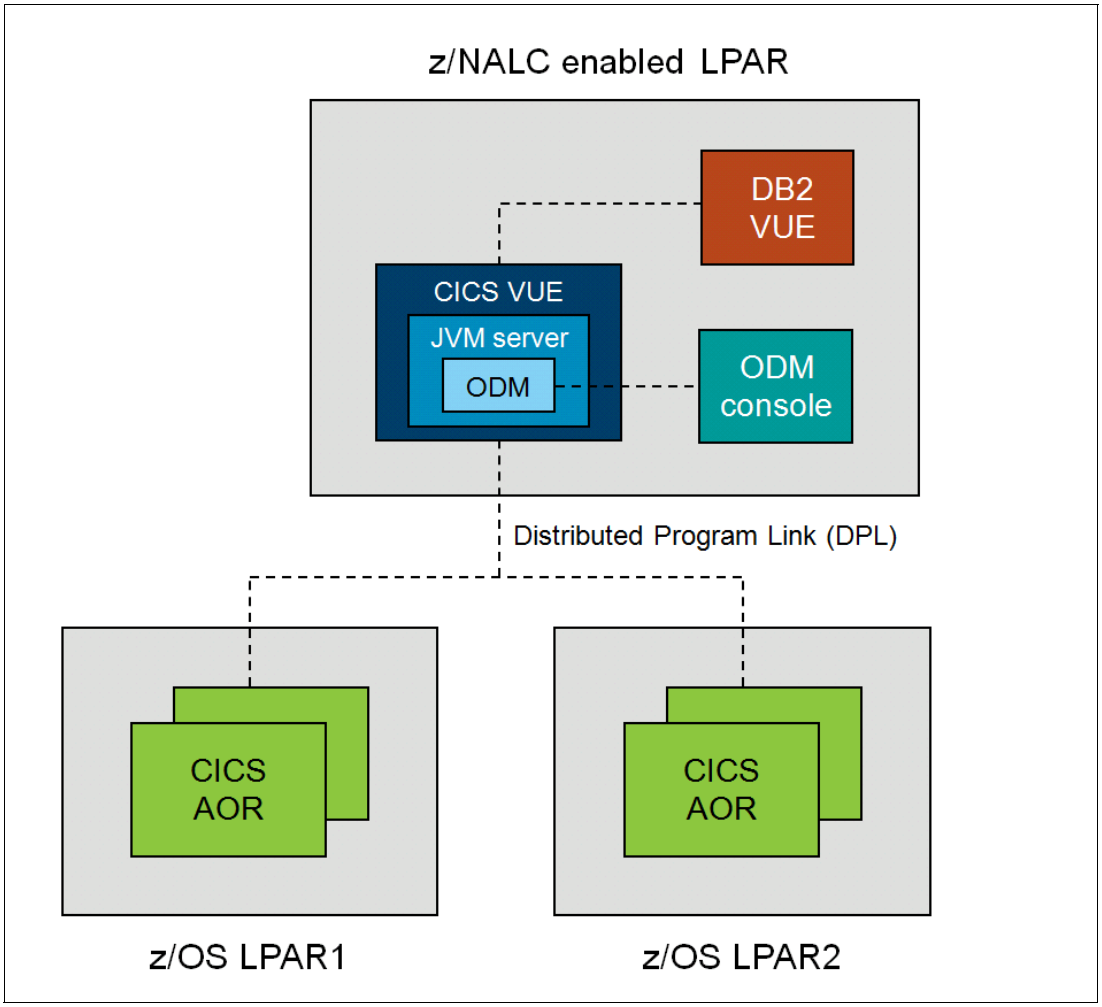


Figure 9-8 CICS AORs executing decisions remotely on a z/NALC LPAR

Note: The ability to configure a rule owning region is available in Operational Decision Manager for z/OS version 8.5.1 onwards.

CICS AORs can communicate with the ROR using a Distributed Program Link (DPL) or by using CICSPlex SM Workload Management (WLM). The use of DPL or WLM allows the decision request to be routed dynamically to the ROR. This provides a highly available and work load managed solution when two or more RORs are used.

Then next section details how this architecture also provides a cost effective way for the COBOL and PL/I applications running in the AORs to execute business decisions using Operational Decision Manager for z/OS running in the ROR.

9.3.1 Cost effectiveness

The cost of Operational Decision Manager for z/OS is based on the size of the LPARs that it is deployed into regardless of whether Decision Server is running in a zRule Execution Server, WebSphere Application Server for z/OS or CICS Transaction Server for z/OS. The pricing of Operational Decision Manager for z/OS is not affected by the CICS pricing model (VUE or MLC). The products are sold and priced independently.

However, IBM CICS Transaction Server for z/OS Value Unit Edition offers a unique way to contain the cost of Operational Decision Manager for z/OS for CICS applications running in the AORs through two means:

- The CICS Rule Owning Region allows the isolation of the costs of ODM into a single LPAR
  - There is no cost for the Operational Decision Manager for z/OS client libraries running on the AOR. The client libraries provide the API required to execute rules on the ROR.
  - Multiple AORs can route decision requests to Operational Decision Manager for z/OS running in the ROR
- z/NALC LPARs are separate from z/OS LPARs and are often smaller in size. Therefore the cost of Operational Decision Manager for z/OS is reduced

Operational Decision Manager for z/OS also requires a DB2 database to store the runtime artefacts and also to support runtime warehousing features. Another benefit of running Operational Decision Manager for z/OS inside a z/NALC LPAR is that DB2 for z/OS Value Unit Edition can be used to provide this persistence layer, further reducing the cost of implementing Decision Management on z/OS.

9.4 Decision Management summary

Making business decisions quickly and correctly is vital for a successful business. This chapter has shown that taking business logic out of application code and storing it centrally in a Decision Management solution increases the agility, visibility and maintainability of business logic.

The recommended method for implementing Decision Management on z/OS is to use IBM Operational Decision Manager for z/OS which can support both COBOL and PL/I applications running in Batch, CICS or IMS. Operational Decision Manager for z/OS can also be deployed into multiple execution environments so that it can be as close to the applications as possible.

Further more, Operational Decision Manager for z/OS is Java based and can execute in the CICS JVM server. The movement of existing business logic from COBOl or PL/I code into Operational Decision Manager for z/OS is also considered as a net new workload. Therefore it is a strong candidate for approval for execution within IBM CICS Transaction Server for z/OS Value Unit Edition. This solution provides a cost effective way to enable Decision Management for existing CICS COBOL and PL/I applications. Finally, the use of a Rule Owning Region architecture allows multiple Application Owning Regions to execute decision requests without the cost of running the Decision Server locally on each AOR.

In the next chapter a real world loan application scenario is introduced. The business rules are extracted from the existing COBOL program, coded into rule sets using Rule Designer and are then deployed to Operational Decision Manager for z/OS running inside IBM CICS Transaction Server for z/OS Value Unit Edition. Finally, the API calls are made from the COBOL program to execute the business logic in the Decision Server.

10

Implementing Decision Management in CICS

This chapter looks at implementing Decision Management in a CICS integration scenario based on a messaging infrastructure using WebSphere MQ. We review how Business Rules are implemented, deployed and executed inside a IBM CICS Transaction Server for z/OS Value Unit Edition region. The scenario discusses a loan approval process.

10.1 Objectives

Many banks have CICS-based core banking services, such as loan approval process, that require real-time near instant decisions. In addition, the competitive nature of the banking industry means that banks have to quickly adapt to changes in the marketplace. This often poses challenges as business rules are embedded in application code and any changes take a long time and may casue a financial impact, especially during peak holiday season.

IBM Operational Decision Manager for z/OS enables a business to respond to real-time data with intelligent, automated decisions. With IBM Operational Decision Manager for z/OS, IT and business users alike can manage the business decision logic that is used by operational systems within an organization.. The Business Rules are developed and deployed to an Decision server running inside the CICS JVM Server. CICS based COBOL and PL/I applications can easily invoke the deployed decision service inside CICS with the use of simple API's. Business Rules are changed and deployed without the need for application changes. The use of IBM Operational Decision Manager for z/OS enables this type of loan processing processing solution to be highly agile, available and scalable.

10.1.1 Solution requirements

Table 10-1 shows how the use of IBM ODM server deployed in a CICS JVM server meets the major requirements of the loan processing project.

Table 10-1 Project requirements

Requirement	Solution
Ability to change decision logic quickly and frequently for greater business agility	Application modernization by incrementally externalizing business rules from COBOL and PL/I applications and moving them into IBM ODM
Ensure implementing ODM does not increase costs	IBM ODM server running inside CICS JVM Server is eligible to run in CICS VUE regionsl
High availability	Because applications can connect to any CICS VUE region in a CICSPlex environment, client applications are not dependent on the availability of a specific CICS region.
High scalability	The use of a CICSPlex enables a scalable solution that takes advantage of the resources across the parallel sysplex. New instances of CICS VUE regions can be easily introduced into the CICSPlex as business growth dictates.
Workload balancing	The CICSPlex automatically enables full workload balancing.

10.2 Architecture

The key feature of running the Decision Server inside the CICS JVM Server is its availability across the sysplex. This enables CICS transactions to run in multiple Application Owning

Regions (AOR) and access the rules running in multiple Rule Owning Regions (ROR). This is illustrated in the high level architecture for this project shown in Figure 10-1. <<Diagram with graphics team >>

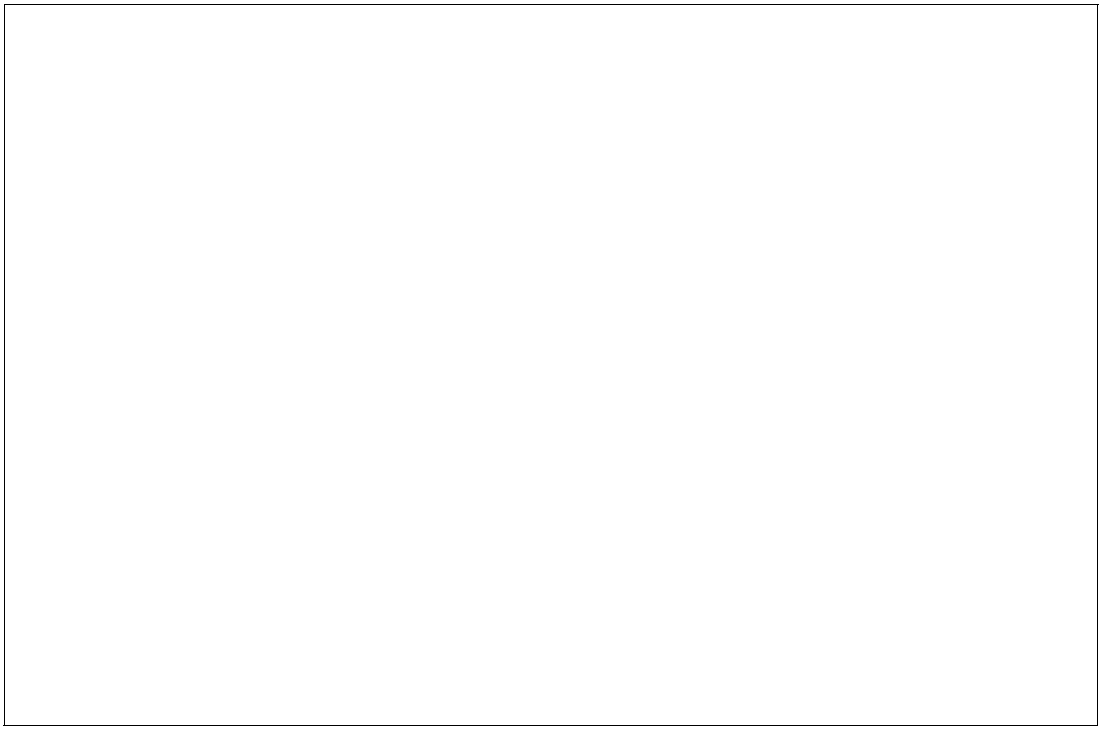


Figure 10-1 IBM ODM in CICS JVM Server scenario

In << figure xref>> we can see how multiple instances of the loan processing application can be running in AORs across the sysplex, all processing messages from the same shared-request queue to maximize throughput using a request reply messaging pattern. The scenario consists of loan processing request arriving from multiple channels in a Shared Queue. It triggers the loan processing CICS transaction in the AOR.

The loan processing transaction needs to call a decison service for loan eligibility. The decison service has been deployed in the ODM Decision Server in the CICS VUE region identified as an ROR. The transcation in the AOR dynamically routes the request to the ROR using CICSPlex SM Workload Manager (WLM). The use of WLM allows the decision request to be routed dynamically to the ROR. This provides a highly available and work load managed solution.

10.3 Implementation

The configuration consists of a set of cloned CICS AOR regions spread across two LPARs in a parallel sysplex environment. The configuration also consists of a set of cloned CICS ROR regions in zNALC LPAR's. The CICS AOR regions share access to a request queue that is held in the coupling facility.

10.3.1 Rule Application Development

How does the bank implement a rule based system? There were multiple phases in the implementation of a rule based system

- Rules Discovery, Analysis and Design

  The first plase consisted of rules harvesting, analysis and design. The eligibility rules are embedded in the customers CICS COBOL application. The application team and the business analysts worked together in this phase. A sample code snippet of eligibility rules are shown in Figure 10-2

```
*****************************************
 P1000-VALIDATE-ELIGIBILITY.
*****************************************
     MOVE 'Y'       TO WS-LOAN-APPROVAL
     IF WS-LOAN-AMT > 1000000
        MOVE 'N'    TO WS-LOAN-APPROVAL
     END-IF
     IF WS-CUST-AGE > 65
        MOVE 'N'    TO WS-LOAN-APPROVAL
     END-IF
     IF WS-CUST-CREDIT < 200
        MOVE 'N'    TO WS-LOAN-APPROVAL
     END-IF
```

Figure 10-2 Eligibilty code

During the discovery phase, duplicate rules were found embedded in the appliication code. The business team validated the rules and worked in conjunction with the Information Technology (IT) team.

The existing program used a generic copybook for the entire application and was 5 KB long. The loan processing rules only used certain attributes to make a decision and it is not a good practice to send unwanted data to the rules engine. A new copybook was built for the loan processing ruleset. The COBOL copybook layout was designed for parameters to ODM. The layout of the newly designed COBOL COPYBOOK is shown in Figure 10-3.

```
01 Borrower.
  05 name PIC X(20).
  05 creditScore PIC 9(10).
  05 yearlyIncome PIC 9(10).
  05 age PIC 9(3).
    88 teenager VALUE 0 THRU 17.
    88 adult VALUE 18 THRU 60.
    88 retired VALUE 61 THRU 150.
01 Loan.
  05 amount PIC 9(10).
  05 yearlyInterestRate PIC 99.
  05 yearlyRepayment PIC 9(10).
  05 effectDate PIC X(8).
  05 approved PIC X.
  05 messageCount PIC 9(2).
  05 messages PIC X(60)
                        OCCURS 0 TO 99 TIMES
```

Figure 10-3 Rules parameters

- Rules Authoring and Testing

  Once the rules are designed, authoring is done through Rules Designer or Decision Center. The rules are then choreographed into a rule flow to be exposed as a loan eligibility decision service. Business rules can be authored with

  – Action Rules
  – Decision Tables
  – Decision Trees

  A detailed explanation of authoring business rules can be found at:

  http://www-01.ibm.com/support/knowledgecenter/SSQP76_8.6.0/com.ibm.odm.dserver.rules.designer.author/authoring_topics/tpc_authoring_brules_intro.html?lang=en

  A sample of the “Minimum Credit Score” rule is shown in Figure 10-4. This is an example of an action rule.

```
if
    the credit score of 'the borrower' is less than 200
then
    add "Credit score below 200" to the messages of 'the loan';
    reject 'the loan';
```

Figure 10-4 Credit Score Rule

Once the rules are developed and packaged, they are choreographed into a rule flow as shown in Figure 10-5. This shows how a loan eligibilty decision service is authored.

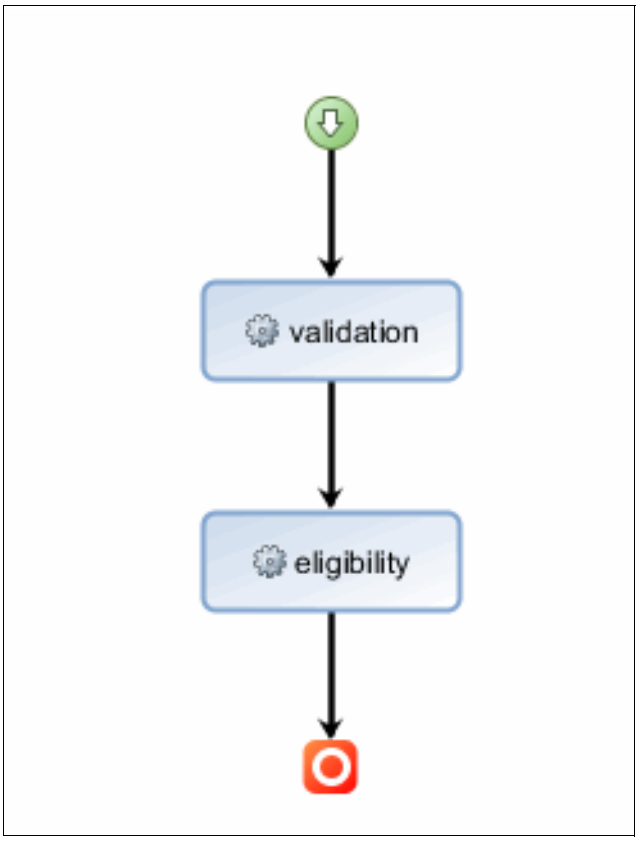


Figure 10-5 Eligibility Decision Service Rule Flow

The next step is the testing of the ruleset so that it gives the desired results. Decision Validation Services (DVS) is used for testing the accuracy of the rules. The tests are executed in either Decision Center or in Rule Designer, which provides added debugging capabilities. Test scenarios are built with input variables and expected results. A scenario represents the values of the input parameters of a ruleset, that is, the input data to ruleset execution, plus any expectations on the execution results that you want to test. The scenaruis are built in an Excel format as shown in Figure 10-6

5		the borrower				the loan		
6	Scenario ID	first name	last name	credit score	yearly income	duration	amount	rate
9	Big Loan	John	Smith	600	80000	24	500000	5
10	Small Loan	John	Smith	600	80000	24	25000	5

Scenarios HELP

Figure 10-6 Scenario

10.3.2 Runtime configuration

Need text explaining this area....

- CICS Setup

  This topology uses two different types of CICS region (the rule-owning region and the application-owning region) to run rules. The rule-owning region hosts a zRule Execution Server for z/OS instance that runs locally in a CICS JVM server. The rule-owning region is

enabled so that rules applications can start rules in the zRule Execution Server for z/OS instance. The application-owning region uses a CICS Distributed Program Link (DPL) on WLM to run rules in a rule-owning region. The rule-owning region requires a Rule Execution Server console that runs in a separate address space from the regions and has a unique subsystem identifier (HBRSSID). A DB2 database provides the persistence layer. The rule-owning region must be on CICS Transaction Server V4.2 or later. The application-owning region must be on CICS Transaction Server V3.2 or later. The following Figure 10-7 depicts the topology.

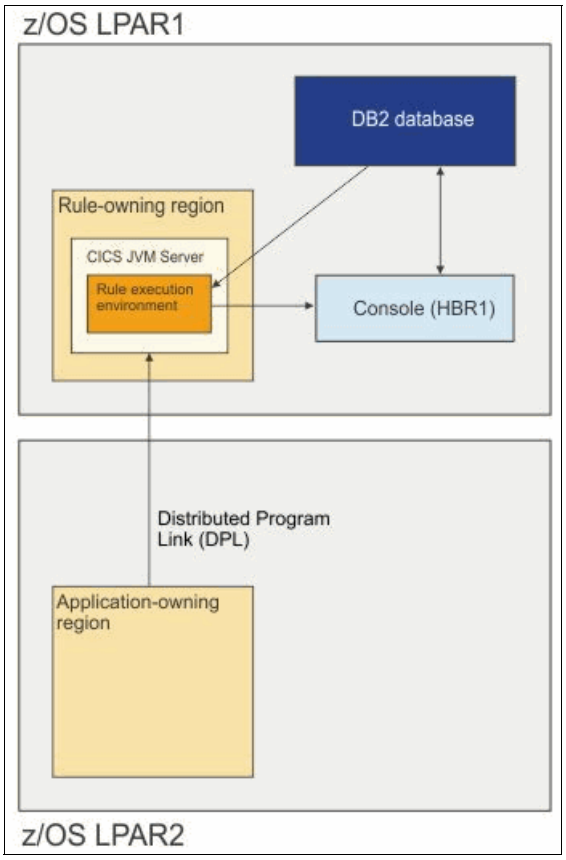


Figure 10-7 CICS AOR and ROR topology

A detailed CICS setup for ROR and AOR can be found at the knowledge center at the following URL:

http://www-01.ibm.com/support/knowledgecenter/SSQP76_8.6.0/com.ibm.odm.zos.config/topics/tsk_ds_config_topol_2_3.html

From a CICS TS infrastructure standpoint, the following was done to configure the topology

– CICS ROR Setup

- Install ODM using SMP/E
- Customize the above ODM topology to create the ROR JCL.
- Run ODM JCL to create the JVM Profile and working paths in HFS or ZFS.
- Enable the CICS started task JCL for DB2 by adding the DB2 libraries in the STEPLIB concatenation. The LE library and ODM library SHBRCICS is added to the DFHRPL concatenation.
- Define CICS resources by submitting the HBRCSD and HBRCSDJ JCL.

- Edit the CICS System Initialization Table (SIT) parameters for JVMPROFILEDIR and GRPLIST
- Copy the profile HBRJVM from the workpath location to the JVMPROFILEDIR directory for the CICS region.
- Start CICS and issue the HBRC transaction to initialize Decision Server in CICS.

– CICS AOR Setup

- Customize the above ODM topology to create the AOR JCL.
- Edit the ODM JCL HBRCSD so that HBRCJVMS program is defined as a remote server program and submit the job.
- Edit the SHBRPARM(HBRCICSZ) parm member. Change the HBRTARGETRES variable to RCICSJVM (for remote execurion)
- Edit the CICS started task JCL to add ODM library SHBRCICS to the DFHRPL concatenation. Also add a new HBRENVPR DD card with the data set members SHBRPARM(HBRCICSZ) and SHBRPARM(HBRCMMN).
- Start CICS and issue HBRC transaction.

► Deployment and Integration

We will introduce the concept of the Rule Execution Sever Console here. A console is a started task on z/OS and there is only one console for the above topology. The Rule Execution Server console provides a web-based graphical interface that you use for managing and monitoring RuleApps, ruleset archives, Java XOMs, and transparent decision services. It can also be used to monitor execution traces from Decision Warehouse, test ruleset execution, manage transparent decision services, and view server information. The following Figure 10-8 shows a screen shot of the Rule Execution Server console.

Figure 10-8 Rule Execution Server console

Once the RuleApp is packaged, it is then deployed to all of the zRES servers through the console. There are many ways to deploy the RuleApp. For additional documentation, please see the following:

http://www-01.ibm.com/support/knowledgecenter/SSQP76_8.6.0/com.ibm.odm.dserver.rules.deploying/topics/con_dserver_deploying_intro.html?lang=en

The next step is to integrate the rule execution inside the CICS COBOL program. The existing code was remediated and an API call call was introduced as shown below in the following Figure 10-9

```
*
-
     MOVE "/LoanRuleApp/Loan" TO HBRA-CONN-RULEAPP-PATH
     MOVE WS-CUST-NAME        TO NAME
     MOVE WS-CUST-CRE-SCORE   TO CREDITSCORE
     MOVE WS-CUST-YRL-INCOM   TO YEARLYINCOME
     MOVE WS-CUST-AGE         TO AGE
     MOVE WS-LOAN-AMT         TO AMOUNT
     MOVE WS-LOAN-INT-RATE    TO YEARLYINTERESTRATE
     MOVE WS-LOAN-YRL-RPAY    TO YEARLYREPAYMENT
     MOVE WS-LOAN-EFF-DT      TO EFFECTDATE

     MOVE     'T'              TO APPROVED
     MOVE     0                TO MESSAGECOUNT
* Invoke the rule
     CALL 'HBRRULE' USING HBRA-CONN-AREA
```

Figure 10-9 Rule Execution API Call

The value proposition with the ODM approach is the agility. If the business rules are modified and deployed to the Decision Server, the above piece of code remains the same.

- Figure 10-10 shows the bank's chosen implementation of the CICS AOR and ROR in a CICSPlex environment.

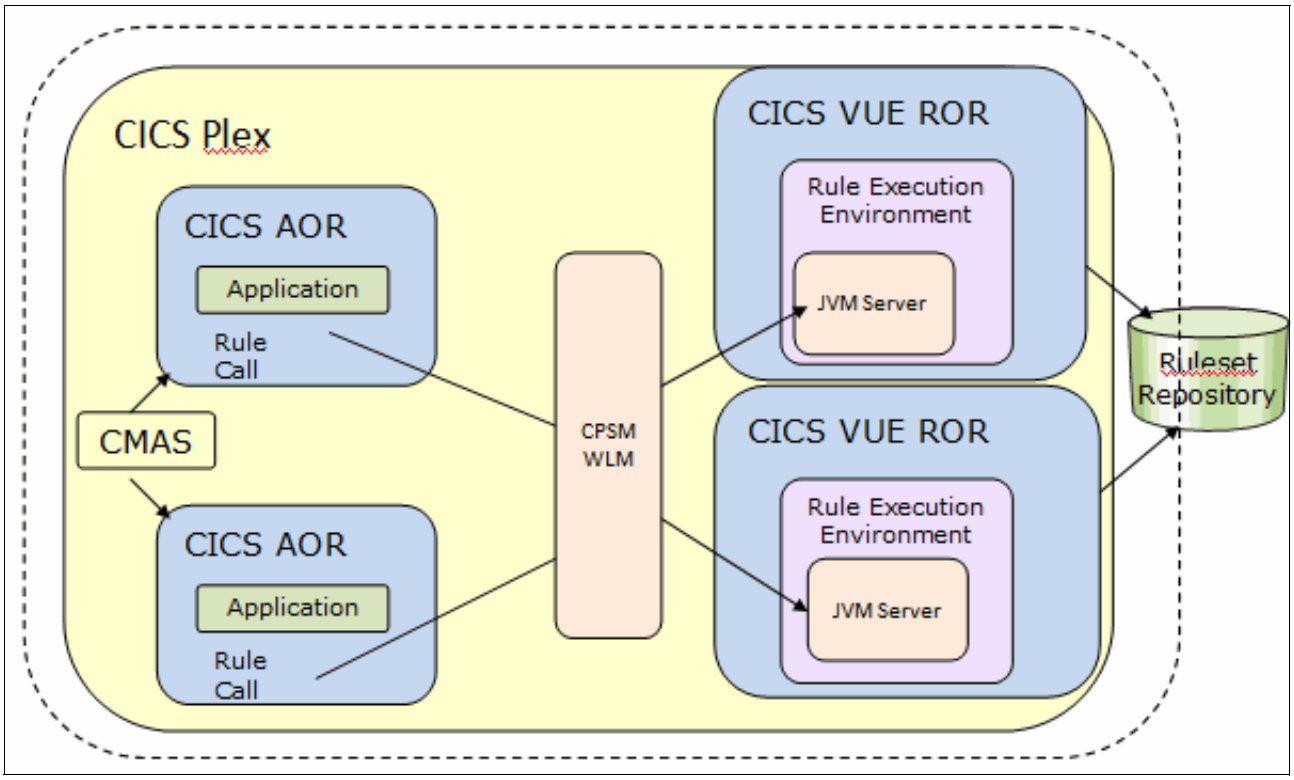


Figure 10-10 CICS and ODM High Availability configuration

The configuration used in Figure 10-8 consists of the following components:

1. 2 CICS AOR's
2. 2 CICS ROR's

10.4 Solution summary

The use of ODM inside a CICS JVM Server with a queue sharing group provides a scalable solution for loan processing and ensures maximum availability for planned and unplanned outages.

Part 5

Modern Batch

Need statement

11

Modern Batch

This chapter discusses the CICS Transaction Server for z/OS (CICS TS) Feature Pack for Modern Batch. This capability can be installed into CICS TS V4.2 or higher and enables the WebSphere Batch Environment to schedule and manage batch applications in CICS.

This chapter first introduces the need for a “Modern Batch” environment. It then reviews the types of workloads that it is suitable for and the architecture of the overall solution. The key design considerations when building a batch application to run in CICS TS are highlighted.

We discuss the following:

11.1 Business pressures on Traditional Batch

Business models have changed and the need is to have access to data near real time. The traditional Batch model is under constant pressure from busines to change into a realtime model. Some of the reasons are described in the following sections.

11.1.1 Concept of “Dedicated Batch” window going away

There used to be a time when “batch” and “online” processing were quite separate from one another. It used to be the case that online processing was stopped altogether so batch processing would have access to the system resources to complete its work.

But those days are well behind us. Online processing is becoming more and more a 24 x 7 operation. This is because client access is increasingly global, with people from different time zones seeking access at all times during the day. There may be times during the day when online processing is greatly reduced, but in most cases it is never stopped altogether. That means batch processing must work at the same time as online. The window of time available for dedicated batch processing is shrinking. The following Figure 11-1 shows the batch processing today and the shrinking batch window.

Figure 11-1 Batch Processing today

The advent of mobile devices means client access is now even more frequent than before. The transaction work that flows back to information processing systems because of mobile device activity is increasing. And mobile device activity may occur at any time of night or day. It is truly a 24 x 7 world for online processing. But batch processing has not gone away. There are still requirements to do batch work. So it is evident batch and online must be done at the same time and in a manner that implies both work cooperatively.

11.1.2 The value of shared services

It is not just that the batch window is shrinking, it is also the cost pressures on maintaining the batch and Online Transaction Processing (OLTP) environments. In addition to the batch window shrinking, there are pressures to contain and reduce costs related to information processing as well. Having separate infrastructures for online and batch, needs separate computers, separate tools and separate support staff. The following Figure 11-2 shows the efficiencies of consolidation of Batch and OLTP staff

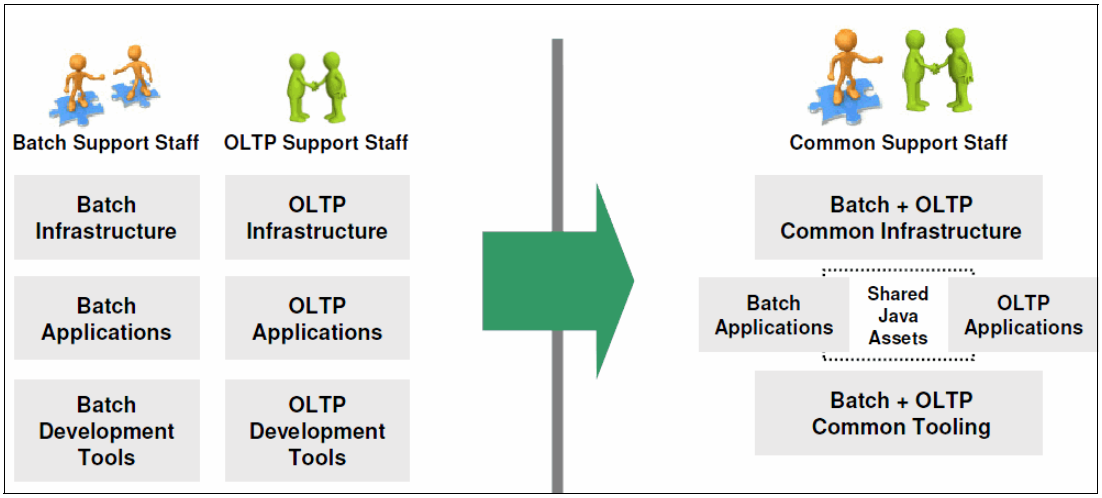


Figure 11-2 Efficiencies through consolidation

The trend is in the direction of convergence. Online and batch processing are both information processing, and as such can and should be handled in a cooperate, converged manner. This allows for efficiencies in infrastructure, staff, development tooling, and potentially Java assets shared between OLTP and batch.

11.1.3 Java for Batch Processing

Some may ask the question "Why use Java when COBOL works well?"

First, the question is not Java or COBOL, but not both. There is a very practical realization in the world that batch assets written in COBOL are still very useful. To the extent Java enters the discussion, it is to complement COBOL, not a total replacement.

Second, there are business pressures that make Java an attractive batch solution:

1. Skills: Java skills are simply more plentiful than COBOL skills. Many organizations have very good Java development skills. It makes sense to leverage those Java skills for batch work as well.
2. Tooling: Today's development tooling products are powerful and quite sophisticated. Acquiring them also represents an investment in technology and an investment in skills. It makes good sense to leverage that investment across multiple information processing disciplines.
3. CICS VUE: Java Batch workload running inside a CICS JVM Server is eligible for CICS VUE pricing. This attractive pricing model is a financial motivation for running Batch inside CICS.
4. Specialty Engines: COBOL runs on general processors, Java runs on specialty engines. Specialty engines (zIIPs) provide a way to lower overall System z costs both in acquisition

costs as well as ongoing licensing costs. Specialty engines are an attractive solution and leveraging them for batch work is desirable.

5. Cooperative Processing: Online processing runtime infrastructures are very often designed around Java. For example, CICS Transaction Server for z/OS is a powerful online processing runtime. There is an investment in maintaining that online infrastructure. So to the extent that same investment can be leveraged for batch work as well.

The bottom line is that Java for batch processing is very much an area of growing interest, and in fact is in use in many very large processing operations.

11.1.4 Conflicting needs of CICS applications and z/OS batch applications

It is common for an online CICS application to update resources. For some resources, such as VSAM files, CICS maintains a record and image lock to prevent other applications from making conflicting updates and to be able to restore records in case of a failure. In these cases, the resources need to be opened exclusively by CICS.

A traditional z/OS batch application can be defined as a job written in job control language (JCL) that is submitted to the z/OS job entry subsystem (JES) for execution and does not need user input to complete. For example, it reads all records from a VSAM input file and, for each one, updates a VSAM master file and creates a summary report. However, the master file may be opened for exclusive use by CICS and thus be unavailable to the batch application.

In this scenario, there are a number of choices:

- Close the master file in CICS and start the batch application, which starts by taking a backup of the master file.

  Then process all the records, make the updates, delete the backup and re-open the master file in CICS. If the batch application fails, the backup of the master file is restored, and either the issue is fixed immediately and the batch application is re-started, or the issue is fixed at a later time.

  This choice results in a period of time, referred to as the *batch window*, during which the master file and the online applications that use that file are not available.

- Code the batch application to send a request to an online CICS application to make each update to the resources locked by the online application.

  If each request is committed individually, for example, using the CICS non-transactional External Communications Interface (EXCI), this will cause data integrity issues if the batch application fails, because if the batch application is re-started, some updates will be executed twice. If all requests were committed together, for example, using transactional EXCI, this can result in many records being locked by the online application for an extended time, causing unacceptable delays for other applications.

- For VSAM resources, use Record Level Sharing (RLS) to maintain record locks for the batch application. However, the batch application is unlikely to maintain its own recoverable logs due to the complexity of writing them. Therefore, if the batch application fails, the records already updated will not be restored, leading to data integrity issues.

- For VSAM resources, use DFSMS Transactional VSAM services to lock and log record images before updates. However, unless the application implements its own checkpointing, many records can be locked for an extended period, causing unacceptable delays for other applications.

In addition, as companies provide their services across more geographies and time zones, and customers require services at times of day to suit them, there is a growing need for online

applications to be available continuously, 24x7. Also over time, batch applications are expected to process an increasing amount of data and there is a need to drive down costs. Therefore, in the event of the batch application failing, it is unacceptable to restart it from the beginning. Instead there is a requirement to restart from a frequent checkpoint.

11.2 WebSphere Java Batch and Batch Container Services

In this section we discuss the Batch environment, WebSphere Java Batch and Batch container framework.

11.2.1 What is a Batch Environment ?

The Batch Environment is a managed environment for batch applications that are scheduled, process large amounts of information, and may take many hours to complete. The Batch Environment provides a powerful failover model based on checkpoint and restart scenarios. This is essential to efficiently manage, run, and restart batch applications, in particular when batch application resources are shared with online transaction processing.

The Batch Environment has two primary components:

- The Job Scheduler is responsible for determining when and where to dispatch a job, monitoring the job, and reporting back to its caller.
- Endpoints (Batch Containers) are where the batch application runs. Jobs are dispatched into an Endpoint from the Job Scheduler. The job runs and, upon completion, the job log and return code is provided back to the Job Scheduler.

11.2.2 What is CICS ?

CICS is a modern general-purpose transaction processing environment for online applications that start as a result of a request received through a terminal, web service, or message. It typically processes a small amount of information within subseconds. CICS provides the following capabilities:

- Administration, security, and transaction facilities, such as authorization, data integrity, workload management, logging, tracing, debugging, statistics, and monitoring
- API and development tools, such as named counter and XML conversion
- Shared access to resources, such as temporary storage, data tables, IBM DB2®, IBM IMS™, and VSAM
- Communications, such as web services, WebSphere Optimized Local Adapters (WOLA), IBM MQ, HTTP, and sockets

11.2.3 WebSphere Java Batch

Lets first understand the high level architecture of WebSphere Java Batch. The WebSphere Application Server (WAS) product provides what is known as a "Java EE" (or "Enterprise Edition") runtime. Java EE is an industry standard specification for an application server that provides a long list of standard application specifications. Part of the design of this "Java EE" runtime is the idea of a "container." Containers are simply runtime functions that provide managed services to the programs that run in the containers. Container-managed services mean the application programs themselves may focus on their core business logic and not

have to implement common functions. WAS by itself has Web and EJB containers. The addition of the WebSphere Java Batch function adds a third batch container.

Like other containers, the batch container provides function services to the batch applications that run in the container. The CICS Modern Batch Feature pack extends WebSphere Java Batcn management and execution realm. It allows CICS TS to participate as a WebSphere Java Batch endpoint server. The Figure 11-3 shows a summary listing of some of those services provided by the Batch Container of WebSphere Java Batch.

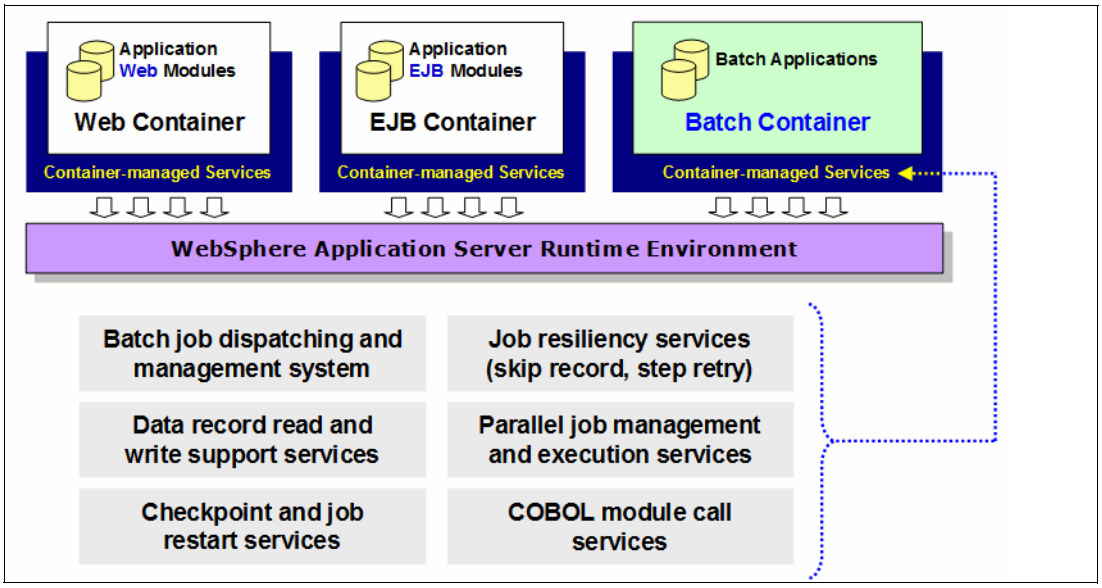


Figure 11-3 WebSphere Batch Container

The next thing to understand is the job management and execution model provided by WebSphere Java Batch. The WebSphere Java Batch provides a separation of several key elements of batch processing as shown in Figure 11-4:

- Job submission: This is done through a defined interface called the Job Management Console (JMC). It provides a view into the batch environment and allows you to submit and manage jobs
- Job description: The job description is specified in an XML file called xJCL. This avoids hard-coding job properties into the application code. The job properties file is used to tell the job submission function what the job is and how to run it. We will describe xJCL later in the chapter.
- Job Dispatching: The Job Dispatching function signals to the endpoint to begin execution of the batch code named in the job declaration file (xJCL). The Job Dispatching function interprets the xJCL, dispatches the job to the endpoint where the batch application resides, and provides ability to stop and restart jobs.
- Job execution: The execution of the job takes place in an "endpoint," a batch container where the Java batch application is deployed.
- Job development and deployment: Java batch applications implement the batch logic, and they are deployed into the batch containers. The development libraries and tooling assist in the creation of the batch applications

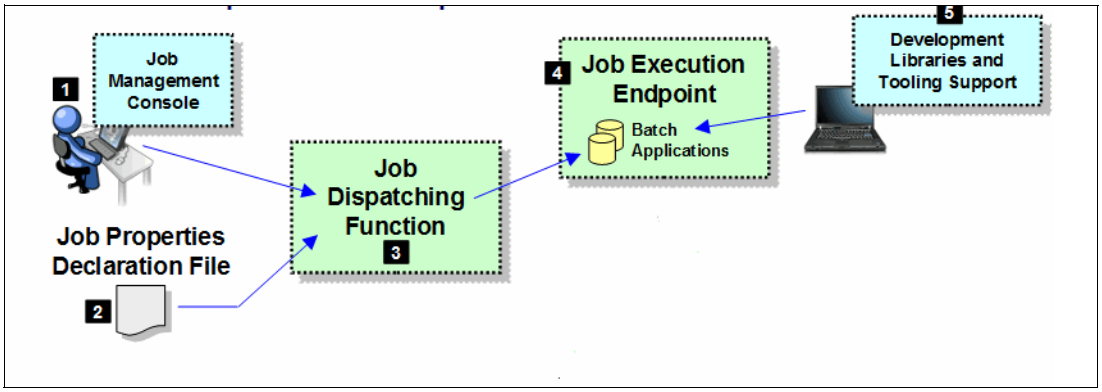


Figure 11-4 Job Management and Execution Model

In the following section, we will now describe the job control language, integration with Enterprise Schedulers and the batch middleware framework that are supported in CICS TS Feature Pack for Modern Batch.

11.2.4 Job control language

The job control language for modern batch is an XML file called xJCL. It describes the Java class files that are used in a batch step and the steps that are included in the batch job. The first thing to understand is the concept of xJCL. What is xJCL? xJCL is a job declaration file. Conceptually, xJCL is just like a normal JCL. It describes the job to be run and the context in which the job is to operate. The difference is xJCL is written in XML. The xJCL file is used to tell the batch runtime that a job invocation is being requested, and to provide the runtime the understanding of what job to run and to provide details about the job (like a regular JCL). Figure 11-5 shows a trimmed version of an actual xJCL.

```
<?xml version="1.0" encoding="UTF-8" ?>
<job name="MyJob" ... ">
  <substitution-props>
    <prop name="inputDataStream" value="/tmp/input-text.txt" />
    <prop name="outputDataStream" value="/tmp/output-text.txt" /
    <prop name="checkPoint" value="10" />
  </substitution-props>

  <job-step name="MyStep1">
    <classname>com.ibm.ws.batch.MyStep1</classname>
    <batch-data-streams>
      <bds>
       <logical-name>inputStream</logical-name>
         <props>
           <prop name="PATTERN_IMPL_CLASS" value="com.ibm.websph
           <prop name="FILENAME" value="${inputDataStream}" />
         </props>
       </bds>
    </batch-data-streams>
  </job-step>
      :
  <job-step ...>
  </job-step>
</job>
```

If job consists of more steps they are specified in sequence.

Figure 11-5 xJCL Snippet

11.2.5 Integration with Enterprise Schedulers

The point of integration on z/OS is still with Enterprise Scheduler submitting JCL. How does the integration happen between the submitted JCL and the job dispatching function of WebSphere Java Batch? The following Figure 11-6 shows the integration of Enterprise Schedulers and Job Dispatcher function.

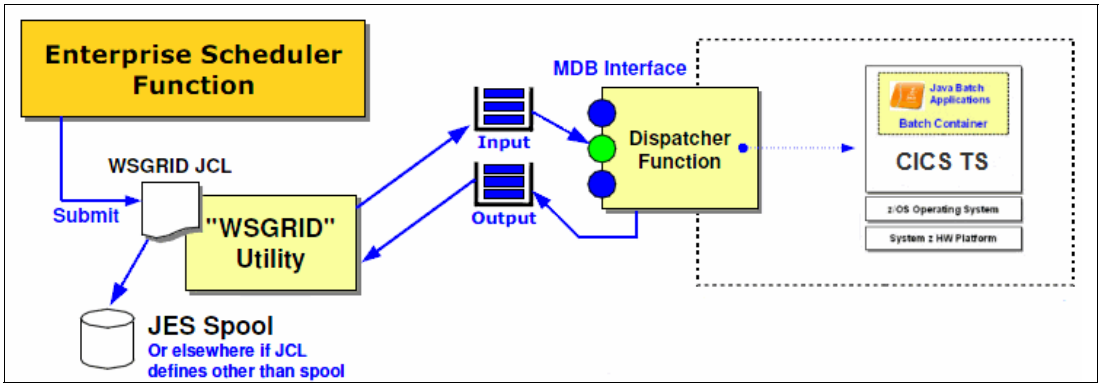


Figure 11-6 Integration with Enterprise Schedulers

The Job Dispatcher function is hosted in a WAS server, and has several interfaces. The one used for this integration is a Message Drive Bean (MDB) interface. The "glue" between Enterprise Scheduler JCL submission and the Dispatcher is a supplied program that uses messaging (IBM MQ or SIBus of WAS) to submit Java batch jobs into WebSphere Java Batch. That "glue" utility is known as "WSGRID." Enterprise Scheduler sees batch job as the WSGRID invocation.

11.2.6 Checkpoint and job restart services

Checkpoint commit and rollback is a function of the batch container. This function is abstracted away from the batch job and is handled by the batch container. It relies on the CICS Syncpoint API to do this. Some key items to note are:

- Checkpoint interval (record or time) specified in the xJCL
- As checkpoint intervals are reached, the container commits the records the checkpoint attained
- In the event of a failure, the job may be restarted at the last good checkpoint.

Figure 11-7 shows the concepts of checkpoint processing.

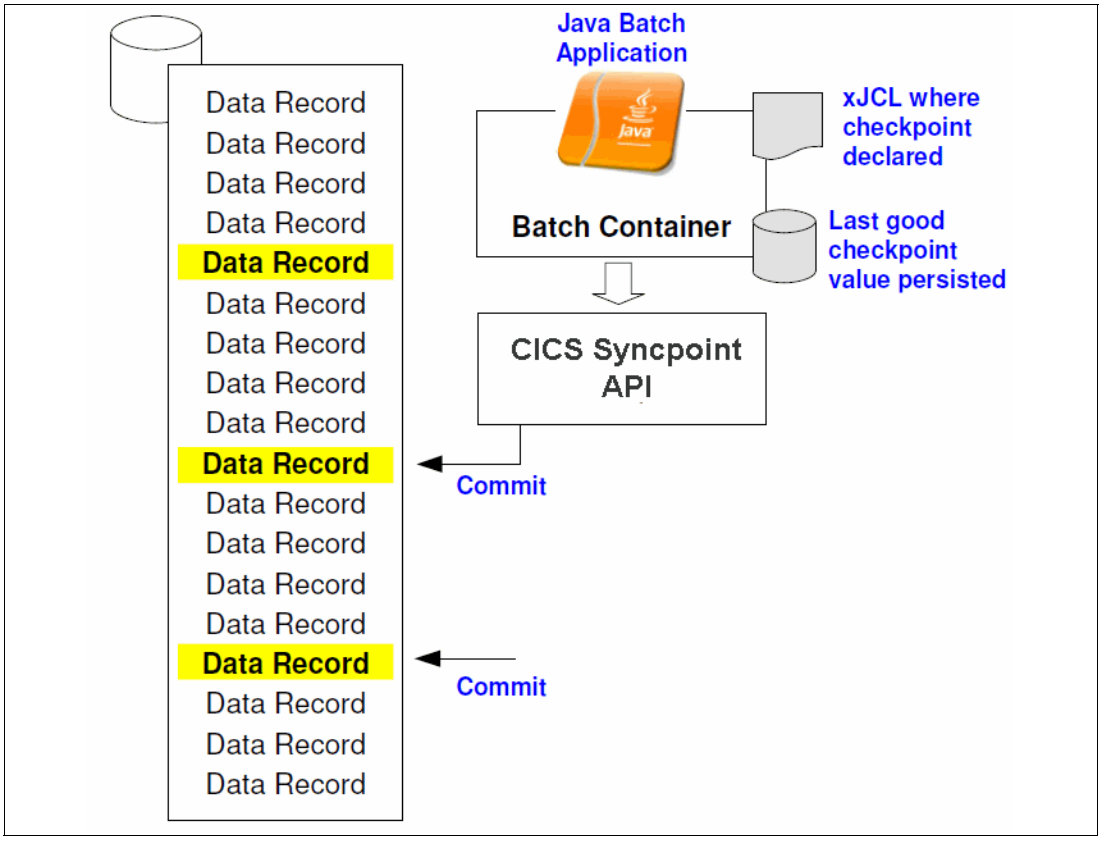


Figure 11-7 Checkpoint Processing

11.2.7 Data record read and write support services

The batch container provides something called “Batch Data Streams” (BDS) as a means of externalizing data access from the job step. This provides a way of abstracting the data read and write logic away from the batch step code. Several Batch Data Streams are provided including:

- Read and write byte data from file
- Read and write text file
- Read from a VSAM KSDS as input data to the job
- update in a VSAM KSDS dataset
- Retreive data from a database using a JDBC connection connection

- Write data to a databse using a JDBC connection

In addition, the CICS Modern Batch Feature Pack provides access to the full set of JCICS APIs.

11.2.8 Job resiliency services

The batch container provides services for the batch job to skip records where data read or write operation throws an exception. It also has the ability to retry job steps when an unhandled exception is thrown.

- Skip Record Processing

  This service provides a way of tolerating a data read or write errors so the job itself may continue. The objective is to provide mechanism to survive the odd data exception rather than stop the whole job. Figure 11-8 shows skip record processing.

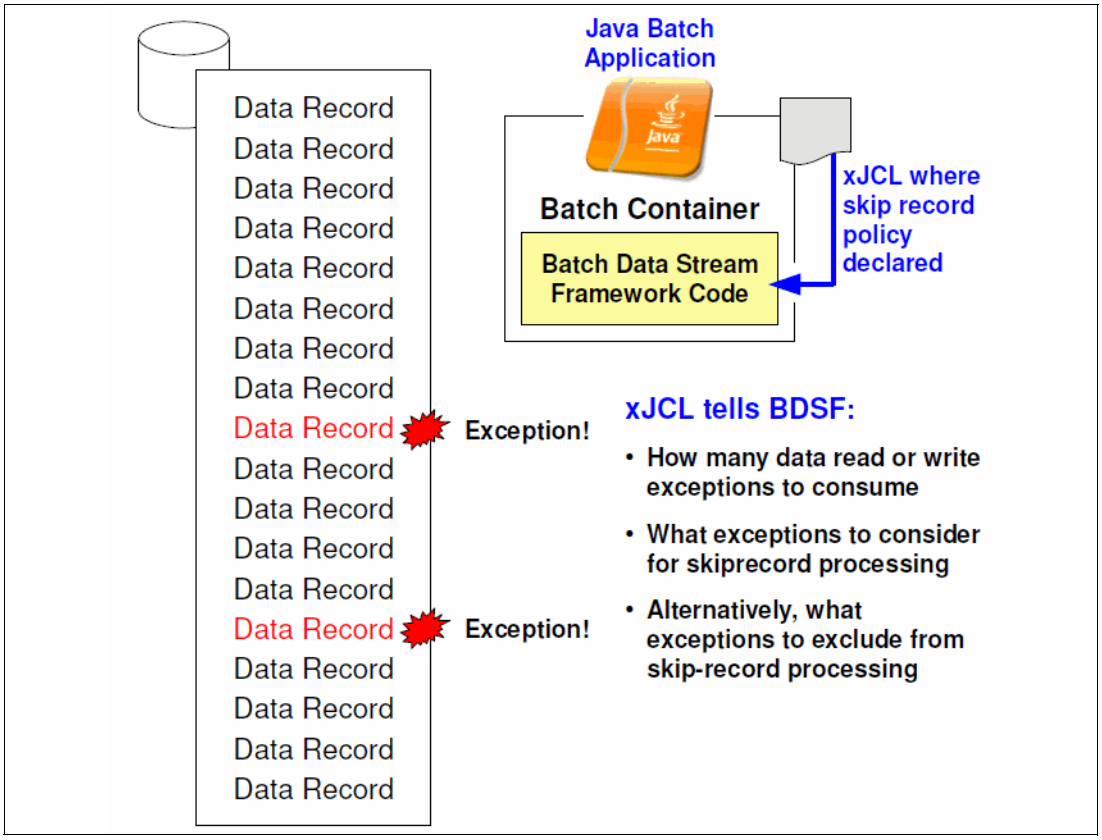


Figure 11-8 Skip Record Processing

- Retry Step Processing

  This service provides a way of retyring the job step in the event of an exception. If successful on retry then the job continues and your processing completes. This is at a higher level from Skip Record. This is a the “invoke batch step” level. This provides a way to retry the step for exceptions. The Batch Container falls back to the last good checkpoint and restarts from there. Figure 11-9 on page 99 depicts the Retry Step Processing. The xJCL tells the container:

  – How many retry steps may be attempted
  – What exceptions to consider for retry-step processing

– Alternatively, what exceptions to exclude from retry-step processing

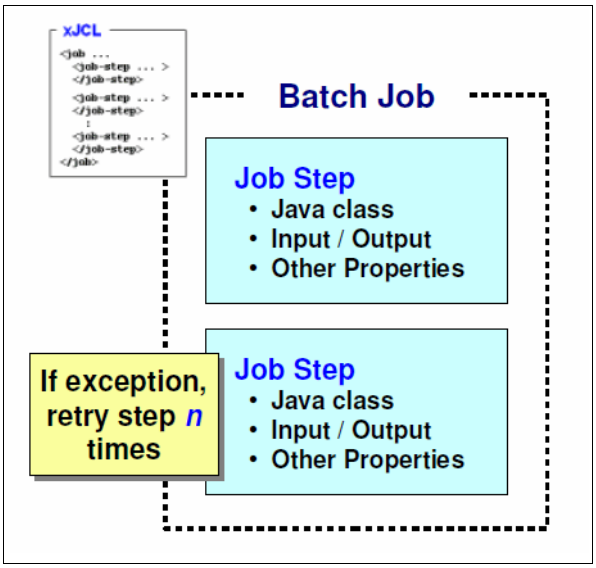


Figure 11-9 Retry-step processing>>>>

11.3 Introduction to CICS batch Support

In this section, we will discuss the CICS support for Modern Batch and how batch fits inside the CICS environment.

11.3.1 CICS Support for Modern Batch

The CICS TS Feature Pack for Modern Batch installs into the CICS region and presents itself to a WebSphere Java Batch dispatcher (Dispatcher) as another endpoint able to dispatch work to. That provides Java batch management in the hands of the WebSphere Java Batch runtime model with CICS being an endpoint.

The Feature Pack for Modern Batch function is configured to communicate over the network to the Dispatcher, telling the Dispatcher about the CICS Feature Pack presence and the Java batch application deployed there. When an xJCL is submitted to the Dispatcher for the Java batch program deployed in the CICS Feature Pack, the Dispatcher communicates across the network to invoke and monitor the progress of the batch program. This allows a CICS region to become an job endpoint for a WebSphere Java Batchdispatching server. This puts batch logic much closer to the CICS data as shown in Figure 11-10 on page 99.

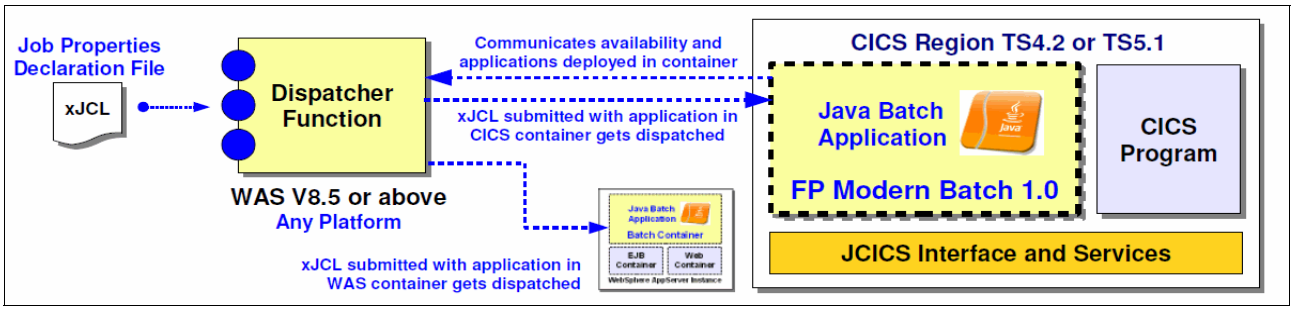


Figure 11-10 CICS TS Feature Pack for Modern Batch

11.4 Running batch applications in CICS

In this section, we review a solution to resolve the conflicting needs of batch and online applications. It can be used to develop a batch application that uses the batch programming model and runs the application in CICS.

These are the key behavioral aspects of such a batch application:

- The batch application shares access to resources with online applications.
- The batch application takes regular checkpoints to free up transactional resources, so that online applications are not blocked from completing for excessive amounts of time.

  Note: The Endpoint provides this checkpointing capability on behalf of the applications.

- If the batch application fails, it can be restarted from its most recent checkpoint.
- Both batch and online applications run concurrently.

The batch application can be divided into job steps that execute in parallel against different subsets of the input data, to shorten the overall elapsed time to process the job.

11.4.1 WebSphere Batch Environment architecture

The CICS TS Feature Pack for Modern Batch provides an Endpoint called the Batch Container that runs in a Java Virtual Machine (JVM) in the CICS address space. The Job scheduler interacts with the Batch Container to start, stop, and manage batch applications. The key component required for the Batch implementation inside CICS TS are:

- WebSphere Application Server 8.5 or above
- CICS Transaction Server 4.2 or above
- CICS TS Feature Pack for Modern Batch

Figure 11-11 depicts this architecture and the interaction between the components.

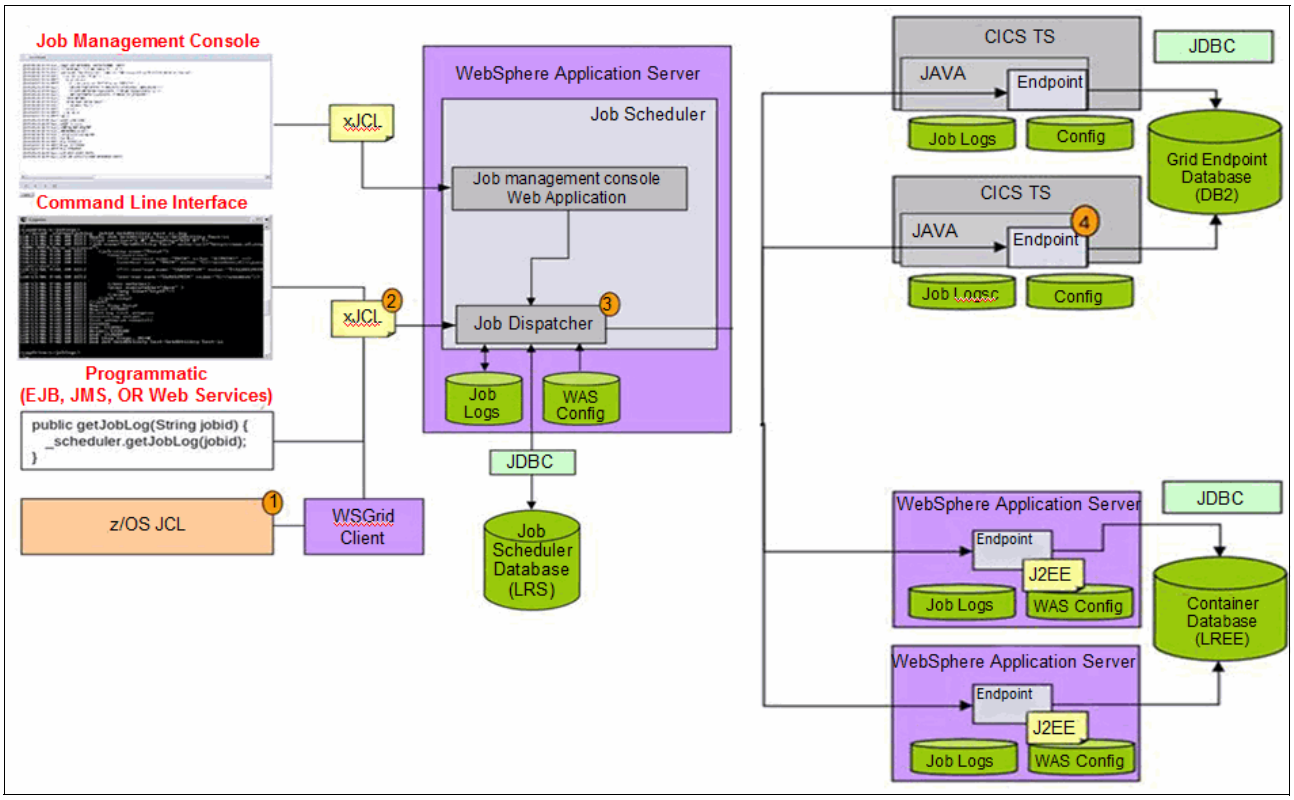


Figure 11-11 CICS provides an Endpoint for the WebSphere Batch Environment

When started in CICS, the Batch Container loads a configuration file that details which batch applications it can run, how to connect to the job scheduler, and how the job scheduler can connect to it. The Batch Container registers with the job scheduler and informs the scheduler of the batch applications it can run. It then periodically sends status information to the scheduler, which states that it is still active and available for work.

The starting and processing of a job are detailed using the numbers in the diagram:

1. JCL is submitted on z/OS to request that the batch application start. The JCL runs program WSGRID and passes it the location of an xJCL document.
2. The WSGRID program connects to the job scheduler and passes the xJCL location. Alternate interfaces are provided to start a batch application, including a console, command line interface, and programmatic API.
3. The job scheduler examines the xJCL to establish the name of the batch application to dispatch and uses the information published to it from all Endpoints to select which Endpoint should run the batch application. The job scheduler chooses a Batch Container hosted in CICS. It then connects to the Batch Container passing it the xJCL.
4. The batch application runs inside the CICS Batch Container.

When the batch application is running in the Batch Container in CICS, it can use Java APIs such as JDBC, or the CICS Java APIs (JCICS), to access CICS resources and services including VSAM files, and to call existing CICS programs written in other languages, such as COBOL, C/C++, PL/I, and Assembler. As the job runs, the Batch Container takes checkpoints, which enables the batch application to be restarted from the last successful checkpoint in the event of a failure. When the batch application completes, the Batch Container notifies the job scheduler.

11.5 Why run a batch application in CICS ?

There are various resons to run batch applications in CICS

- Reduce costs by taking advantage of the CICS VUE pricing model.

  CICS Modern Batch workload is eligible to run in a CICS VUE region and take advantage of the pricing model. This would substantially reduce the processing costs.

- When there is pressure to reduce or eliminate the batch window:

  If CICS is used for online transaction processing and those online applications need to be available for longer each day, then it can make sense to run batch jobs in CICS.

- When CICS has opened resources exclusively that are needed for batch:

  If your batch needs access to resources that are opened exclusively by CICS, then it can make sense to run the batch application under the control of CICS.

- When the batch application does not need exclusive resource access:

  If the batch application runs at the same time as online applications, then updates to the resources being used by the batch job can be made by the online applications. To work in this environment, the batch application needs to be tolerant of these changes.

- When you want to re-use CICS business logic in batch:

  Re-use of existing business logic between online applications and batch helps to reduce duplication and can make it quicker and easier to develop new applications. It is likely that there is significant business logic contained within existing CICS online applications. This logic can be invoked using the JCICS equivalent of the EXEC CICS LINK API.

11.6 What are the benefits of running batch inside CICS ?

The following benefits can result from running batch jobs in CICS:

- It enables online CICS applications to be available closer to 24 hours a day:

  By running online applications and batch applications in parallel, there is less need to take the CICS managed resources offline.

- Capabilities provided by the Batch Container simplify application development:

  Capabilities such as checkpointing, recovery to last checkpoint after a failure, logging, and trace are provided by the Batch Container, removing the need for the application developer to provide these capabilities.

- A common batch programming model is used:

  The Batch Environment provides a common batch programming model across runtimes and platforms. Therefore, any developers skilled in Batch Environment application development should be able to write a batch application to run in CICS.

- Java skills are readily available:

  Java is a well known, popular, modern language. Batch Job Steps and Batch Data Streams are written in Java.

- Modern Batch insice CICS is eligible for VUE:

  CICS Modern Batch workload is eligible to run in a CICS VUE region and take advantage of the pricing model. This would substantially reduce the processing costs.

- Java functionality is available off the shelf:

Functions such as email, PDF generation and XML processing are readily available for Java. This can make it easy to develop batch functions that might be more challenging to implement in languages such as COBOL or PL/I.

11.7 What are the implications of running batch inside CICS ?

Running a batch job under the control of CICS means that the job has different behavioral characteristics to be aware of. Following are some of the implications of running a batch job in CICS.

- Batch jobs may take longer to run.

  Traditional z/OS batch jobs are typically optimized to run as quickly as possible. When a batch job is moved into CICS, it may have to share resources with online applications. These resources can be CPU, files or databases. It is therefore possible that the batch job may take longer to complete. At the same time, with the removal of the batch window, it may be that the batch job can start earlier or complete later. For batch jobs with hard time deadlines, investigation would be required to understand if the job can complete in the time required.

- Implications on online application performance needs to be factored

  It is important that the batch processing does not negatively impact the performance of online applications. A user invoking a CICS transaction might expect a response time of tenths of a second. If the batch application locks too many resources at a time, it could impact the performance of the online applications. The checkpoint interval of the batch job can be adjusted to change the number of updates made within a checkpoint. If the batch job consumes too much CPU, this can also affect the online applications performance.

- Data being updated by batch can be changed by online applications

  Batch applications that rely on a set of records to remain consistent may not work when run in parallel with online applications, because the online applications can update records that the batch job has read, or is about to read. This would need to be considered on a per application basis to determine if it is likely to be an issue.

- Traditional batch job needs factoring

  If you aim to move an existing batch job into CICS, the job needs refactoring to fit into the batch programming model.

11.8 Summary

Batch processing has proved to be an efficient, manageable and reliable method for bulk pro-cessing of updates to data. As businesses expand, the volume of data to be processed also expands. At the same time, the growth of online transactional workloads suggests that a new paradigm is needed to enable the two processing styles to work better together. The CICS batch container provides this new capability.

12

Modern Batch scenario

This chapter describes a CICS modern batch scenario where a Java batch application is developed to reduce the batch window. The Java batch accesses data in VSAM or DB2 without the need to stop the online transactions that access to the same data. The Java batch application may comes from a new requirement, or may be re-written from existing business logic. The architecture is reviewed as well as the implementation of the modern batch Java solution.

12.1 Objective

Traditional batch is a widely used technology to process a bulk of data in a single request with high efficiency. But it usually takes a relatively long time to complete and holds exclusive lock on the files or data tables it reads or updates. This means that the online transactions that rely on the same resources to stop, and thus the batch window impacts real 7*24 hours availability. Many customers are facing very tight batch windows now days. Can we run batch concurrently with on line transactions? Modern batch is the solution for this question.

Most traditional batch on the mainframe is developed in COBOL. Java skills are more common nowadays and is powerful and capable of being used for batch programming as well. What is more, modern batch written in Java is a candidate for approval for IBM CICS Transaction Server for z/OS Value Unit Edition.

To sum up, modern batch in Java can help to reduce the batch window, is easy to develop, and can gain the benefit from IBM CICS Transaction Server for z/OS Value Unit Edition.

The approaches for Java batch include:

- Java batch for new applications

  In this approach, we can leave existing batch processes as they are, but engineer new requirements into the Java batch model. If a batch application has complex algorithms and huge calculations, needs to access to the data controlled by CICS where the data access is either in share mode, or holds the exclusive lock shortly, the application is a good candidate for modern batch.

- Re-engineer existing batch processes

  Re-engineer is not a easy thing to do, but IBM provides some tools to help Java interact with structured data. Refer to IBM Redbooks publication, IBM CICS and the JVM server: Developing and Deploying Java Applications, SG24-8038-00, Chapter 8 which can be downloaded here:

  http://www.redbooks.ibm.com/abstracts/sg248038.html?Open

12.2 Architecture

In this section, we introduce the architecture to use modern batch.

12.2.1 Work flow

In the Figure 12-1 on page 108, we show the architecture of how a Java batch works with CICS. At the front, Enterprise schedulers like IBM Tivoli Workload Scheduler can be used to schedule a batch request from z/OS JCL. The JCL is submitted to WSGRID client which is provided by WebSphere Application Server (WAS) and is used to interact with Java Batch and to integrate the batch system with the enterprise schedules. WSGRID client reads the job properties from XML Job Control Language (xJCL) and then sends the request to Job scheduler in WAS. Job scheduler then schedules the job to endpoints (batch containers) in CICS.

To allow a CICS region to become a job endpoint for a WebSphere Java Batch Dispatching server, CICS TS Feature Pack for Modern Batch V1.0 must be installed. Once a Java batch application is deployed in CICS batch container, CICS communicates with the WAS

Dispatcher Function host and port. That lets the Dispatcher know the endpoint is present and what batch application is deployed.

The batch container provides “Batch Data Streams” (BDS) as a means of externalizing data access from the job step. This provides a way of abstracting the data read and write logic away from the batch step code. Several Batch Data Streams are provided including:

- Read and write byte data from file
- Read and write text file
- Read from a VSAM KSDS as input data to the job
- update in a VSAM KSDS dataset
- Retrieve data from a database using a JDBC connection
- Write data to a database using a JDBC connection

For more information about BDS, see the following website:

http://www-01.ibm.com/support/knowledgecenter/SSGMCP_5.2.0/com.ibm.cics.ts.modernbatch.doc/interfaces_ref.html?lang=en

In addition, the CICS Modern Batch Feature Pack provides access to the full set of JCICS APIs. For more information about JCICS services and examples, see the following website:

http://www-01.ibm.com/support/knowledgecenter/SSGMCP_5.2.0/com.ibm.cics.ts.java.doc/topics/dfhpjla.html?lang=en

To access to MQ from CICS modern batch, you can refer to the article below for detailed steps:

https://www.ibm.com/developerworks/community/blogs/cicsdev/entry/using_websphere_mq_java_bindings_with_a_liberty_jvm_server?lang=en

Here, put more focus on how to access to VSAM files because VSAM is the most common data source for batch. CICS JCICS services support to read and update a VSAM file, but do not support to open or close a file. So if the file is not owned by the Java batch region, but owned by another CICS region for on line transactions. You have two solutions:

1. Function Ship file operations to File Owned Region (FOR)

   This solution is shown in Figure 12-1 on page 108. Like what we usually do for the concurrent access by online transactions, we can route all the file work by Function Shipping to a FOR running in a normal z/OS LPAR to complete the file read and update.

   In the solution, Java workload is in zNALC enabled LPAR, and the workload for file operations are shipped to a normal z/OS LPAR. We do not need to care about file open and close, and the best thing is that modern batch can run concurrently with online transactions.

2. Use Record Level Sharing (RLS) for VSAM file access

   RLS is an another solution which enables modern batch to work concurrently with online transactions and thus reduces the batch window. RLS is a VSAM data set access mode, introduced in DFSMS, and supported by CICS. RLS enables VSAM data to be shared, with full update capability, between many applications running in many CICS regions. With RLS, CICS regions that share VSAM data sets can reside in one or more MVS images within an MVS sysplex.

FOR and RLS are preferred solutions for Java modern batch. You can select a solution based on the existing VSAM access method by your online transactions.

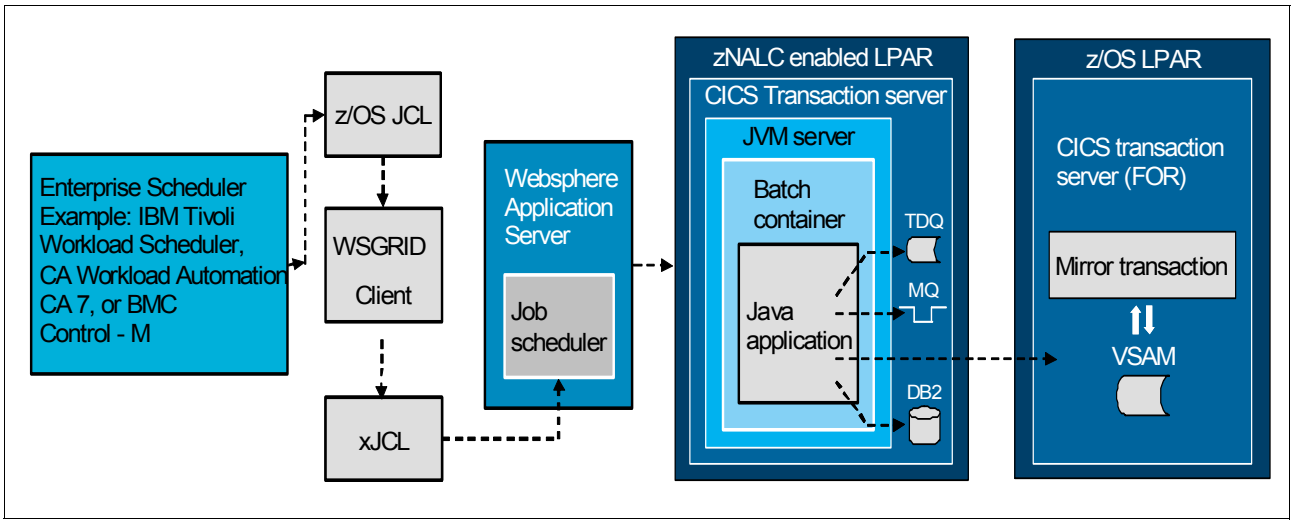


Figure 12-1 Architecture of a pure Java batch solution

12.2.2 High Availability consideration

To achieve the High Availability, we can set up two or even more batch containers for a same batch application. The batch containers can be deployed in the same CICS region, or in separate CICS regions in the same LPAR, or even in CICS regions in separate LPARs. All the batch containers send the heartbeat information to the job scheduler at regular intervals. So from the view of the job scheduler, all these batch contains are candidate to run the same batch applications. Once one container fails, the job scheduler can not hear the heartbeat from that container, so all the requests then go to the alive ones.

Even if we select to have only one batch container, CICS modern batch can achieve better availability than the traditional batch by using Checkpoint. In the traditional batch, we usually store a backup copy of the master file before running the batch, restore the file back after a failure and then rerun the whole batch again. In the modern batch, the batch container takes checkpoints, which enables the batch application to be restarted from the last successful checkpoint in the event of a failure. This reduces the time to recover from a failure, and thus improves the availability.

12.2.3 Security consideration

CICS Modern Batch Feature Pack supports to secure the communication between the batch container and the job scheduler with SSL client authentication. You can turn on the security setting in the batchcontainer-config.xml file. For more information about the configuration, see the following website:

http://www-01.ibm.com/support/knowledgecenter/SSGMCP_5.2.0/com.ibm.cics.ts.modernbatch.doc/tasks/config_fp.html?lang=en

After the SSL handshake, the job scheduler can schedule the job to the batch containers in CICS. The batch containers use TCPIPService to listen to the requests from the job scheduler. Then URIMAP is used to route the request to the batch applications. In the URIMPAP definition, we can define the transaction ID and the user ID to run the batch applications. The default transaction is CBCR. Make sure the transaction is running under an ID that have enough authority to access the resource. As online transactions, the security of the resource is controlled by External Security Manager, in most case RACF.

12.3 Implementation

In this section, we describe basic steps to develop a Java batch application, deploy it in CICS and schedule it. IBM Redbooks publication New Ways of Running Batch Applications on z/OS Volume 1 CICS Transaction Server, SG24-7991-00 explains more detailed steps of how to use RAD to develop a Java batch application and to schedule it. You can download the Redbook here:

http://www.redbooks.ibm.com/abstracts/sg247991.html?Open

To use the CICS® Modern Batch Feature Pack you must have the correct versions of software and service installed.

Your environment must comply with the following prerequisites:

- CICS Transaction Server for z/OS, Version 4.2 or higher.
- CICS Transaction Server for z/OS®, Version 4.2 requires APAR PM82511
- CICS Transaction Server for z/OS, Version 5.1 requires APAR PM82519
- z/OS® Version 1 Release 12 or higher.
- WebSphere® Application Server Version 8.5 for z/OS or higher, or WebSphere Application Server Network Deployment 8.5 or higher.
- DB2® Version 9.1 or higher.

12.3.1 Install and configure CICS TS Feature Pack for Modern Batch

CICS TS Feature Pack for Modern Batch is installed by using the SMP/E RECEIVE, APPLY, and ACCEPT commands. The program number for CICS Modern Batch Feature Pack is 5655-Y50, and the FMID is HCIF51B. You can use the sample jobs that are provided to perform part or all of the installation tasks, or you can use the SMP/E dialogs to perform the SMP/E installation steps.

For details about the installation procedure, see the Program Directory for Modern Batch Feature Pack:

http://www-05.ibm.com/e-business/linkweb/publications/servlet/pbi.wss?CTY=US&FNC=SRX&PBL=GI13-3324

To configure Modern Batch Feature Pack, detailed steps are described here:

http://www-01.ibm.com/support/knowledgecenter/SSGMCP_5.2.0/com.ibm.cics.ts.modernbatch.doc/tasks/configuring_modernbatch.html?lang=en

12.3.2 Developing a batch application

You can develop applications with Rational Application Developer (RAD) or the CICS Explorer SDK. RAD is a preferred IDE to develop the batch application. But if you do not have RAD, you can use Eclipse platform with CICS Explore SDK for free.

The two most important things you need to develop is XML Job Control Language (xJCL) and Java class to implement the Job step logic.

If you have RAD, you can set up a Batch Project to easily generate xJCL as well as Java class template to implement your batch job steps by friendly GUI panels.

Figure 12-2 on page 110 shows an example of the generated xJCL from RAD. xJCL describes how to run a batch job and defines the input and output streams to that batch job. In

the xJCL, you can define batch job steps, checkpoint algorithms, results algorithms, Batch Data Streams (BDS) and batch job return codes. You can refer to IBM Knowledgecenter:

http://www-01.ibm.com/support/knowledgecenter/SSGMCP_5.2.0/com.ibm.cics.ts.modernbatch.doc/concepts/devapps_batchprogmodel.html?lang=en

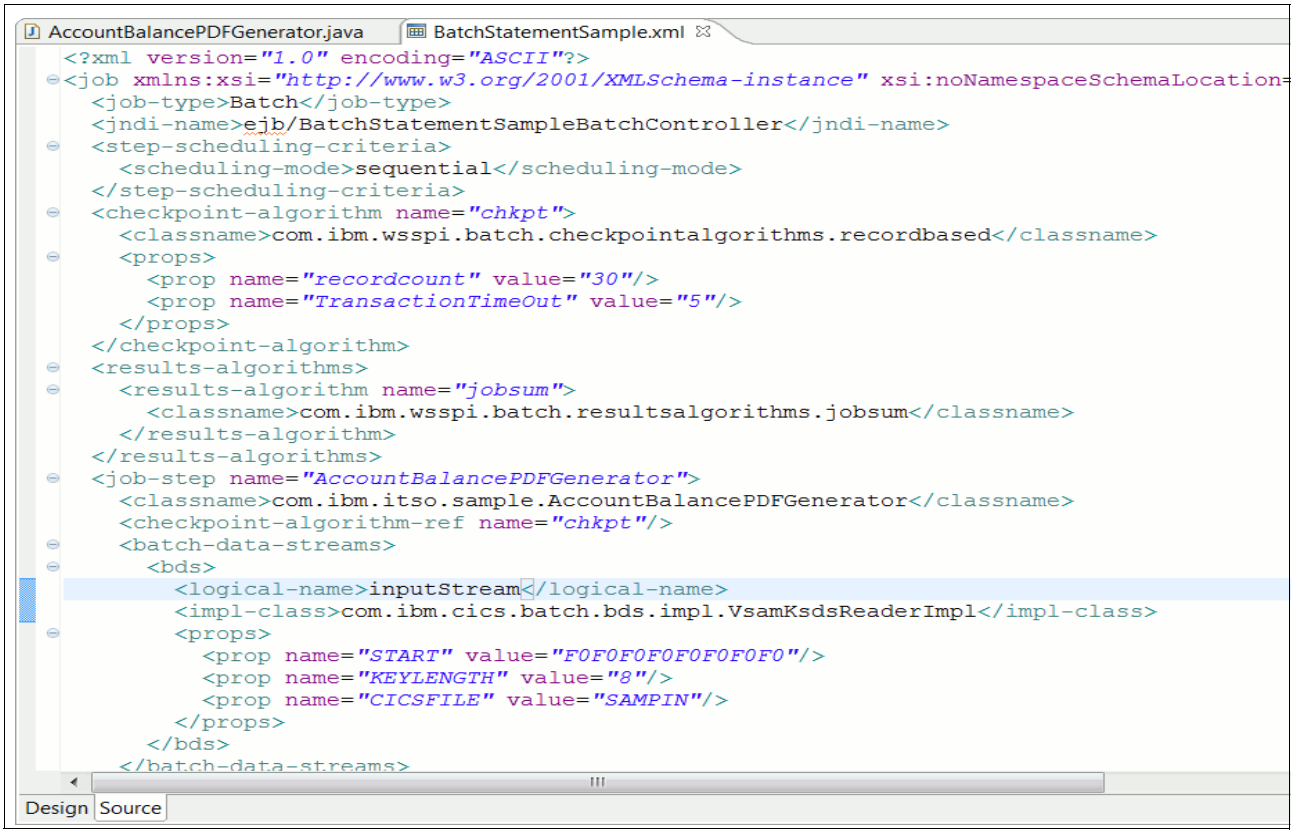


Figure 12-2 Extract of the generated xJCL

Figure 12-3 on page 111 shows an extract of the generated Java class for the Batch Job Step. This class is where the code for the job step will be added.

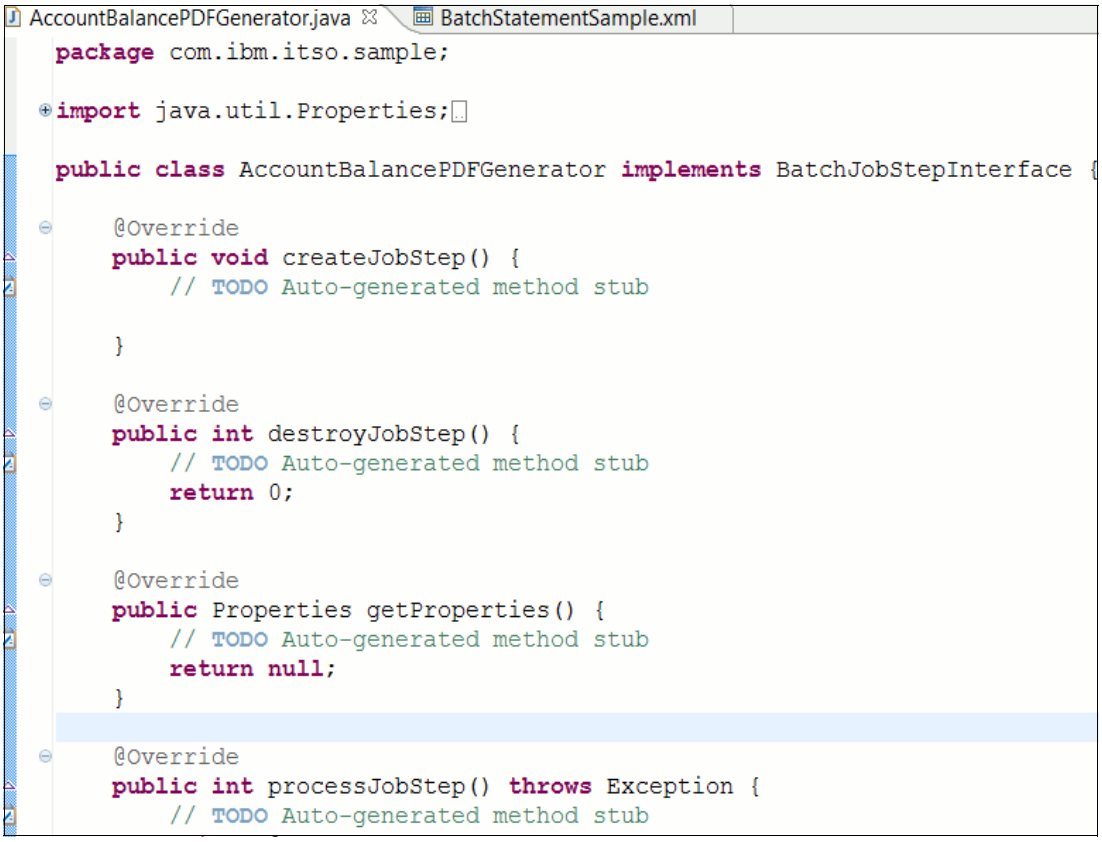


Figure 12-3 Extract of the generated Batch Job Step implementation class

After you get the generated Batch Job Step implementation class, you can add your specific business logic there. Batch steps are implemented as Plain Old Java Object (POJO) classes that implement the interface, com.ibm.websphere.batch.BatchJobStepInterfance. Batch job steps are performed sequentially. Callback methods in the BatchJobStepInterface allow the grid endpoints to run batch steps when it runs a batch job. For more information, you can see the following website:

http://www-01.ibm.com/support/knowledgecenter/SSAW57_8.5.5/com.ibm.websphere.nd.iseries.doc/ae/cgrid_xdbatchstp.html?lang=en

If you do not have RAD, then you need to develop the xJCL and the job step Java classes by yourself. You need to create a Java project first, to add additional Jars in the build path, then to write a Java class for each job step and to write a xJCL to describe how to run the job. Detailed steps can be found in IBM Knowledge center:

http://www-01.ibm.com/support/knowledgecenter/SSGMCP_5.2.0/com.ibm.cics.ts.modernbatch.doc/tasks/devapps_using_sdk.html?lang=en

12.3.3 Deploying the batch application in CICS

The first thing to do is to export the application as a Jar file containing the compiled Java classes. Then follow these steps to install the application into CICS:

1. Copy the JAR file from your workstation onto the mainframe. We used FTP to transfer the file in binary mode. Store the file in a directory that CICS has permissions to access. Ensure that CICS has permission to read the JAR file too.

2. Modify the JVM profile for the JVM server that the Batch Container is running in. Add the Jar to the CLASSPATH_SUFFIX. If you use external interfaces from other Jar files in your batch project, then these Jar files should also be added to the CLASSPATH_SUFFIX
3. The CICS JVMSERVER resource will need to be disabled and re-enabled to pick up the changes to the classpath.
4. Modify a configuration file named `batchcontainer-config.xml to notify the scheduler that CICS can run the batch job.`
5. After re-enabling the JVMSERVER, run the CBCH transaction if it is not already running.

12.3.4 Submit the xJCL to run the batch job

Before you submit the xJCL, please make sure the Default Application Name is the same as the application name you specifies in the *batchcontainer-config.xml* file.

Then you can submit this xJCL from any supported Enterprise Scheduler. For example, if you use Job Management Console, logon the web interface, specify the path to the xJCL on your local workstation, then submit the xJCL to the job scheduler. Figure 12-4 shows how to submit a job from the Job Management Console.

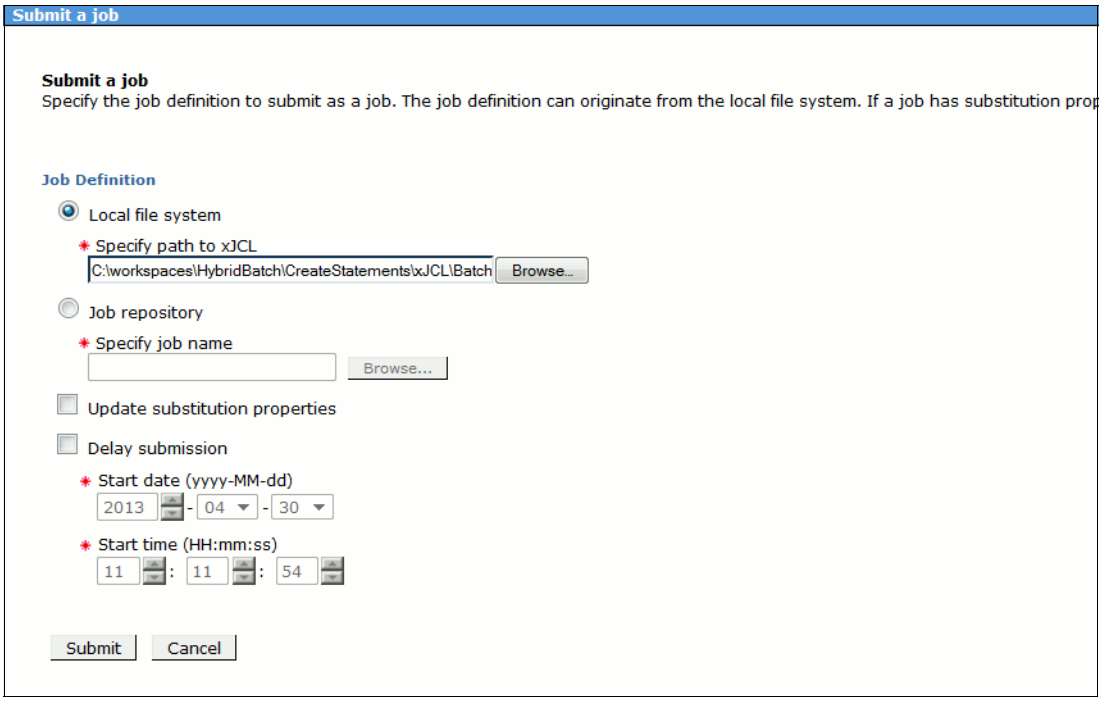


Figure 12-4 Submitting a job from the Job Management Console

A message is shown, which confirms that the job has been submitted successfully and provides the job ID. When a job is successfully submitted, it means the job scheduler has identified at least one Batch Container which claims to be able to run the job. The scheduler will choose a Batch Container and dispatch the job to it. We can examine the log for the job to see which Batch Container the scheduler has chosen to dispatch the job to. Figure 12-5 on page 113 shows a job log from the Job Management Console.

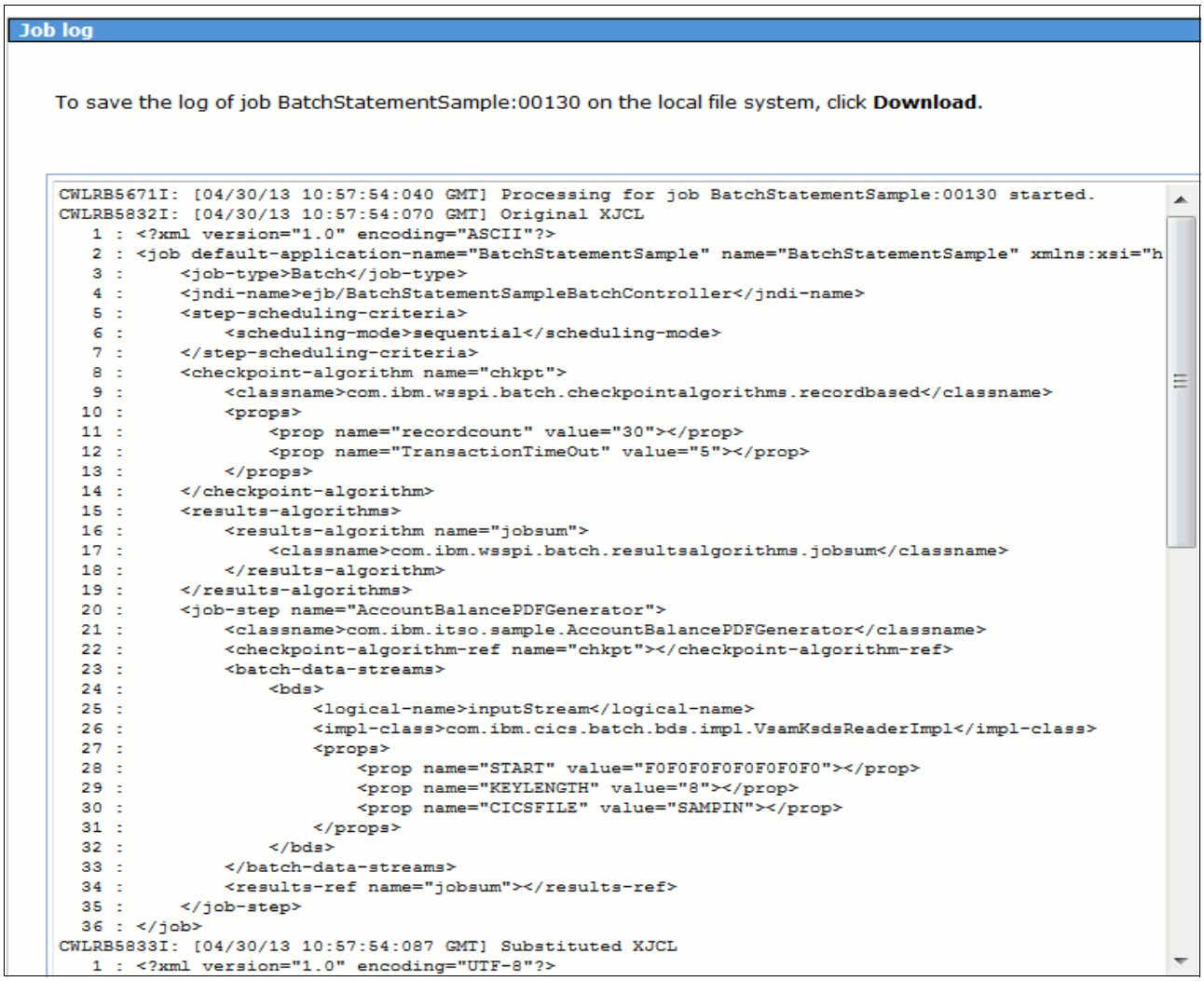


Figure 12-5 The Job log view in the Job Management Console

The steps above shows how to develop a batch application, to deploy it and to schedule it. More detailed steps can be found in the Redbook mentioned at the beginning of this section.

Related publications

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

IBM Redbooks

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

- *New Ways of Running Batch Applications on z/OS: Volume 1 CICS Transaction Server,* SG24-7991-00
- *WebSphere Application Server V8.5 Administration and Configuration Guide for Liberty Profile,* SG24-8170-00
- *WebSphere Application Server Liberty Profile Guide for Developers*, SG24-8076-01
- *Flexible Decision Automation for Your zEnterprise with Business Rules and Events,* SG24-8014-01
- *Implementing IBM CICS JSON Web Services for Mobile Applications,* SG24-8161-00
- *Securing Your Mobile Business with IBM Worklight*, SG24-8179-00
- *Configuring and Deploying Open Source with WebSphere Application Server Liberty Profile*, SG24-8194-00

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

ibm.com/redbooks

Online resources

These websites are also relevant as further information sources:

- Description1

  http://????????.???.???/
- Description2

  http://????????.???.???/
- Description3

  http://????????.???.???/

Help from IBM

IBM Support and downloads

ibm.com/support

IBM Global Services

ibm.com/services

To determine the spine width of a book, you divide the paper PPI into the number of pages in the book. An example is a 250 page book using Plainfield opaque 50# smooth which has a PPI of 526. Divided 250 by 526 which equals a spine width of .4752". In this case, you would use the .5” spine. Now select the Spine width for the book and hide the others: **Special>Conditional Text>Show/Hide>SpineSize(-->Hide:)>Set** . Move the changed Conditional text settings to all files in your book by opening the book file with the spine.fm still open and **File>Import>Formats** the Conditional Text Settings (ONLY!) to the book files.

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

To determine the spine width of a book, you divide the paper PPI into the number of pages in the book. An example is a 250 page book using Plainfield opaque 50# smooth which has a PPI of 526. Divided 250 by 526 which equals a spine width of .4752". In this case, you would use the .5” spine. Now select the Spine width for the book and hide the others: **Special>Conditional Text>Show/Hide>SpineSize(-->Hide:)>Set** . Move the changed Conditional text settings to all files in your book by opening the book file with the spine.fm still open and **File>Import>Formats** the Conditional Text Settings (ONLY!) to the book files.

An Architect's Guide to New Java Workloads in CICS (Transaction Server)

Value Unit Edition family of products allows a choice on how to budget for new projects

IBM CICS Transaction Server for z/OS Value Unit Edition running on a zNALC LPAR is a one time charge

Use Value Unit Edition products to provide an environment to develop and deploy net new Java based workload

This IBM® Redpaper Redbooks® publication introduces the System z® New Application License Charge (zNALC) pricing structure and examples of zNALC workload scenarios which is the focus of this book. This book describes the products that can be run on a zNALC LPAR and reasons why one would consider such an implementation.

This book provides an introduction to the WebSphere Application Server Liberty Profile, its ability to host applications within a CICS environment, and how it will interact with CICS applications and resources. In addition, it describes the security technologies that are available to applications that are hosted within a WebSphere Application Server Liberty Profile in CICS.

This book describes how to implement the modernization of presentation in CICS with CICS Liberty and share scenarios to develop CICS Liberty applications to gain the benefit from IBM CICS Transaction Server for z/OS Value Unit Edition.

This book discusses some general considerations when using mobile devices to interact with CICS applications and explains specific CICS technologies for connecting mobile devices using IBM CICS Transaction Server for z/OS Value Unit Edition.

This book introduces the concept of decision management and describes how IBM Operational Decision Manager for z/OS can run inside the CICS Transaction Server for z/OS to provide decision management for CICS COBOL and PL/I applications.

This book discusses the CICS Transaction Server for z/OS (CICS TS) Feature Pack for Modern Batch. This capability can be installed into CICS TS V4.2 or higher and enables the WebSphere Batch Environment to schedule and manage batch applications in CICS.

INTERNATIONAL TECHNICAL SUPPORT ORGANIZATION

BUILDING TECHNICAL INFORMATION BASED ON PRACTICAL EXPERIENCE

IBM Redbooks are developed by the IBM International Technical Support Organization. Experts from IBM, Customers and Partners from around the world create timely technical information based on realistic scenarios. Specific recommendations are provided to help you implement IT solutions more effectively in your environment.

For more information:
ibm.com/redbooks

SG24-8225-00 ISBN