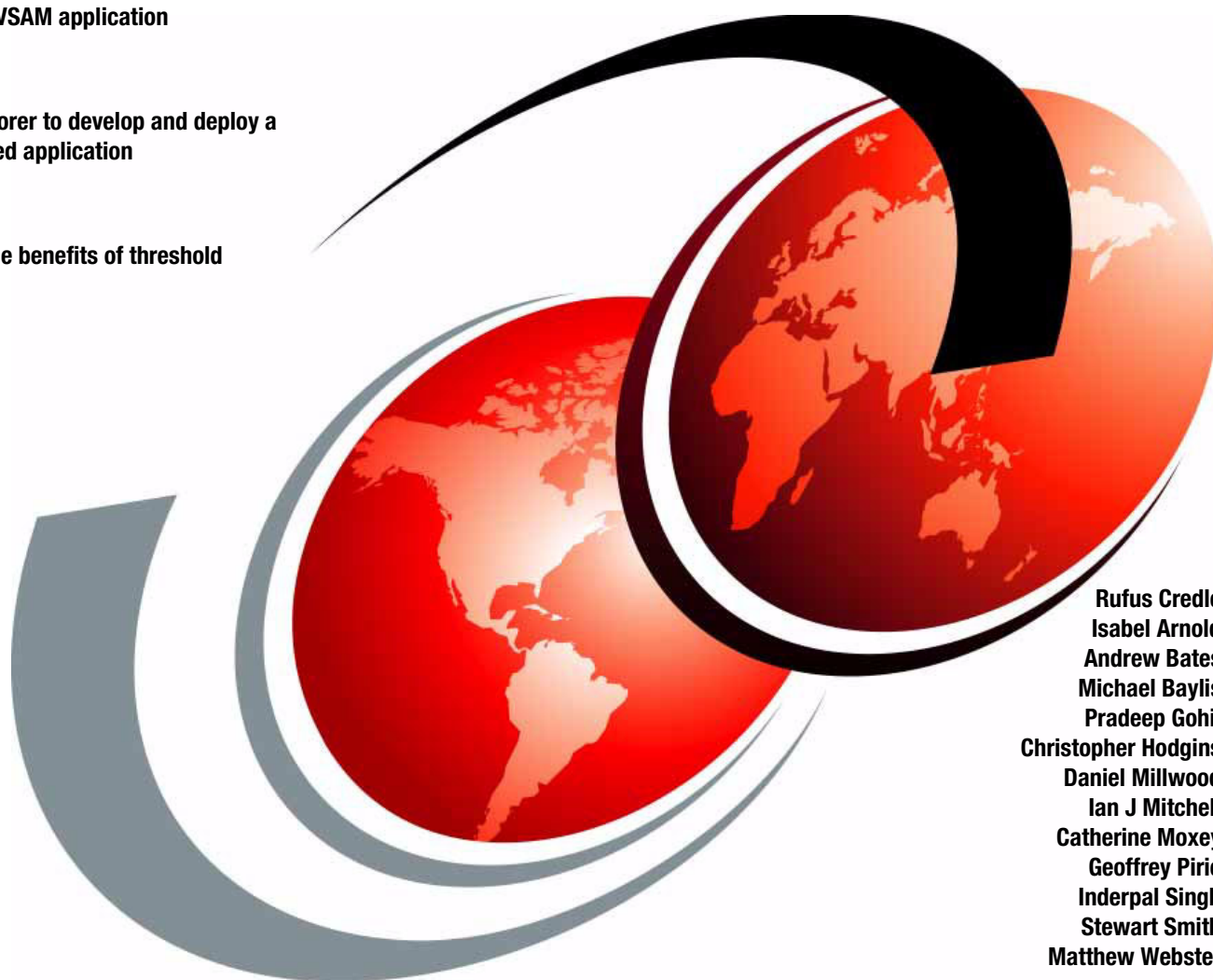


Cloud Enabling IBM CICS

Discover how to quickly cloud enable a traditional IBM 3270-COBOL-VSAM application

Use CICS Explorer to develop and deploy a multi-versioned application

Understand the benefits of threshold policy



Rufus Credle
Isabel Arnold
Andrew Bates
Michael Baylis
Pradeep Gohil
Christopher Hodgins
Daniel Millwood
Ian J Mitchell
Catherine Moxey
Geoffrey Pirie
Inderpal Singh
Stewart Smith
Matthew Webster

Redbooks



International Technical Support Organization

Cloud Enabling IBM CICS

December 2014

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

First Edition (December 2014)

This edition applies to CICS Transaction Server for z/OS version 5.1, 3270-COBOL-VSAM application.

© Copyright International Business Machines Corporation 2014. All rights reserved.

Note to U.S. Government Users Restricted Rights -- Use, duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp.

Contents

Notices	vii
Trademarks	viii
IBM Redbooks promotions	ix
Preface	xi
Authors	xii
Now you can become a published author, too!	xv
Comments welcome	xv
Stay connected to IBM Redbooks	xvi
Part 1. Introduction	1
Chapter 1. Cloud enabling your CICS TS applications	3
1.1 Did you know?	4
1.2 Business value	4
1.3 Solution overview	5
1.4 Cloud computing in a CICS TS context	6
1.5 Overview of the cloud-enabling technologies in CICS TS V5	11
1.5.1 Platform overview	12
1.5.2 Application overview	13
Chapter 2. GENAPP introduction	15
2.1 CICS TS topology	16
2.2 Application architecture	17
2.2.1 GENAPP in a single managed region	17
2.2.2 GENAPP in a CICS TS topology	19
2.3 Limitations of the current design	21
2.3.1 Meaningful names	21
2.3.2 Taking the application through the lifecycle	22
2.3.3 Managing updates to GENAPP	22
2.3.4 Dependencies	22
2.3.5 Multiple region separation	22
2.3.6 Control of resource usage	22
Part 2. How to cloud-enable your IBM CICS TS	23
Chapter 3. Creating a platform	25
3.1 The CICS TS platform	26
3.2 The role of a CICS TS platform	26
3.3 The components of a CICS TS platform	26
3.3.1 Status	27
3.3.2 Lifecycle	28
3.3.3 Scope	29
3.3.4 Region types	29
3.3.5 Services	29
3.3.6 Policies	30

3.4 Example single-region platform project	30
3.4.1 Defining the development topology	31
3.4.2 Defining the platform	36
3.4.3 Deploy the platform	44
3.4.4 Multi-region platform for test or production	49
3.5 Outcome	49
Chapter 4. Creating an application	51
4.1 The CICS TS application	52
4.1.1 Gaining value for existing CICS TS applications	53
4.2 The components of a CICS TS application	53
4.3 Developing a CICS TS application	55
4.4 Example	56
4.4.1 Prerequisites	56
4.4.2 Creating application entry points	57
4.4.3 CICS Application Project	62
4.4.4 CICS Application Binding Project	66
4.4.5 Deploying the CICS TS application and creating the application definition	69
4.4.6 Installing and managing the CICS TS application	73
4.4.7 Adding dependencies into the CICS TS application	76
4.4.8 Updating the CICS Application Project	84
4.4.9 Update the CICS Application Binding Project	86
4.4.10 Redeploying the CICS TS application	87
4.4.11 Installing and managing the updated CICS TS application	90
4.4.12 Diagnosing dependency errors	93
4.5 The outcome	95
Chapter 5. Applying a policy	97
5.1 CICS Policy	98
5.1.1 Policy rule thresholds	98
5.1.2 Policy rule actions	99
5.1.3 Policy rule scope	99
5.2 Example CICS policies	100
5.2.1 Platform policy example	100
5.2.2 Application operation policy example	109
5.2.3 Installing the updated application	122
5.2.4 View active policy	125
5.3 Results of adding a policy	126
Chapter 6. Packaging an application for multiversion deployment	127
6.1 Prerequisites	128
6.2 Example 1: Repackaging CICS TS resources into the GeneralInsuranceCustomer CICS TS application	128
6.2.1 Discarding the existing application and resources	129
6.2.2 Allocating a new data set, preparing load modules, and creating a dynamic load library definition	133
6.2.3 Updating CICS TS resources	137
6.2.4 Updating the application version	144
6.2.5 Exporting, installing, and enabling the application	148
6.2.6 Making the application available and running it	153

6.3 Example 2: Provisioning a logic change to the GeneralInsuranceCustomer CICS TS application	154
6.3.1 Fixing the bug and compiling	154
6.3.2 Updating data set name in LIBRARY	157
6.3.3 Updating the application version	160
6.3.4 Exporting, installing, and enabling	163
6.3.5 Validating that application version 1.0.2 is still operational	168
6.3.6 Making application version 1.100.0 available and running it	169
6.4 Possible extensions for the application examples	171
6.5 Summary	172
Part 3. What you get by cloud-enabling your CICS TS	173
Chapter 7. Measurement by application	175
7.1 Overview	176
7.1.1 The application context within the CICS Monitoring Facility	176
7.1.2 Problems with traditional chargeback and performance monitoring	177
7.1.3 Pre-requisites to using application context in CICS Monitoring Facility records	178
7.2 Examples of using the application context in the CICS Monitoring Facility	178
7.2.1 Enabling chargeback within the GENAPP application using CICS Performance Analyzer	178
7.2.2 Monitoring performance of the GENAPP application using CICS Performance Analyzer	179
7.2.3 Diagnosing problems in GENAPP using CICS Explorer	180
Chapter 8. Managing by policy	181
8.1 Benefits of CICS Policy	182
8.2 Reviewing CICS Threshold Policy	182
8.2.1 Policy rule types	182
8.2.2 Policy actions	184
8.3 Policy scope	186
8.3.1 Policy for applications	190
8.3.2 Policy for platforms	190
8.4 Setting policy	190
8.5 Policy scenarios	191
8.6 Concluding thoughts	192
Chapter 9. Application lifecycle management	193
9.1 Managing application change through versioning	194
9.1.1 Semantic versioning	194
9.1.2 Development lifecycle	195
9.2 Managing application deployment through versioning	195
9.2.1 Switching application versions with no outage	196
9.2.2 Phasing in a new application	197
9.2.3 CICS TS region consolidation	199
9.3 Applying versioning to GeneralInsuranceCustomer	200
9.3.1 Using semantic versioning	200
9.3.2 Semantic versioning in the development lifecycle	201
9.3.3 Semantic versioning in the deployment lifecycle	201
9.4 Evolving the GeneralInsuranceCustomer application	202
9.5 Promoting GeneralInsuranceCustomer from dev to test	203
9.6 GeneralInsuranceCustomer deployment lifecycle	203
9.7 Outcomes	204

Part 4. Appendixes	205
Appendix A. Setup and environment.	207
Installing the general insurance application	208
Before you begin	208
Procedure to customize	209
Results	212
Building the application environment	212
About this task	212
Procedure	212
Build the CICS TS environment	213
Single managed region	213
CICS TS topology	214
Add GENAPP load library to CICS TS as a dynamic LIBRARY	215
Testing the general insurance application	217
Appendix B. IBM CICS Explorer setup	219
Overview	220
Download and Install CICS TS	220
Install the CICS TS resources required for CICS Explorer	220
Define credentials to CICS Explorer	220
Configure a CMCI connection to CICS TS from CICS Explorer	221
Configure an FTP connection to CICS TS from CICS Explorer	221
Change to the CICS Cloud perspective	221
Related publications	223
IBM Redbooks	223
Other publications	223
Online resources	223
Help from IBM	223

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

IBM, the IBM logo, and ibm.com are trademarks or registered trademarks of International Business Machines Corporation in the United States, other countries, or both. These and other IBM trademarked terms are marked on their first occurrence in this information with the appropriate symbol (® or ™), indicating US registered or common law trademarks owned by IBM at the time this information was published. Such trademarks may also be registered or common law trademarks in other countries. A current list of IBM trademarks is available on the Web at <http://www.ibm.com/legal/copytrade.shtml>

The following terms are trademarks of the International Business Machines Corporation in the United States, other countries, or both:

CICS®	MVS™	Resource Measurement Facility™
CICS Explorer®	Parallel Sysplex®	RMF™
CICSplex®	RACF®	System z®
DB2®	Rational®	VTAM®
IA®	Redbooks®	WebSphere®
IBM®	Redpapers™	z/OS®
Language Environment®	Redbooks (logo)  ®	z/VM®

The following terms are trademarks of other companies:

Windows, and the Windows logo are trademarks of Microsoft Corporation in the United States, other countries, or both.

Java, and all Java-based trademarks and logos are trademarks or registered trademarks of Oracle and/or its affiliates.

UNIX is a registered trademark of The Open Group in the United States and other countries.

Other company, product, or service names may be trademarks or service marks of others.

Find and read thousands of IBM Redbooks publications

- ▶ Search, bookmark, save and organize favorites
- ▶ Get up-to-the-minute Redbooks news and announcements
- ▶ Link to the latest Redbooks blogs and videos

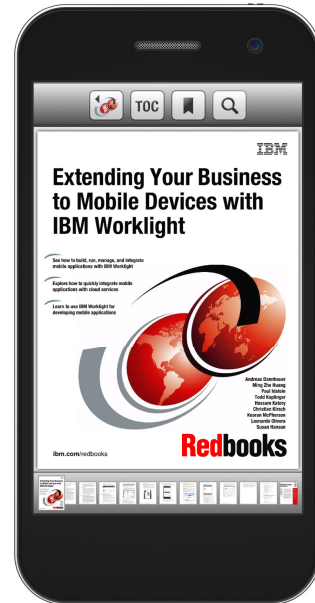
Get the latest version of the Redbooks Mobile App



iOS

Download
Now

Android



Promote your business in an IBM Redbooks publication

Place a Sponsorship Promotion in an IBM® Redbooks® publication, featuring your business or solution with a link to your web site.

Qualified IBM Business Partners may place a full page promotion in the most popular Redbooks publications. Imagine the power of being seen by users who download millions of Redbooks publications each year!



ibm.com/Redbooks

About Redbooks → Business Partner Programs

THIS PAGE INTENTIONALLY LEFT BLANK

Preface

This IBM® Redbooks® publication takes an existing IBM 3270-COBOL-VSAM application and describes how to use the features of IBM Customer Information Control System (CICS®) Transaction Server (CICS TS) cloud enablement. Working with the General Insurance Application (GENAPP) as an example, this book describes the steps needed to monitor both platform and application health using the CICS Explorer CICS Cloud perspective.

It also shows you how to apply threshold policy and measure resource usage, all without source code changes to the original application. In addition, this book describes how to use multi-versioning to safely and reliably apply and back out application changes.

This Redbooks publication includes instructions about the following topics:

- ▶ How to create a CICS TS platform to manage and reflect the health of a set of CICS TS regions, and the services that they provide to applications
- ▶ How to quickly get value from CICS TS applications, by creating and deploying a CICS TS application for an existing user application
- ▶ How to protect your CICS TS platform from erroneous applications by using threshold policies
- ▶ How to deploy and run multiple versions of the same CICS TS application on the same CICS TS platform at the same time, enabling a safer migration from one application version to another, with no downtime
- ▶ How to measure application resource usage, enabling a comparison of the performance of different application versions, and chargeback based on application use

This book describes how CICS TS cloud enablement uses existing operational facilities, including monitoring, events, transaction tracking, CICS TS bundles, and IBM CICSplex® System Manager (CICSplex SM), to integrate with existing deployment and management processes.

Authors

This book was produced by a team of specialists from around the world working at the IBM Hursley Park, Hursley UK.



Rufus Credle is a Certified Consulting information technology (IT) Specialist at the ITSO, Raleigh Center. In his role as project leader and information developer, he conducts residencies and develops IBM Redbooks, IBM Redpapers™, Solution Guides, and so on. Subjects include network operating systems, enterprise resource planning (ERP) solutions, voice technology, high availability (HA), clustering solutions, web application servers, pervasive computing, IBM and original equipment manufacturer (OEM) applications, IBM WebSphere® Commerce, IBM MQ, CICS TS, IBM industry technology, IBM 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 B.S. degree in Business Management from Saint Augustine's College. Rufus has been employed at IBM for 34 years.



Isabel Arnold has been working as a CICS TS Technical Sales Specialist since 2004. She started working in the field of IT in 2001 and holds a German Diploma from the University of Corporate Education, Stuttgart. Her areas of expertise include CICS TS with a focus on integration and modernization, in addition to modern application development for IBM System z®. In her role, she has worked on three IBM International Technical Support Organization (ITSO) Redbooks, and has been presenting at conferences, such as the System z and WebSphere Technical Conferences. Isabel teaches System z software basics at German universities, is an IBM representative for the German CICS TS user group, and leads a team of CICS TS TechSales specialists across Europe.



Andrew Bates is the IBM Product Manager for CICS TS based in Hursley, UK. Andrew has worked in the CICS TS organization for almost 15 years. This includes a three-year assignment to IBM China in 2007, where Andrew was the IBM CICS TS portfolio Business Development Manager for the Asia Pacific region. Andrew has been the CICS TS Product Manager since returning to the UK at the end of 2009.



Michael Baylis is a software developer in CICS TS, with a current responsibility for enhancing the automated test infrastructure used by the CICS TS development team.



Pradeep Gohil is a software developer for IBM CICS Transaction Server for z/OS®. Since joining IBM in 2002, he has developed CICS Systems Management, CICS TS Feature Packs, and several releases of CICS TS. With a keen interest in application modernization and web services, he most recently has been part of the development team that provided the CICS TS cloud enablement enhancements, specializing in CICS TS bundles.



Christopher Hodgins is a software developer for CICS TS for z/OS, specializing in CICSplex System Manager. Since joining IBM in 2005, he has been the technical lead for the CICS Management Client Interface (CMCI) in CICS TS V4.1 and V4.2, and cloud technical lead in CICS TS V5.1. He also regularly blogs about the CICSplex and the cloud.



Daniel Millwood joined IBM in 1995, working as a developer for IBM WebSphere MQ for z/OS and the messaging components of IBM WebSphere Application Server. He joined the CICS TS team in 2007, initially to lead a cross-product integration test team, testing CICS TS web services interoperability. Later, he joined the CICS TS development team, where he led the development of the CICS TS Feature Pack for Modern Batch, and helped develop the CICS TS cloud capabilities in CICS TS V5.1 and V5.2.



Ian J Mitchell is an IBM Distinguished Engineer with responsibility for the technical architecture of the CICS TS product portfolio. He has more than 20 years of experience as a technical leader in IBM Hursley, and has been in the forefront of creating and introducing many important CICS TS innovations, including workload management, IBM Parallel Sysplex® support, business transaction services, Java and Enterprise JavaBeans (EJB), web services, event processing, and the CICS Explorer. In this role, Ian has worked with many technical leaders across most mainframe technology components. Ian also has close relationships with many of the largest IBM customers running critical systems using CICS TS and the mainframe.



Catherine Moxey is an IBM Senior Technical Staff Member in the CICS Transaction Server for z/OS team, based in Hursley, UK. She holds an M.A. in Chemistry from Oxford University, and has 30 years of experience as a software engineer, 25 of those with IBM. Her areas of expertise include CICS TS, System z, performance, event processing, and cloud enablement. Catherine frequently presents at conferences, and has written several articles and papers on the capabilities of CICS TS.



Geoffrey Pirie is a CICS TS product marketing manager with the IBM Software Group - Application and Integration Middleware Software, at IBM Hursley Park, United Kingdom. He joined IBM as a developer, and later moved into the management of the CICS TS System Test and Performance team. Today, his experience and growth led him to join the marketing team for CICS TS, enjoying and experiencing the management of the CICS TS portfolio.



Inderpal Singh is a software engineer for the CICS TS portfolio at IBM Hursley UK. Often heard ranting about the mainframe's importance for the future, Indi is a keen advocate for adopting new technologies on System z.



Stewart Smith started as an IBM employee in 2003 after contracting since 1988. He moved through application then systems programming on IBM z/VM®, IBM MVS™, CICS TS, IBM DB2®, IBM VTAM®. He specialized in performance and stress testing using IBM Workload Simulator for z/OS and IBM Teleprocessing Network Simulator (TPNS) before joining the CICS TS System Test team. Now working on the CICS TS Client Success Team, he delivers services engagements to customers for CICS TS health checks and HA competence studies.



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

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

Tamikia Barrow-Lee, Richard Conway, Robert Haimowitz
ITSO, Raleigh and Poughkeepsie Center

Fraser Bohm, Senior Technical Staff Member (STSM) CICS TS Application Modernization
IBM Hursley

Claudia Prawirakusumah
IBM Germany

Phil Wakelin, CICS TS Strategy and Planning: Java, access to CICS TS
IBM Hursley

Now you can become a published author, too!

Here's an opportunity to spotlight your skills, grow your career, and become a published author -- all at the same time. Join an ITSO residency project and help write a book in your area of expertise, while honing your experience using leading-edge technologies.

Your efforts will help to increase product acceptance and customer satisfaction, as you expand your network of technical contacts and relationships. Residencies run from two to six weeks in length, and you can participate either in person or as a remote resident working from your home base.

Discover more about the residency program, browse the residency index, and apply online:

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:

ibm.com/redbooks

- Send your comments in an email:

redbooks@us.ibm.com

- Mail your comments:

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

Introduction

This part of this IBM Redbooks publication provides information about cloud enabling your IBM Customer Information Control System (CICS) Transaction Server (CICS TS) applications, and the introduction and use of the General Insurance Application (GENAPP), an application available with IBM SupportPac CB12.



Cloud enabling your CICS TS applications

IBM Customer Information Control System (CICS) Transaction Server (CICS TS) for z/OS is the premier application hosting platform for IBM z/OS for high-volume updates to shared data. It is designed to enable customers to create and maintain a competitive advantage by building custom, differentiated CICS TS applications.

For many decades, IBM has continuously delivered market-leading CICS TS technologies that companies have built upon to differentiate themselves in the marketplace. As technology has evolved to deliver increasingly complex, interconnected, and distributed services, the industry has developed new approaches to building these solutions.

The introduction of service-oriented architectures (SOA) laid the foundation for what has become the next transformational era in technology, as businesses around the world seek to deliver solutions based on the principles of cloud computing. CICS TS V5 takes the concepts and values promised by cloud computing, and provides new cloud capabilities designed to increase operational efficiency and service agility without going off-premise.

This IBM Redbooks publication introduces the concepts that form the foundations of the cloud capabilities in CICS TS. It also provides real examples of how to take an existing CICS TS topology and application and apply these capabilities to it, using the General Insurance Application (GENAPP) support pack.

1.1 Did you know?

CICS TS is used by businesses around the world to deliver mission-critical applications and services. With a heritage that spans over 40 years, CICS TS is unparalleled in the industry, delivering a mixed-language transaction processing and application hosting platform.

Although it might have been around longer than most, CICS TS is no stranger to new technologies, offering users the following advantages:

- ▶ Multiple languages
 - Java
 - C++
 - Common Business Oriented Language (COBOL)
 - PHP
- ▶ A broad range of connectivity and integration options
 - Web services
 - Representational State Transfer (RESTful) web services
 - Database connectivity
 - Messaging integration
- ▶ Security
- ▶ An advanced multi-threaded run time (with transaction isolation and storage protection)
- ▶ Event processing
- ▶ Exceptional scalability (customers regularly drive 1 billion+ transactions per-day)
- ▶ Integration with the broad range of IBM application development and lifecycle management tools

In the era of cloud and mobile, CICS TS delivers the essential services necessary to build robust, mission-critical applications. The cloud capabilities introduced in CICS TS V5 are the next logical step in the evolution of the CICS Transaction Server, offering increased efficiency and agility to the enterprise.

1.2 Business value

CICS Transaction Server V5 has been designed to address the following main challenges that customers are facing today:

- ▶ Ongoing operational pressure to drive down costs and improve efficiency
- ▶ Increased business pressure to deliver faster and with greater agility
- ▶ The ability to manage and use the rapidly changing technology landscape

Cloud computing, although it is a relatively new concept, offers many values that existing CICS TS users claim to enjoy already, so why introduce new cloud-style concepts into CICS TS today? At a high level, these concepts are being introduced because cloud is more than just a technology.

Cloud offers a conceptual shift in how a business delivers services. It promises increased operational efficiency over the management and operation of these services, and increased agility when developing and deploying them. By introducing cloud capabilities, CICS TS enables both new and existing customers to gain the benefits offered by cloud computing, while maintaining the solid foundation that CICS TS offers.

Specifically, CICS TS provides the following cloud capabilities and benefits:

- Platforms as first-class entities

First-class platforms enable the creation of agile service delivery runtimes. CICS TS regions can be grouped as platforms for rapid application deployments, decoupling applications from the underlying topology, which increases flexibility. When regions within a platform are started, applications are deployed to them, without any further interaction from a system administrator. In turn, reliability is increased by automatic resource validation, provisioning, and de-provisioning. Platforms can be managed dynamically by applying policies during run time.

- Applications as first-class entities

First-class applications enable the creation of agile services from new or existing assets. Disparate application resources can be combined and managed as a single entity, which can be versioned and rapidly moved through the development, test, and production lifecycle. Using applications improves dependency management, and entire applications can be measured for resource usage and internal billing. Applications can be managed dynamically by applying policies during run time.

- Policy-based operations

Automated control over critical system resources can now be managed using *policies*. Task thresholds can be set for data access requests, storage usage, program loops, and processor time used. Policy breaches can be managed by issuing messages, using tasks to cause an abnormal end (abend), or emitting events that can trigger further actions. Policies can be applied dynamically during runtime operation.

These three capabilities, when combined with the existing features of CICS Transaction Server, provide the building blocks to enable you to transform your existing CICS TS topologies and applications into cloud-style platforms and services.

1.3 Solution overview

This IBM Redbooks publication begins by examining the National Institute of Standards and Technology (NIST, part of the US Department of Commerce) Special Publication (SP) 800-145 recommendations, which represent the widely accepted definition of cloud computing:

<http://csrc.nist.gov/publications/nistpubs/800-145/SP800-145.pdf>

This book compares these recommendations to the existing features and capabilities that CICS TS offers. As you will see, the strong heritage that CICS TS has, positions it extremely well as a cloud platform.

Having understood how CICS TS relates to the concepts of cloud, this book proceeds to look at each of the features introduced in CICS TS V5 that support its cloud capabilities:

- Chapter 2, “GENAPP introduction” on page 15 introduces the GENAPP in detail, covering the application’s current topology and architecture. This application is the bedrock for all later chapters in this book, and is used to demonstrate how existing applications can quickly benefit from the additional offerings provided by the CICS TS cloud.
- Chapter 3, “Creating a platform” on page 25 introduces the platform concept, describing the role of platforms as a host for applications and provider of application services. This chapter also details the components of the platform, then provides an introduction to the concept of a development platform. It also provides a step-by-step guide to define, deploy, and install this platform.

- ▶ Chapter 4, “Creating an application” on page 51 introduces the application concept, and explores the business value of, and the components that make up, an application. This chapter also provides a step-by-step guide to define, deploy, and install an application. The created application is built around GENAPP, introduced in Chapter 2, “GENAPP introduction” on page 15.
- ▶ Chapter 5, “Applying a policy” on page 97 introduces the new policy capabilities, and how these can be applied to provide more efficient operation in the CICS TS cloud. This chapter also provides a step-by-step guide to define, deploy, and add your policies to both platforms and applications.
- ▶ Chapter 6, “Packaging an application for multiversion deployment” on page 127 expands on the application defined in Chapter 4, “Creating an application” on page 51, and introduces the concept of application multi-versioning. This chapter walks you through the process of repackaging your application resources, and of rolling out a new version while the previous version is still active.
- ▶ Chapter 7, “Measurement by application” on page 175 explains how to take advantage of the application context for performance monitoring and chargeback. This chapter describes the improvements over traditional techniques, and provides examples using the application introduced in Chapter 2, “GENAPP introduction” on page 15.
- ▶ Chapter 8, “Managing by policy” on page 181 looks again at policies in more detail from a cloud perspective. This chapter examines the types of policies, and the actions that can be taken when a policy rule is breached. It looks at scoping policies for platforms and applications, and provides good usage scenarios.
- ▶ Chapter 9, “Application lifecycle management” on page 193 provides information about application versioning in more detail. This chapter explains semantic versioning, and takes the application introduced in Chapter 2, “GENAPP introduction” on page 15 through various upgrade scenarios, from releasing a new application version to version backout.

1.4 Cloud computing in a CICS TS context

When planning and implementing CICS Transaction Server for z/OS, version 5.1, the CICS TS design and development team used many different sources to establish and validate the new features that make cloud computing real to customers embracing the product. This included other IBM product and solution experts, industry experts outside of IBM, IBM Business Partners, and clients.

This provided a tremendous amount of insight into what the generally accepted principles of cloud computing are, and also validated that existing CICS TS capabilities were an excellent foundation on which to build additional new features that would transform the CICS TS application-management and system-management facilities to meet the expectations of a cloud computing platform.

During this design process, NIST issued its definition of cloud computing.

This document recommends five *essential characteristics*, three *service models*, and four *deployment models* related to cloud computing. These recommendations are aimed at an audience of system planners, program managers, technologists, and others adopting cloud computing as consumers or providers of cloud services. They provide a reference point for discussion of CICS TS V5.1 cloud enablement features.

The NIST Special Publication acknowledges that cloud computing is an evolving paradigm, and does not intend to prescribe or constrain any particular method of deployment, service delivery, or business operation.

Cloud computing essential characteristics

As CICS TS V3 did for service orientation and the IBM SOA reference architecture, CICS TS V5 takes accepted principles and enables them to be realized in a manner suited to the existing business application assets and system deployment styles used across thousands of CICS TS installations. The aim is to enable the positive outcomes associated with cloud computing to be gained in an incremental and easily managed fashion, minimizing cost and disruption.

The five essential characteristics recommended by NIST provide a good basis to evaluate how CICS TS V5 enables these outcomes:

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

Definition: The following definition is the first essential characteristic from the NIST SP 800-145 recommendations:

<http://csrc.nist.gov/publications/nistpubs/800-145/SP800-145.pdf>

On-demand self-service *A consumer can unilaterally provision computing capabilities, such as server time and network storage, as needed automatically without requiring human interaction with each service provider.*

In a CICS TS context, the processes used to provision services vary widely from installation to installation. It is overwhelmingly common that business stability, audit, and other regulatory requirements demand highly stringent testing and validation of both application and system changes before they are deployed into production.

However, there are competing demands on skills and cycle times that are motivating a need to enable more rapid deployment of new or modified services, or additional system capacity in simplified and more responsive ways. Some installations have very mature and responsive processes to satisfy these needs, while others suffer from bottlenecks in processes, or are constrained in other resources, such as skilled staff.

An outcome enabled by CICS TS V5 is a simpler, more repeatable, and more controlled deployment process that provides a balance between the demands for business agility and the real demands for resilience and stability. New first-class management objects for application assets and system resources enable a more effective separation of concerns between application developers, deployers, and system provisioning and operations staff.

Consumers benefiting from such features might be application developers who require the provisioning of systems for development and unit-test activities, or line-of-business consumers requiring new or changed services to be rolled out to end-users.

Definition: The following definition is the second essential characteristic from the NIST SP 800-145 recommendations:

<http://csrc.nist.gov/publications/nistpubs/800-145/SP800-145.pdf>

Broad network access *Capabilities are available over the network and accessed through standard mechanisms that promote use by heterogeneous thin or thick client platforms (e.g., mobile phones, tablets, notebooks, and workstations).*

Recent versions of CICS TS, especially version 3 and later, have greatly expanded the range of protocols and formats that can be used to expose business services from the application assets that it hosts. Service-enablement options have been heavily used by most customers, commonly using synchronous request-reply styles over Transmission Control Protocol/Internet Protocol (TCP/IP) or message-based asynchronous styles with IBM WebSphere MQ.

Services to clients can be made available using structured payloads, such as SOAP, according to tightly specified definitions, such as Web Services Description Language (WSDL). These services are subject to rigorous governance using service registries, or more informally provided using the principles of the RESTful style. Another outcome enabled by CICS TS V5 is that the development and deployment process for service enablement to achieve broader network access is simplified as application assets become better defined, with a better managed lifecycle and explicit versioning.

Definition: The following definition is the third essential characteristic from the NIST SP 800-145 recommendations:

<http://csrc.nist.gov/publications/nistpubs/800-145/SP800-145.pdf>

Resource pooling *The provider's computing resources are pooled to serve multiple consumers using a multi-tenant model, with different physical and virtual resources dynamically assigned and reassigned according to consumer demand. There is a sense of location independence in that the customer generally has no control or knowledge over the exact location of the provided resources but may be able to specify location at a higher level of abstraction (e.g., country, state, or datacenter). Examples of resources include storage, processing, memory, and network bandwidth.*

The invention of the class of platform known as a *transaction processor* was to provide more efficient use of precious computing resources, and to scale to meet the demands of global enterprises. CICS TS, the z/OS operating system, and IBM System z hardware practically define the gold standard for a mission-critical, mixed-workload application hosting platform.

The concept of a CICS TS transaction and its context has been an effective declaration of an application as a tenant in a multi-tenant environment from the inception of CICS TS. Within a single server, CICS TS has control of the dispatching and resource allocation to tasks running the transaction. Features, such as hardware-mediated transaction isolation, support the requirements for multiple requests to share resources without the risk of compromising each other.

The adoption of the CICS TS Open Transaction Environment (OTE) style of execution, which enables workloads to scale more effectively with the hardware cores provided by System z and z/OS, has meant that single CICS TS regions can provide many more millions of instructions per second (MIPS) to any given workload. Recent use of the expanded storage available to a region in the 64-bit address space means that those MIPS can be used to serve the ever-growing storage needs of today's applications.

CICS TS exploitation of z/OS Parallel Sysplex topologies has meant that it can support very large workloads with stringent high-availability (HA) requirements. IBM CICSplex System Manager (CICSplex SM) works in partnership with z/OS Workload Manager (WLM) to achieve both efficient use of resources for workloads classified as the most critical, and continuation of service during planned or unplanned outages of processing resources.

Although the exploitation of OTE and Sysplex-enablement features is a partnership between CICS TS and the applications, the results are immediate benefits to the service consumers. There is no need for a consumer to be aware of their request being one of a few hundred or thousand such requests per day, or one of a billion or so.

CICS TS V5.1 provides new policy-based control of these resources in terms of the applications running on platforms. This enables the outcome of improved control of pooled resources, such as processing power and storage. For example, 24-bit storage is often a constraint on the workload that an individual region can support. Programming standards can be in place to restrict or prohibit use of this type of storage, but old applications might not have been modified to eliminate its use.

It is possible to define a policy that tracks and limits 24-bit storage on a task-by-task basis. Older applications might need to be enabled to have a limited amount of such storage, but new or well-maintained applications could be banned from allocating any. It is also possible to associate different threshold policies with each application to accommodate their varying demands.

Definition: The following definition is the fourth essential characteristic from the NIST SP 800-145 recommendations:

<http://csrc.nist.gov/publications/nistpubs/800-145/SP800-145.pdf>

Rapid elasticity	<i>Capabilities can be elastically provisioned and released, in some cases automatically, to scale rapidly outward and inward commensurate with demand. To the consumer, the capabilities available for provisioning often appear to be unlimited and can be appropriated in any quantity at any time.</i>
-------------------------	--

With today's demand for 24 x 7 availability, and the exposure of services to ever-wider groups of consumers to use as and when they expect to use the service, the need to be ready for a workload spike is a common requirement.

The ability for regions to scale, and for workload management to prioritize critical requests to maintain service levels, is provided by the resource pooling capabilities of CICS TS. However, long-term capacity management will inevitably demand a more dynamic environment, with the number of regions growing and shrinking to match demand.

In the past, an installation's capacity could largely be measured by the number of regions employed. Region usage and topology patterns could also be a measure of the resilience and sophistication of the design of the applications and infrastructure.

As a consequence of regions as the *unit-of-scale* in infrastructure terms, it is not surprising that in many cases it is a region-centric view of capacity that dominates, and decisions to provision and decommission regions are inelastic. This can lead to suboptimal infrastructure deployments and inefficiencies.

By creating a new management object (a platform) over the individual regions, but one which fully embraces the reality of the structure of types of regions within the platform topology, CICS TS V5 is anticipating a future where regions or entire platforms are more actively managed on a day-by-day basis according to need, but without requiring a large provisioning effort.

A common example where more dynamic management of regions and platforms has a positive outcome is the provisioning of development and test regions. If an infrastructure process makes it burdensome to request resources for each project to use for development and test, projects are likely to retain those resources for longer than is optimal.

Alternatively, a more dynamic environment promotes flexibility to give up under-utilized resources on the understanding that they can be easily reacquired when necessary. This leads to a net reduction in the overall resource consumption, while maintaining an agile development environment.

Definition: The following definition is the fifth essential characteristic from the NIST SP 800-145 recommendations:

<http://csrc.nist.gov/publications/nistpubs/800-145/SP800-145.pdf>

Measured service *Cloud systems automatically control and optimize resource use by leveraging a metering capability at some level of abstraction appropriate to the type of service (e.g., storage, processing, bandwidth, and active user accounts). Resource usage can be monitored, controlled, and reported, providing transparency for both the provider and consumer of the utilized service.*

As noted previously, the measurement and reporting on a transaction-oriented basis has long been a strength of CICS TS. However, as the transaction view of the world has been superseded by a service-oriented view, there has been a need to deal in more expressive terms than a four-character transaction name.

In recent releases, CICS TS has improved the transaction-tracking capabilities for both monitoring and problem resolution purposes. Now with CICS TS V5.1, the application provides a context that is a better reflection of the business view of the application.

Every release brings new metrics for resource usage or time spent in certain functions. For example, in CICS TS V5.1, new data is available to account for time spent running on a *general-purpose processor* versus time on a *specialty engine* for each task in the CICS TS monitoring records. This is a finer-grained accounting than was previously possible with the address space IBM z/OS Resource Measurement Facility™ (RMF™) data.

Comparing with other cloud solutions

Many other cloud technologies currently do not support the same granularity of provisioning. They commonly define entire virtualized application stacks, including operating system (OS), middleware, database, and application, as a single deployable package.

This use case is valid, because applications are defined on virtualized UNIX or Windows servers. However, it does not match the long-established, mixed-workload efficiency design point of the z/OS operating system and CICS TS.

The platform and application specifications that are provided by CICS TS V5.1 enable the existing system programmers to own, structure and maintain platforms, with the applications owned and maintained separately. All such assets are appropriate for formal maintenance using your source code change management tools and processes.

The move to more clearly differentiate platforms from the applications that are hosted on them can provide motivation for more clearly differentiated roles for staff involved in the creation, deployment, maintenance, and operation of the platforms and applications.

It can be common for the staff maintaining the CICS TS system infrastructure to be closely involved in the design and deployment of applications. In addition, application developers are relatively unfamiliar with some of the important resource usage and binding concerns that are resolved as the application is deployed.

A better distinction between these concerns is being enabled with the platform and application patterns implemented in CICS TS V5.1. Existing system programmers would likely be ideal candidates to own platforms, but the ownership of applications should be more closely aligned with the application development team.

However, it is not proposed that every application developer needs to become an expert in the definition of the new resources or the new threshold policies. The distinct role of application deployment, especially to production platforms, might be best given to staff in a newly identified role of Application Deployer. Of course, in some cases this might just be formalizing an informal but well-understood role that exists in your organization.

1.5 Overview of the cloud-enabling technologies in CICS TS V5

As explained previously, in many respects deploying CICS TS platforms and applications is an evolution of the existing leading practices enhanced with new management objects (namely platforms, applications, and policies). This evolution has been achieved by building on some solid foundations.

Platforms have been created by employing the foundations of CICSplex SM topology definitions, such as System Groups (CSYSGRP), and the operations to deploy and manage both platforms and applications built on the single-point-of-control capabilities of CICSplex SM. It is a prerequisite to deploy CICSplex SM to fully use the cloud enablement features of CICS TS V5.

The rich objects introduced in CICS TS V5.1 demand the facilities of a similarly rich user experience, such as that provided by the IBM CICS Explorer and CICS Explorer® software development kit (SDK). Applications, platforms, and policies are defined using Extensible Markup Language (XML) files, of a similar style to those already used for bundle-based deployment of web services and event processing in Version 3 and Version 4.

The CICS Explorer SDK provides *wizards* for creation, *editors* for easy maintenance of the assets, and *export capabilities* that simplify the deployment of the assets into a runtime environment.

1.5.1 Platform overview

Platforms are a prerequisite for hosting applications, providing well-defined scopes called *region types*, for the installation of application resources. Platforms can additionally host policies and services, or declare dependencies for existing services. They also provide a single point of reference for status about system activity and services.

Figure 1-1 shows an existing three-tier CICS TS region topology managed by a CICS TS platform. Superficially, this does not look much different from an existing CICSplex management structure. However, the difference becomes apparent when we start to appreciate how CICS TS can use these artifacts to simplify the management of both the CICS TS regions and the applications that run within them.

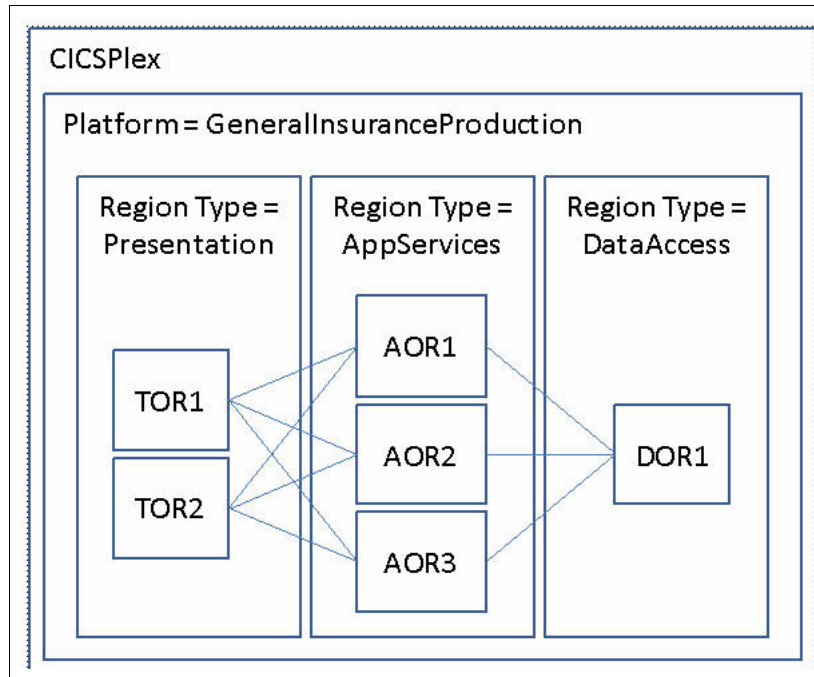


Figure 1-1 Conceptual diagram of existing CICS TS region topology managed by a CICS TS platform

The most obvious new element in Figure 1-1, aside from the platform itself, is the region type. A region type is a logical grouping of a collection of CICS regions that share common characteristics. In the example here, we have defined three region types to represent the three-tier nature of our application:

- ▶ Presentation
- ▶ Application logic
- ▶ Data

How you choose to define a region type is up to you, so you might also choose to have a region type of PayrollApplication, or perhaps to represent different geographical areas. There is a great deal of flexibility about how you define a region type, and we explain this in detail in Chapter 2, “GENAPP introduction” on page 15.

One of the powerful features of the platforms becomes apparent when you enable or disable the platform resource. When a platform is enabled, CICS TS automatically enables any bundle resources that were installed with the platform, for example, policies across the CICS TS regions in the appropriate region type. We look at policies in more detail in Chapter 4, “Creating an application” on page 51.

1.5.2 Application overview

Applications provide a powerful and capable container around CICS TS bundled resources. These resources track the application lifecycle, from installation to eventual discard. They can host resources, application dependencies, and policies. The overall health of these can be tracked using the application's status.

The application can additionally provide *entry points*, whose declared *operations*, for example QueryCustomer or AuthorisePayment, can be used for simple and accurate resource monitoring and billing through System Management Facilities (SMF) records. Policies can also be deployed as part of an application to take effect only on those *tasks* related to specific operations. Finally, applications support advanced multi-versioning, enabling the user to simultaneously run current and newly patched versions of the same application.

Figure 1-2 shows an example application and an accompanying binding. The *application* is itself a collection of CICS TS *bundles*, each of which define or import a set of *resources*. Imported resources enable the application to declare a dependency on a specific resource that is not defined by this application, for example a file that is shared by multiple applications.

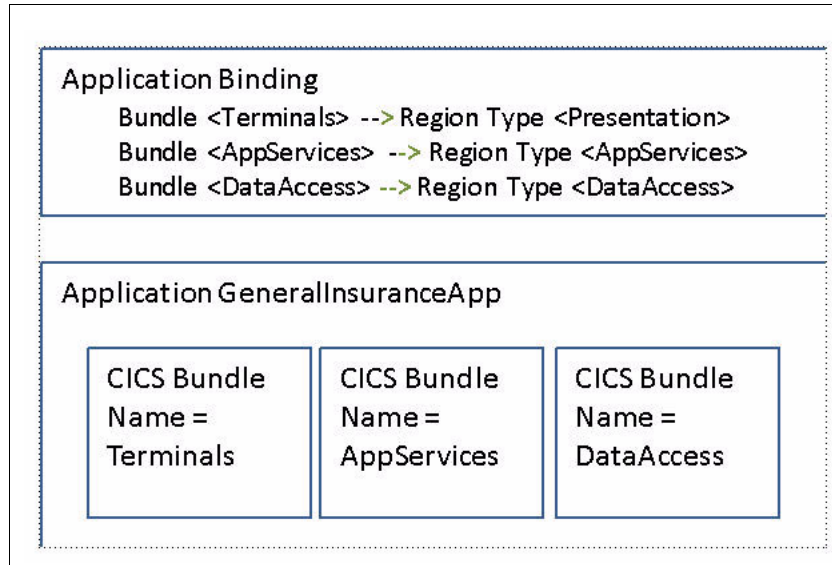


Figure 1-2 Conceptual diagram of an application and application binding

The *application binding* is used by CICS TS when an application is deployed onto a platform. It enables CICS TS to manage the installation of the application's resources into the appropriate CICS TS regions managed by the platform. As you can see, the binding represents the mapping of an application component, for example Terminals, to the associated region type, in this case Presentation.

When an application is installed onto a platform, CICS TS ensures that all of the resources that the application depends upon (imports) are available in the regions of the target region type. If a resource cannot be located, the application reports a failure.

Chapter 4, "Creating an application" on page 51 explains applications in more detail.



GENAPP introduction

The scenarios in this IBM Redbooks publication use a general insurance application available with IBM GENAPP SupportPac CB12, which can be found on the following website:

<http://www-01.ibm.com/support/docview.wss?uid=swg24031760>

The General Insurance Application GENAPP is an IBM Customer Information Control System (CICS) Transaction Server (CICS TS) Common Business Oriented Language (COBOL) application that simulates transactions made by an insurance company to create and manage customer and insurance policy data. It provides sample data and an IBM 3270 interface for creating and inquiring on customers and insurance policy information.

This chapter introduces the components of the application and its topology. To install and set up the application, see Appendix A, “Setup and environment” on page 207.

This chapter contains the following topics:

- ▶ 2.1, “CICS TS topology” on page 16
- ▶ 2.2, “Application architecture” on page 17
- ▶ 2.3, “Limitations of the current design” on page 21

2.1 CICS TS topology

GENAPP can be installed into a stand-alone CICS TS region System Management Single Server (SMSS) that is not managed using CICSplex SM. However, the concepts of *applications* and *platforms* discussed in this IBM Redbooks publication require the region to be managed by IBM CICSplex System Manager (CICSplex SM). Therefore, all steps in the scenarios are applied to a single region, but it is still a part of a CICSplex, and is referred to as a *single managed region*.

In addition, the scenarios separate the GENAPP application into a CICS TS *topology* consisting of three different types of CICS TS regions:

- Presentation
- Application services
- Data access

Chapter 3, “Creating a platform” on page 25 shows how to create a *development platform*, where all three region types are provided by a single managed region, and a *test platform*, where each region type has its own CICS TS region, connected using multiregion operation (MRO) or Internet Protocol interconnectivity (IPIC). One might imagine a production platform taking this a step further, and having multiple regions for each region type, with requests dynamically routed between the CICS TS regions.

Figure 2-1 shows the infrastructure of the example used in this IBM Redbooks publication, where the single managed region is used for development, the simple CICS TS topology for test, and the full CICSplex SM topology in production.

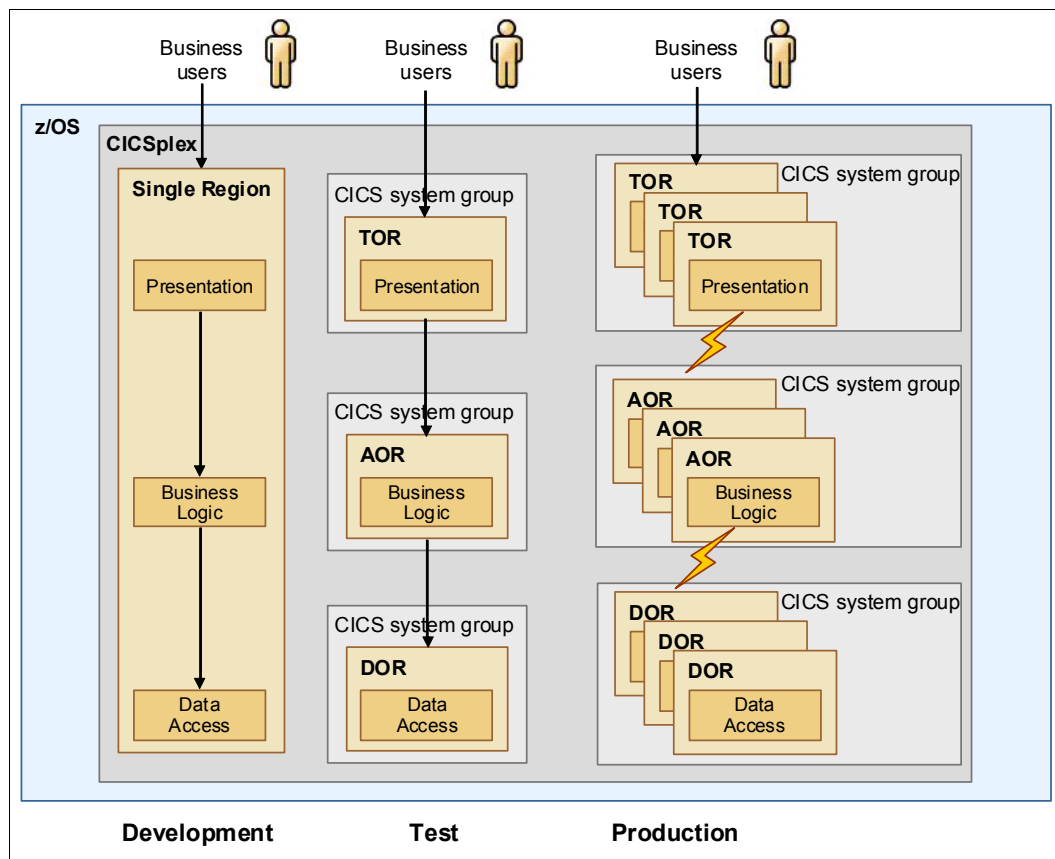


Figure 2-1 CICS TS environments used in this IBM Redbooks publication

Important: Although outlined in Figure 2-1 on page 16, this IBM Redbooks publication will not cover the production scenario. You will work with GENAPP in development and test.

In an actual environment, those different environments would not be part of the same CICSplex.

To install GENAPP to a single managed region and a CICS TS topology, see Appendix A, “Setup and environment” on page 207.

If you follow the instructions described in Appendix A, “Setup and environment” on page 207, you will have defined some CICS TS *system groups* to logically group regions of the same type together in the CICSplex, as shown in Figure 2-2:

- ▶ GENATOR
- ▶ GENAOR
- ▶ GENADOR
- ▶ GENAALL

Chapter 3, “Creating a platform” on page 25, takes you through the process of adding more CICS TS system groups to enable a single managed region to be used as a development platform with three different region types, as shown in Figure 2-1 on page 16.

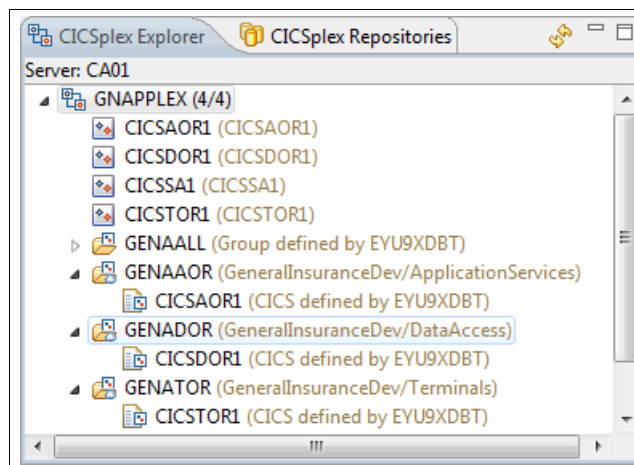


Figure 2-2 CICS TS system groups defined during the installation of GENAPP

2.2 Application architecture

This section describes the basic architecture of GENAPP as it has been used for this IBM Redbooks publication.

2.2.1 GENAPP in a single managed region

GENAPP can be divided into two different application parts:

- ▶ The *customer* application to inquire and add customers
- ▶ The *insurance policy* application to inquire, add, update, and delete policies

The scenarios in this book use the customer application part of GENAPP.

Figure 2-3 shows the architecture of this part of the application.

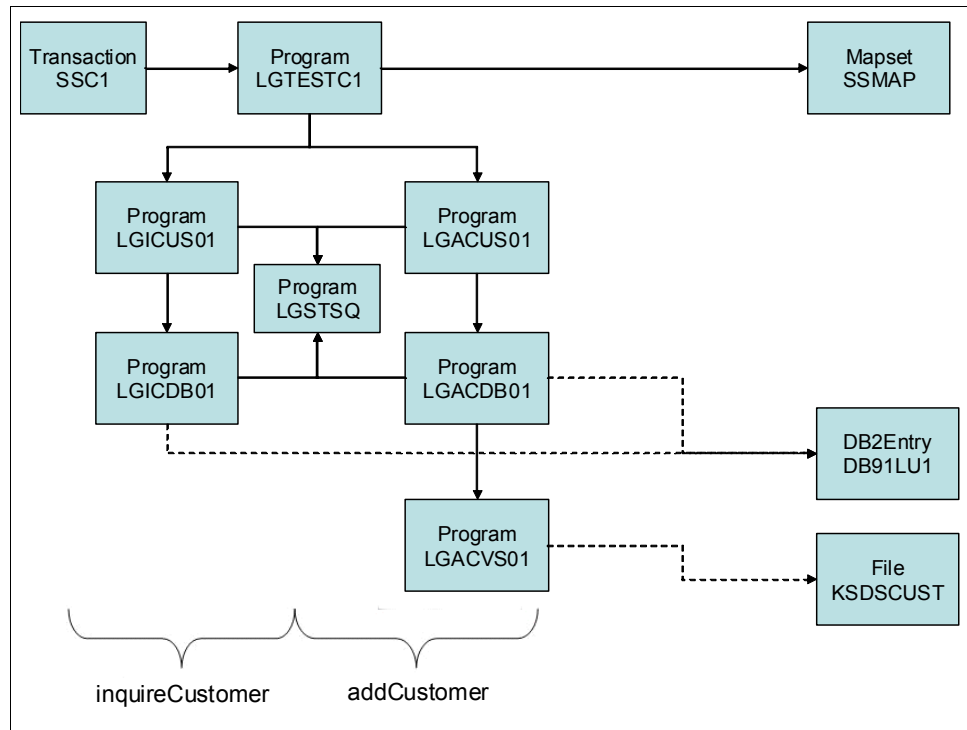


Figure 2-3 Customer application part of GENAPP

The following prefixes are used:

- ▶ LGI prefixes inquiry
- ▶ LGA prefixes add

LGSTSQ is a utility program used by most of the other programs. Figure 2-4 shows a part of a visualization from CICS Interdependency Analyzer (IBM CICS IA®) that collects information about the customer application part of GENAPP. The LGSTSQ program does not show up, because IA only collects programs that ran at run time. LGSTSQ is only called in case of an error.

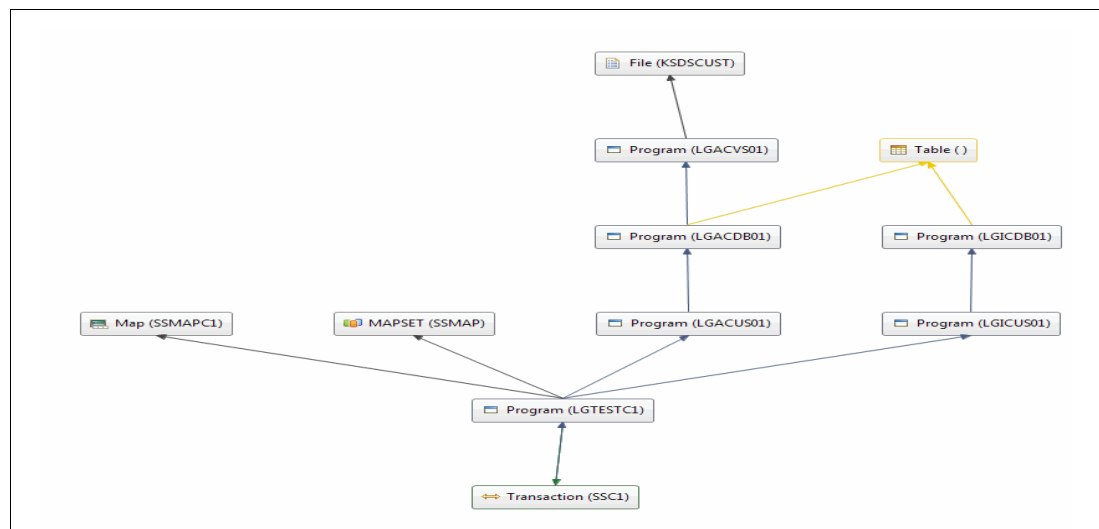


Figure 2-4 CICS IA visualization of GENAPP customer part

2.2.2 GENAPP in a CICS TS topology

Figure 2-5 shows GENAPP deployed across a test platform consisting of three different region types (where each region type consists of a single CICS TS region), as described in Chapter 3, “Creating a platform” on page 25. This strategy separates the different layers into unique address spaces:

- Presentation layer, or terminal-owning region (TOR)
- Application services (business logic) layer, or application-owning region (AOR)
- Data access layer, or data-owning region (DOR)

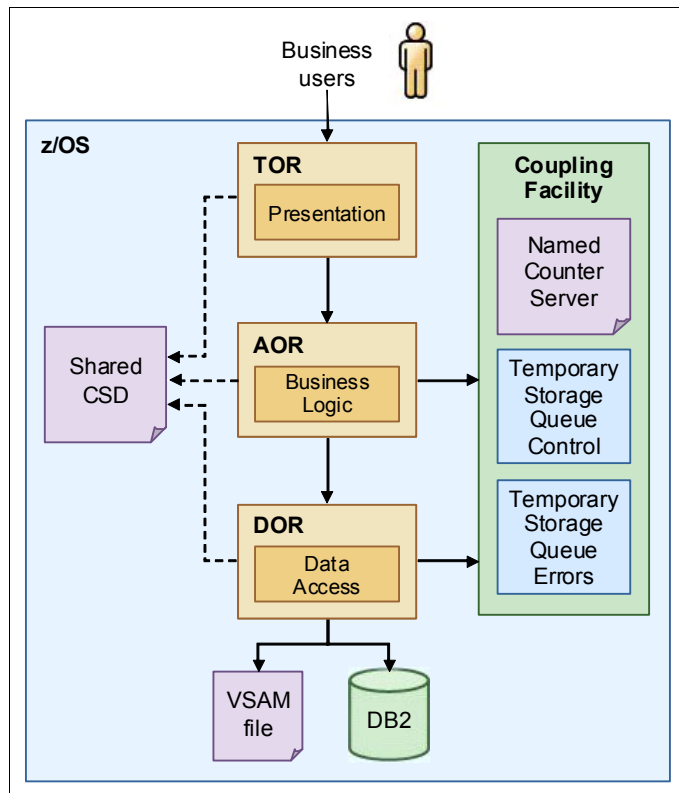


Figure 2-5 CICS TS topology connected by MRO

Remember: This scenario assumes that you are using a shared CICS system definition (CSD) data set as a repository, rather than CICSplex SM Business Application Services (BAS) to define resources to CICS TS, but either approach is equally valid.

As outlined in Figure 2-6, the application design means that it can easily be separated into presentation, business, and data access logic.

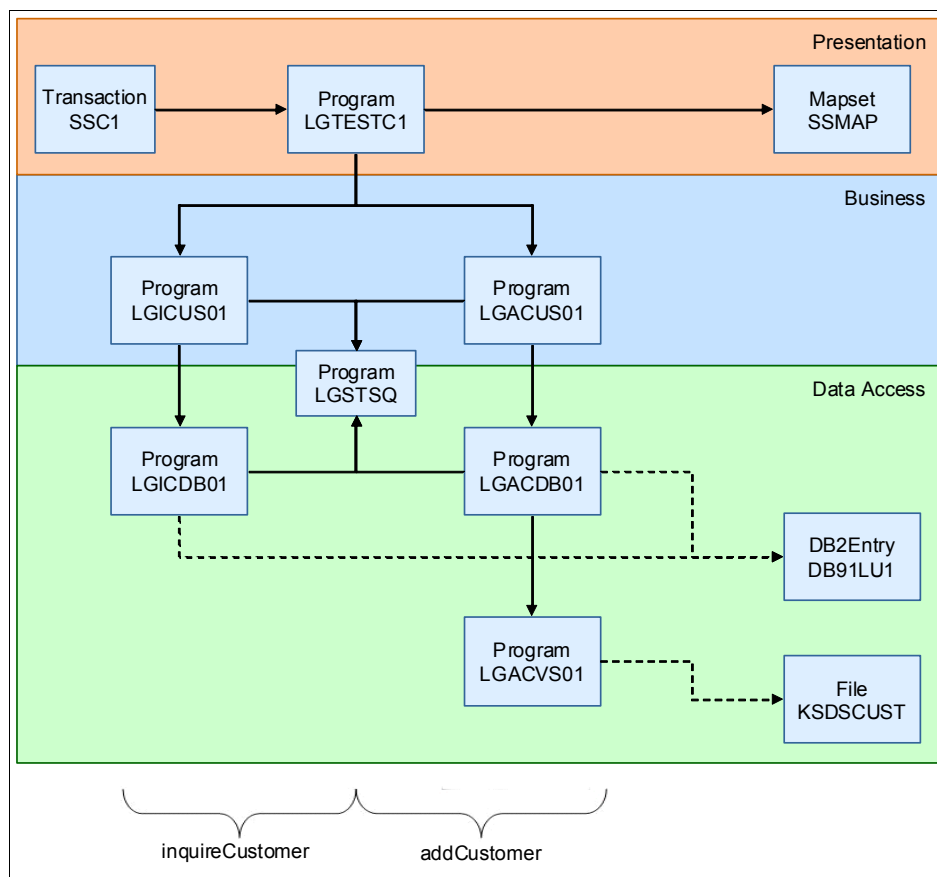


Figure 2-6 GENAPP customer application part: Separated

The GENAPP artifacts is distributed among those layers in the following ways:

- ▶ The presentation layer contains the following definitions:
 - Transaction definitions
 - Local program definitions for the presentation logic (LGTESTC1 and the BMS program SSMAP1)
 - Remote program definitions to route work to the application services layer (LGICUS01, LGACUS01)
- ▶ The application (business) services layer contains the following definitions:
 - Local program definitions for the business logic (LGICUS01, LGACUS01)
 - Remote program definitions to route work to the data access layer (LGIDDB01, LGACDB01, LGACVS01)
- ▶ The data access layer contains the following definitions:
 - Local program definitions for the data access logic (for example, LGIDDB01, LGACDB01, LGACVS01)
 - Definitions for data to be accessed (VSAM file KSDSCUST, DB2Connection and DB2Entry)

Fast Path: The local program definitions can be omitted if the CICS TS autoinstall program function is enabled.

Figure 2-7 shows the topology in a CICSplex SM context.

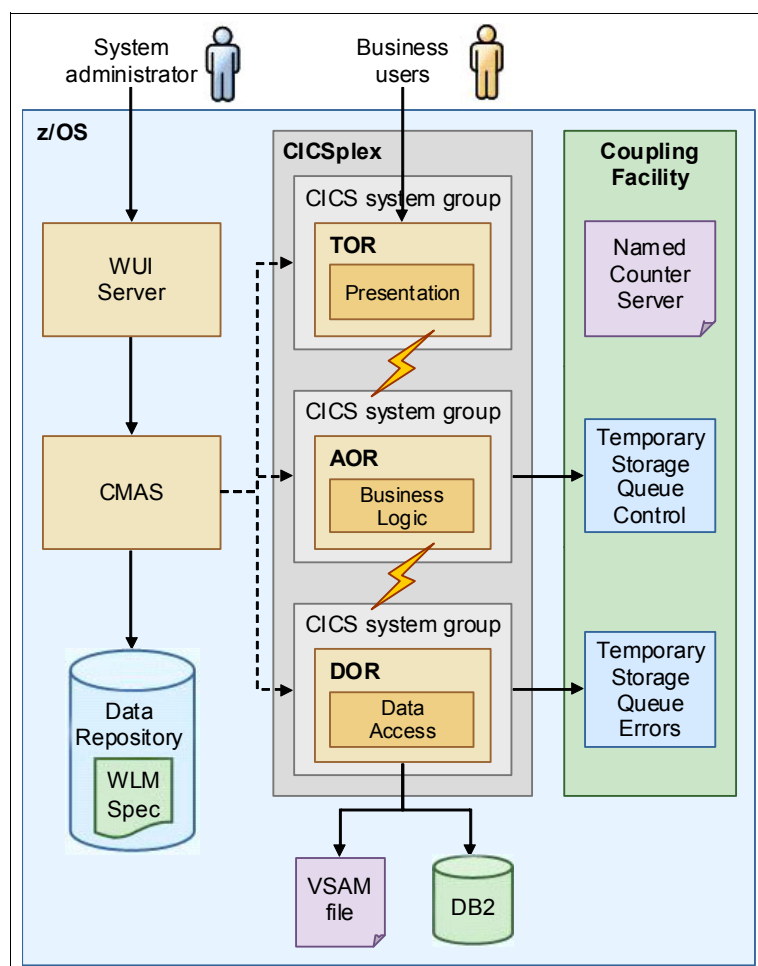


Figure 2-7 CICS TS topology in the context of CICSplex SM

2.3 Limitations of the current design

This section provides information about some limitations in the current design of the GENAPP application, and how these can be addressed using CICS TS cloud capabilities.

2.3.1 Meaningful names

Strict naming conventions enable you to derive the functions of the programs, transactions, and other resources used in GENAPP. But eight characters (or even four, in the case of transactions) do not allow for much flexibility. For example, it would be much more meaningful to identify resources as part of the GeneralInsuranceCustomer application in SMF data, in addition to knowing that they ran under the SSC1 CICS TS transaction.

Chapter 4, “Creating an application” on page 51 addresses the concept of *applications*.

The same applies to the underlying CICS TS infrastructure, for example, it would be more meaningful to deploy part of the application onto the ApplicationServices regions rather than the CICSAAOR CPSM system group.

Chapter 3, “Creating a platform” on page 25 addresses the concept of *platforms*.

2.3.2 Taking the application through the lifecycle

From a CICS TS perspective, GENAPP currently consists of a *load library* and some CICS TS *resources* (programs, transactions, files, IBM DB2 connection, and entries). Although these CICS TS resources are organized in *groups* and *lists*, it is not possible to manage them as a single entity after installation.

A connection between resources can only be concluded manually from the carefully designed naming conventions (programs starting with LGIC belong to the customer application). This makes it a challenge to perform some tasks, such as removing the application from CICS TS.

Chapter 6, “Packaging an application for multiversion deployment” on page 127 describes how to package resources and code as an *entity*.

2.3.3 Managing updates to GENAPP

CICS TS resource names have historically had to be unique within a CICS TS region. This makes it impossible to run two versions of GENAPP on the same CICS TS region if they have the same named CICS TS programs and transactions. This in turn makes it difficult to update the application without affecting its availability to users. Sometimes, if a change is to a single CICS TS program, **NEWCOPY** can be used to update just that module, but when changes affect more than one resource, this approach can lead to inconsistent results.

Chapter 6, “Packaging an application for multiversion deployment” on page 127 describes how to package application resources, such that multiple versions of the application can be deployed into the same CICS TS regions.

Chapter 9, “Application lifecycle management” on page 193 provides information about the value in being able to deploy multiple versions of the same application.

2.3.4 Dependencies

Currently, there is no way to make the availability of a CICS TS program dependent on a CICS TS resource, such as a file or DB2 connection. Chapter 4, “Creating an application” on page 51 describes how to declare application dependencies.

2.3.5 Multiple region separation

As discussed in section 2.2.2, “GENAPP in a CICS TS topology” on page 19, it is a manual process to distribute the definitions among the different regions. This topology uses one single CSD that requires that different lists be maintained to separate resources dedicated to specific types of regions. This is addressed by the introduction of *application bindings* in Chapter 4, “Creating an application” on page 51.

2.3.6 Control of resource usage

The GENAPP application makes DB2 calls, and accesses a CICS TS Virtual Storage Access Method (VSAM) file. In addition, requests into GENAPP are from a user, who expects a response within a reasonable period of time. A sysprog might want to protect the CICS TS regions from changes to GENAPP that result in it making an excessive number of DB2 or FILE access calls, or using too much processor or elapsed time. Chapter 5, “Applying a policy” on page 97, demonstrates how this can be achieved using threshold policies.



Part 2

How to cloud-enable your IBM CICS TS

In this part of the book, we introduce the user to IBM Customer Information Control System (CICS) Transaction Server (CICS TS) platforms, applications, and policies, through the use of worked examples, which the user can follow.

Chapter 3, “Creating a platform” on page 25 demonstrates how a CICS TS platform can be used to simplify the management of a set of CICS TS *regions* to which an application is to be deployed. Regions are grouped into *region types*, based on the role that those regions perform. Regions can be quickly added or removed to a region type within a platform, to scale up or scale down the platform.

Chapter 4, “Creating an application” on page 51 introduces the CICS TS application, and demonstrates how a CICS TS application can be created for an existing user application, without change to the user’s application. The CICS TS application enables monitoring at the application level. It enables threshold policies to be applied to the application, to ensure that the application behaves as expected, and that it provides a single point of management and control of the application’s lifecycle.

Chapter 5, “Applying a policy” on page 97 demonstrates how threshold policies can be applied to an application or a platform, to ensure that the applications behave according to policy, and to enforce that behavior if required.

Chapter 6, “Packaging an application for multiversion deployment” on page 127 demonstrates how to enable multiple instances of the same application to be available on the same CICS TS platform at the same time, enabling roll-out of a new application version without any downtime, and providing a quick way to roll back to the previous application version if required.



Creating a platform

This chapter reviews what a IBM Customer Information Control System (CICS) Transaction Server (CICS TS) *platform* is, and the benefits of using it. Defining platforms on to which the General Insurance Application (GENAPP) can be deployed is a prerequisite to the activities described in the following chapters:

- ▶ 3.1, “The CICS TS platform” on page 26
- ▶ 3.2, “The role of a CICS TS platform” on page 26
- ▶ 3.3, “The components of a CICS TS platform” on page 26
- ▶ 3.4, “Example single-region platform project” on page 30
- ▶ 3.5, “Outcome” on page 49

3.1 The CICS TS platform

CICS TS platforms greatly simplify the deployment of CICS TS applications by providing a clear, well-defined abstraction of a topology that applications can deploy to with confidence. Platforms can both create or reuse existing environments. Platform services can be introduced to support applications by providing infrastructure-focused resources, such as Transmission Control Protocol/Internet Protocol (TCP/IP) services.

Benefits of using platforms include the following abilities:

- ▶ View the status of platform system activity from a single point.
- ▶ View the status of platform services and declared dependencies from a single point.
- ▶ Host applications and platform services.
- ▶ Provision policies to all applications installed in the platform.
- ▶ Dynamically add and remove services and dependencies while a platform is installed.
- ▶ Share systems between region types and platforms in a controlled way.
- ▶ Scale hosted applications while the platform is already installed without changes.

3.2 The role of a CICS TS platform

The *region* is the basic unit of server deployment in a CICS TS installation. CICS TS supports large workloads by employing more regions, which communicate between each other. Within the network of active CICS TS regions, certain regions are devoted to particular functions, and to accessing particular resources.

CICS TS regions are IBM z/OS *address spaces* defined to run as Jobs or Started Tasks. They are identified by various attributes, including their job names, their IBM VTAM Applid, or their CICS TS SysID, according to context. Regions require various dedicated resources, such as data sets. Small installations might only require a very small number of regions to run their workloads, and with so few regions their purposes are often obvious.

As workloads grow and the number of required regions increases, the functions of a single region need to be provided by groups of regions for scale and availability. These regions are clones of the original single region. It makes sense to operate on these clones as a single group. Each action performed against the group is performed against all regions in that group. Inside a platform, a group of cloned regions providing the same capability is called a *region type*.

A platform's primary purpose is to provide an environment that many applications can be deployed into. It must provide one or more region types for the application resources to be installed into, and can provide services that the application might depend on to operate.

3.3 The components of a CICS TS platform

A platform provides a capable hosting environment for applications and platform services, providing both activity and service status information from a single place. They provide a lifecycle that minimizes the impact of environmental change on applications, and policies that enforce the contract between the application and that platform. Additionally, platforms support a more descriptive method for grouping regions, providing new opportunities to consolidate without compromising on control of the services that applications depend on.

3.3.1 Status

The platform resource reflects the current state of the platform. It determines both the overall activity of the regions within the region types, and the overall state of installed platform services.

A platform is **ACTIVE** if at least one region in each region type is active and connected to CICSplex SM, as shown in Figure 3-1. With the platform in the **ACTIVE** state, work is able to flow between region types.

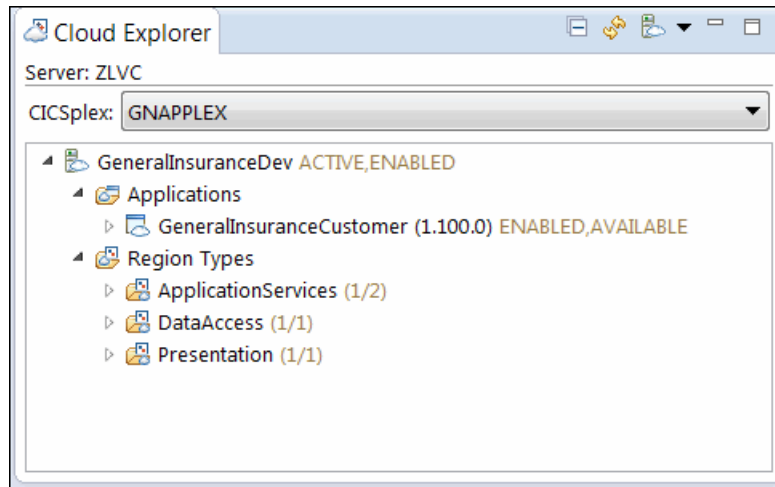


Figure 3-1 An active platform hosting the GENAPP Application

If any region type in the platform has zero active regions, the platform enters a **PARTIAL** state. In this state, deploying applications to the platform should be done only if the application does not depend on that region type. If the application is installed into a **PARTIAL** platform and does install bundles into the region type with zero regions, the application takes on the **INCOMPLETE** state to indicate that part of the application was not installed, as shown in Figure 3-2.

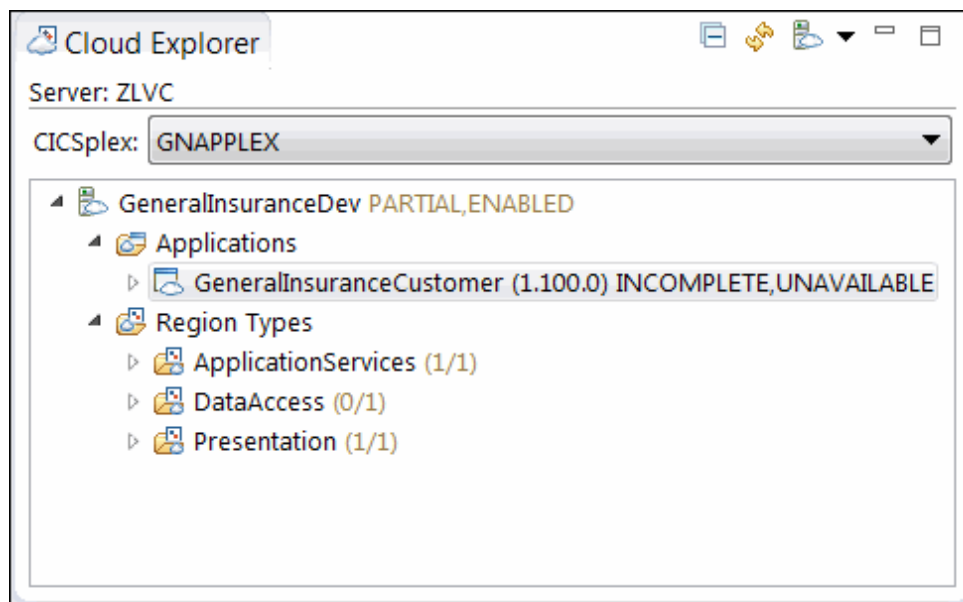


Figure 3-2 A partial platform hosting the now incomplete GENAPP

The platform becomes **INACTIVE** when all regions in all region types are shut down, or are otherwise unavailable to IBM CICSplex System Manager (CICSplex SM). Applications installed into an empty platform take on the empty state because nothing is installed, as shown in Figure 3-3.

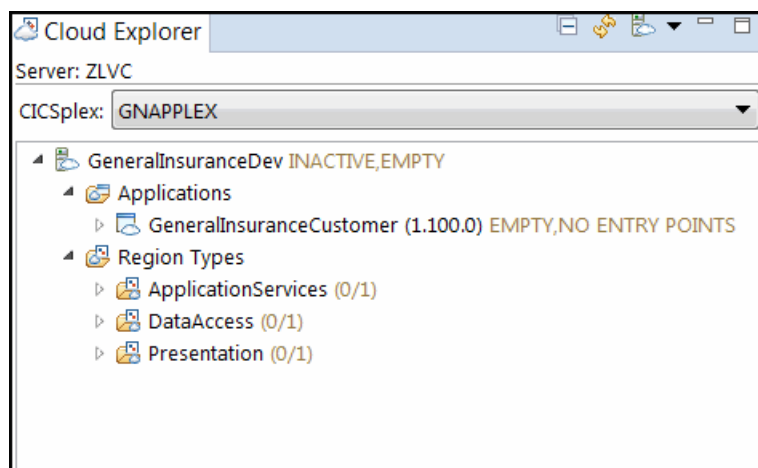


Figure 3-3 An inactive platform hosting the now empty GENAPP application

The platform also tracks the services that it manages, and any declared dependencies. Platform services and dependencies are installed using *bundles*, and it is the health of these bundles that is reflected by the platform **ENABLESTATUS**. When all bundles are enabled across the platform region types, the platform states that it is **ENABLED**. If any bundle fails to become **ENABLED**, the platform reports a state of **SOMEDISABLED**. If all bundles are disabled, the platform reports a state of **DISABLED**.

3.3.2 Lifecycle

Applications require protection from environment change, and this is reflected in the lifecycle of the platform. After a platform is installed, it cannot be discarded until all applications installed within it are also discarded.

While the platform is installed, the region types cannot be modified or discarded. However, using the **ADDREGION** and **REMOVEREGION** actions against an installed platform enables regions to be added and removed from region types. This enables the platform capacity to be expanded in the case of an unplanned growth in workload capacity. Before a region can be removed from a platform, it must be shut down.

Platform services also follow the lifecycle of the platform. When platforms are installed, enabled, disabled, or discarded, all defined services take the same action. For example, when you install a platform and are satisfied that all platform services have been installed, you can enable all platform services immediately by enabling the platform. When all applications have been discarded from a platform, disabling and discarding the platform causes the disabling and discarding of all platform services bundles, ready to be installed again.

If a new region joins the platform or is restarted, the platform services are reinstalled to match the platform's current state.

3.3.3 Scope

Previous to CICS TS V5.1, describing the extent of your infrastructure was only possible using either a CICSplex or a System Group to mark the boundaries. Platforms also provide this capability, but enable a more descriptive and more distinct boundary to be specified.

A CICSplex supports separation of environments, such as production and test or different business units. Within a CICSplex you can define multiple platforms. Compared to the CICSplex, the platform follows a more descriptive route, providing a set of region types to encapsulate the required regions within the scope of the platform. Additionally, platforms can support sharing of regions between region types.

You can use platforms to separate your concerns within a CICSplex. In the development CICSplex, you can choose to allocate a platform per developer, as shown in 3.4, “Example single-region platform project” on page 30. In the test and production CICSplex, you might choose to have a separate application for each line of business.

Finally, if you want to consolidate regions, you can choose to take advantage of a platform’s ability to share regions with other platforms. This could be used, for example, to combine your routing regions for a subset of platforms within the same CICSplex.

When considering whether a CICSplex or a platform would be the best option to scope your regions, consider the following issues:

- ▶ If the regions should be isolated from other regions because they are for a different part of the business or, for example, for development and test regions, using a CICSplex is usually the correct choice. Using a separate CICSplex in this instance ensures that changes made in one scope do not affect the other.
- ▶ If the regions are not required to be isolated, platforms might be a better choice. Platforms can also share resources and regions with other platforms, but only where requested.

3.3.4 Region types

Region types group regions of the same capability and configuration. For example, you can add all routing regions to a single region type, and target regions to another. They define the interface to the platform that applications must be bound to before deployment. When a platform is installed and the region type is created, it creates the region definitions for all defined regions. Alternatively, a region type can also adopt an existing System Group and the regions within it, enabling the reuse of an existing environment.

Where it is beneficial to reuse regions, region types can share regions between each other in the same or different platforms. This is valuable for consolidation scenarios where you might have a group of routing regions servicing several different platforms, or a file owning region hosting files for each platform.

3.3.5 Services

Applications can require specific services. Platforms can install and manage resources to be supplied to the application. For example, a platform might provide a TCP/IP service that the application can consume using a URIMAP. All services installed in this manner can be added, enabled, disabled, and removed from the platform in a single action.

Platform services can be installed with the platform, or added and removed from the platform as required using the `ADDBUNDLE` and `REMOVEBUNDLE` actions. These actions add and remove the bundles associated with a region type in the platform. After an `ADDBUNDLE` action has been issued, the bundle is installed into all regions within the region type. This bundle is then enabled if the platform has previously been enabled. The `REMOVEBUNDLE` action reverses this process by disabling and discarding the bundle in all regions within the region type.

Platform services are not required to be provided by the platform itself. As with applications, platforms can use bundle imports to declare dependencies on existing services provided outside of the platform. For example, if multiple platforms need to share the same File, one platform could create the File definition within a bundle, while the other platforms would import the File definition within their own bundles.

This would enable the platform resource status to also take into account the state of the File. Declared dependencies can be installed with the platform, or added to the platform in the same way as a normal platform service.

For an example of adding a bundle to a platform, see 5.2.1, “Platform policy example” on page 100.

3.3.6 Policies

In addition to providing services, a platform can also install CICS TS policies. These provide a contract for the execution of tasks running in the context of any application within that specific platform. For example, a policy could force an abnormal end of task (abend) if the task exceeds the contracted processor consumption.

Policies are discussed in more detail in Chapter 5, “Applying a policy” on page 97.

3.4 Example single-region platform project

This section provides information about the implementation of CICS TS platforms. After completing the procedures in this section, you will have a CICS TS platform, `GenerallInsuranceDev`, which is used as a single-region development platform for GENAPP in the following scenarios:

- ▶ Chapter 4, “Creating an application” on page 51
- ▶ Chapter 5, “Applying a policy” on page 97
- ▶ Chapter 6, “Packaging an application for multiversion deployment” on page 127

The following sections build on previous instructions described in Chapter 2, “GENAPP introduction” on page 15. You need either IBM CICS Explorer V5.2 or CICS Explorer V5.2 software development kit (SDK) software.

The single region development platform is a useful configuration for getting started with any new development. Any complexities, such as remote resource definitions or IBM z/OS Workload Manager (WLM) routing, can be overlooked at this stage, because all of the resources are installed locally.

Additionally, because the platform is still split into three separate region types all sharing the same single region, it encourages good development practices. For example, it promotes separating the presentation, business logic, and data access tiers of your application.

3.4.1 Defining the development topology

In a multi-region platform, each region type contains one or more regions. In this development platform, each region type contains the same single region, which a developer will install all resources into.

This development platform will adopt our existing infrastructure, meaning that the region types created by the platform map to existing CICSplex SM System Groups. For a test or production environment, these would have already been created as part of the work to assemble the environment for the application. In our development environment, an ad hoc platform must be created quickly for hosting our GENAPP application.

Below the created System Groups are their given simple names, such as GENATOR (the GENAPP presentation layer, also known as the terminal-owning region (TOR), system group). However, in a real development CICSplex, you will most likely give each set of System Groups region-specific or developer-specific names, such as DEV3TOR (Developer 3 TOR System Group).

If they are unique within the CICSplex and fit with business naming guidelines, the name of the System Group is unimportant. Regardless of the name chosen, the region type provides the necessary abstraction so that the underlying System Group name is inconsequential.

Perform the following steps to create the GENATOR, GENA00R, and GENAD0R System Groups using the CICSplex SM Administration perspective in CICS Explorer:

1. In the CICSPlex Explorer view, left-click the CICSPlex name that you want to work with. In this instance, select **GNAPPLEX** CICSPlex.
2. In the main menu, left-click the **Definitions** menu, and then select **System Group Definitions**. The System Group Definitions view displays.
3. Right-click in the System Group Definitions view and then select **New**, as shown in Figure 3-4. The New System Group wizard displays.

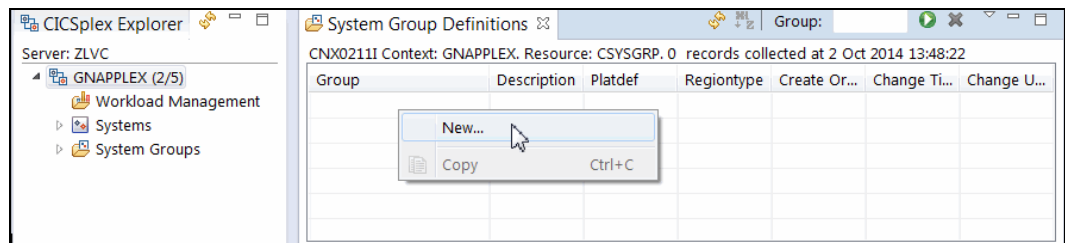


Figure 3-4 Using the System Group Definitions view to create a new System Group

4. In the New System Group wizard, validate that the CICSplex name pre-entered into the CICSplex field is the CICSplex that you want to work with. If not, go back to step 1 and re-select the correct CICSplex.
5. Enter GENATOR into the Group field.
6. Enter General insurance terminal owning regions into the Description field.

7. Clear the **Open editor** option. The New System Group wizard should appear as shown in Figure 3-5.

Figure 3-5 Completed create New System Group wizard for terminal owning regions

8. Click **Finish**.
9. Confirm that the System Group Definitions view now contains the new GENATOR System Group.
10. Return to step 3 and continue until the GENAPP application-owning region (AOR) System Group (GENAOR) and GENAPP data-owning region (DOR) System Group (GENADOR) have been created. The System Group Definitions view appears as shown in Figure 3-6.

Group	Description	Platdef	Regiontype	Create Origin	Change Time	Change User ID
GENAOR	General insurance application owning regions			N_A	02-Oct-2014 13:57:14	
GENADOR	General insurance data owning regions			N_A	02-Oct-2014 13:57:29	
GENATOR	General insurance terminal owning regions			N_A	02-Oct-2014 13:56:26	

Figure 3-6 The terminal, application, and data owning region groups are displayed

Now that the System Groups have been created, the next step is to define the single development region. If the System Definition for the development region has already been created, this part can be skipped.

Use the following steps to create the GENADEV1 System Definition using the CICSplex SM Administration perspective in CICS Explorer:

1. In the CICSplex Explorer view, left-click the CICSplex name that you want to work with. In this instance, select the **GNAPPLEX** CICSplex.
2. In the main menu, left-click the **Definitions** menu and then select **System Definitions**. The System Definitions view displays.

3. Right-click in the System Definitions view and then select **New**, as shown in Figure 3-7. The New System Definition wizard displays.

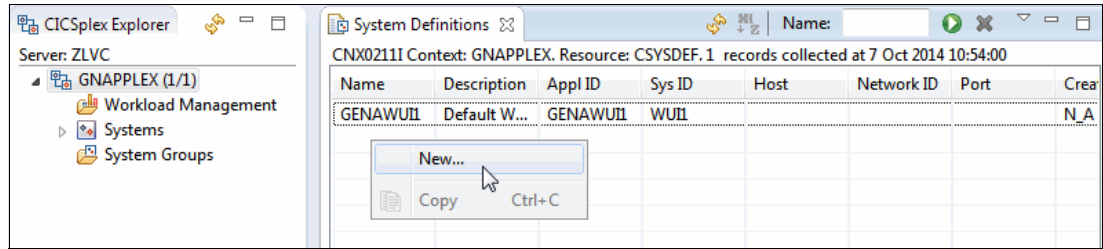


Figure 3-7 Using the System Definitions view to create a new System Definition

4. In the New System Definition wizard, validate that the CICSPlex name pre-entered into the CICSPlex field is the CICSPlex that you want to work with. If not, go back to step 1 and re-select the correct CICSPlex.
5. Enter GENADEV1 into the Name field.
6. Enter Development region into the Description field.
7. Enter GENADEV1 into the Appl ID field.
8. Enter DEV1 into the Sys ID field.
9. Ensure that the Open editor option is cleared. The New System Definition wizard should appear as shown Figure 3-8.

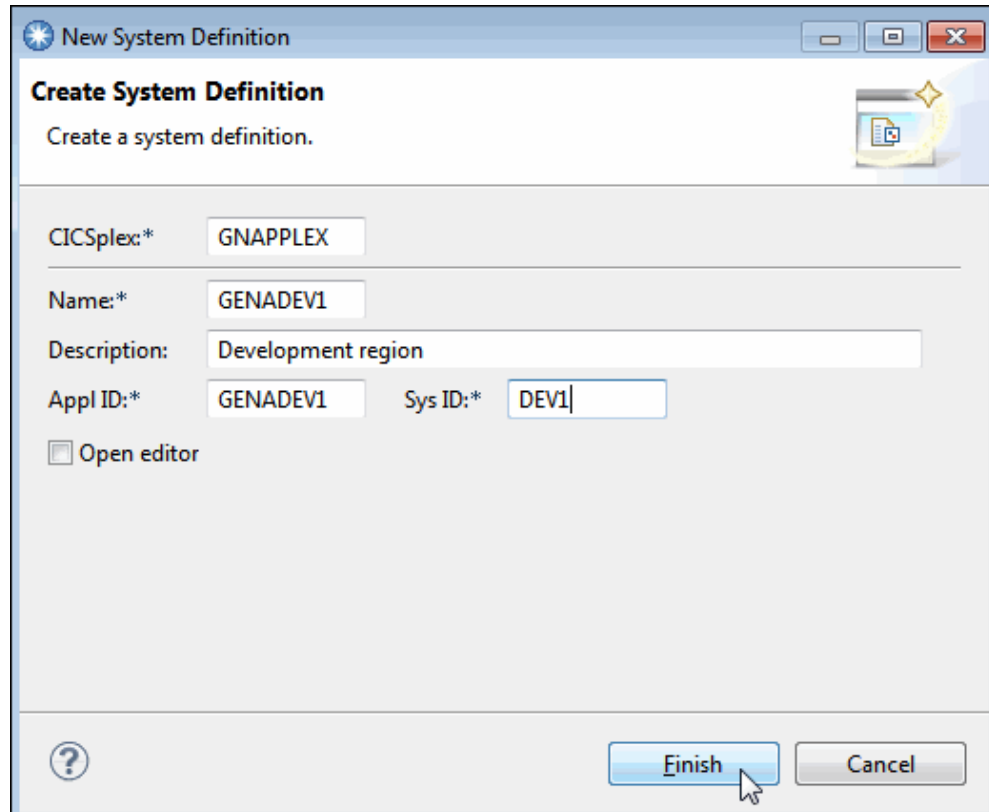


Figure 3-8 Create New System Group wizard

10. Click **Finish**.

11. Confirm that the System Definitions view now contains the new GENADEV1 System Definition.

All System Groups and the System Definition for the development region are now in place. The next step is to add the System Definition to each of the System Groups. This enables the platform region types to point all resources to be installed at the same development region.

Use the following steps to add the GENADEV1 System Definition to the GENATOR, GENAOR, and GENADOR System Groups using the CICSplex SM Administration perspective in CICS Explorer:

1. In the CICSplex Explorer view, left-click the CICSplex name that you want to work with. In this instance, select the **GNAPPLEX** CICSplex.
2. In the main menu, left-click the **Definitions** menu and select **System Definitions**. The System Definitions view displays.
3. Right-click the **GENADEV1** System Definition and select **Add to Group**, as shown in Figure 3-9. The Perform Operation wizard displays.

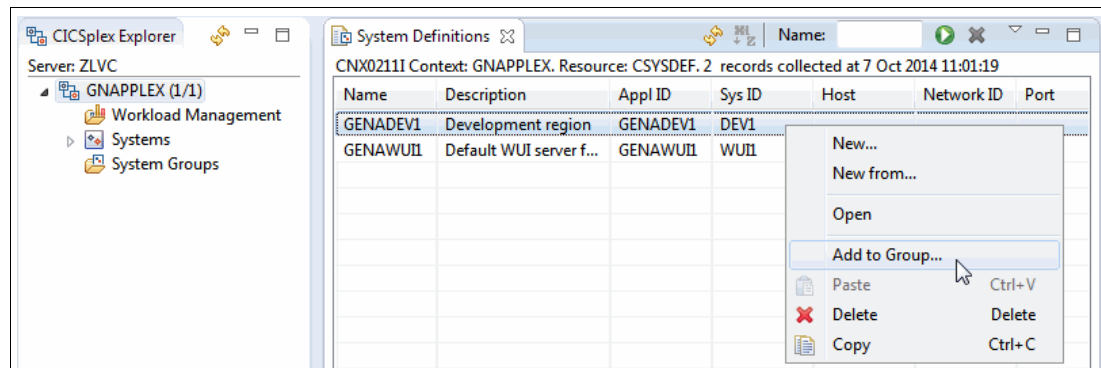



Figure 3-9 Using the System Definitions view to add a System Definition to a System Group

4. In the Perform Operation wizard, enter GENATOR into the Group Name field.
5. Click **OK**.
6. Repeat from step 3 until the GENADEV1 System Definition has been added to all three System Groups.

The GENATOR, GENAOR, and GENADOR System Groups should each now contain a single System Definition, GENADEV1.

Use the following steps to confirm that the GENADEV1 System Definition was added to each of the GENATOR, GENAOR, and GENADOR System Groups using the CICSplex SM Administration perspective in CICS Explorer:

1. Refresh the CICSplex Explorer view by pressing the refresh () button beside the CICSplex Explorer title.
2. In the CICSplex Explorer view, left-click the twistie left of the CICSplex name that you want to work with. In this instance, open the twistie left of the GNAPPLEX CICSplex.

This will expand the CICSplex entry to show the System Groups sub-folder, as shown in Figure 3-10.

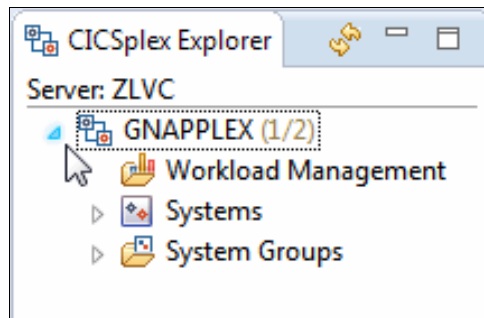


Figure 3-10 Using the CICSplex Explorer view to show information about a specific CICSplex

3. Left-click to open the twistie left of the System Groups sub-folder. This will expand to show a further list of sub-folders with the names of the newly created GENATOR, GENAOR and GENADOR System Groups.
4. Expand each of the System Groups by opening the twistie left of it.
5. If the System Definition GENADEV1 was added correctly, it should appear as a subitem under each group, as shown in Figure 3-11.

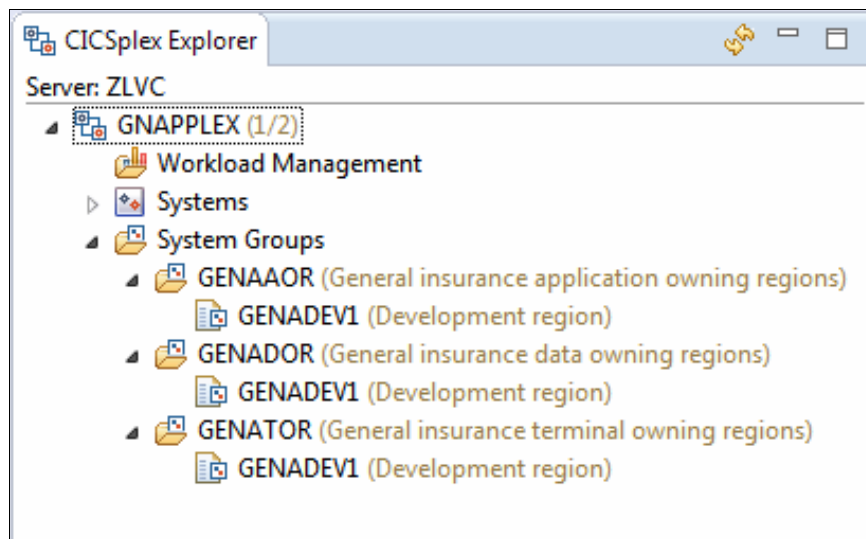


Figure 3-11 Using the CICSplex Explorer view to show System Definitions in System Groups

The development GENADEV1 System Definition has been correctly created and added to the System Groups. In the following section, these System Groups will be mapped onto platform region types. Application or platform resources installed into these region types will now install all resources into the development region GENADEV1, regardless of the region type chosen.

3.4.2 Defining the platform

Perform the following steps to create the GeneralInsuranceDev CICS TS platform project using the CICS Cloud perspective in CICS Explorer:

1. Right-click in the Project Explorer view and then select **New** → **Project**, as shown in Figure 3-12. The New Project wizard displays.

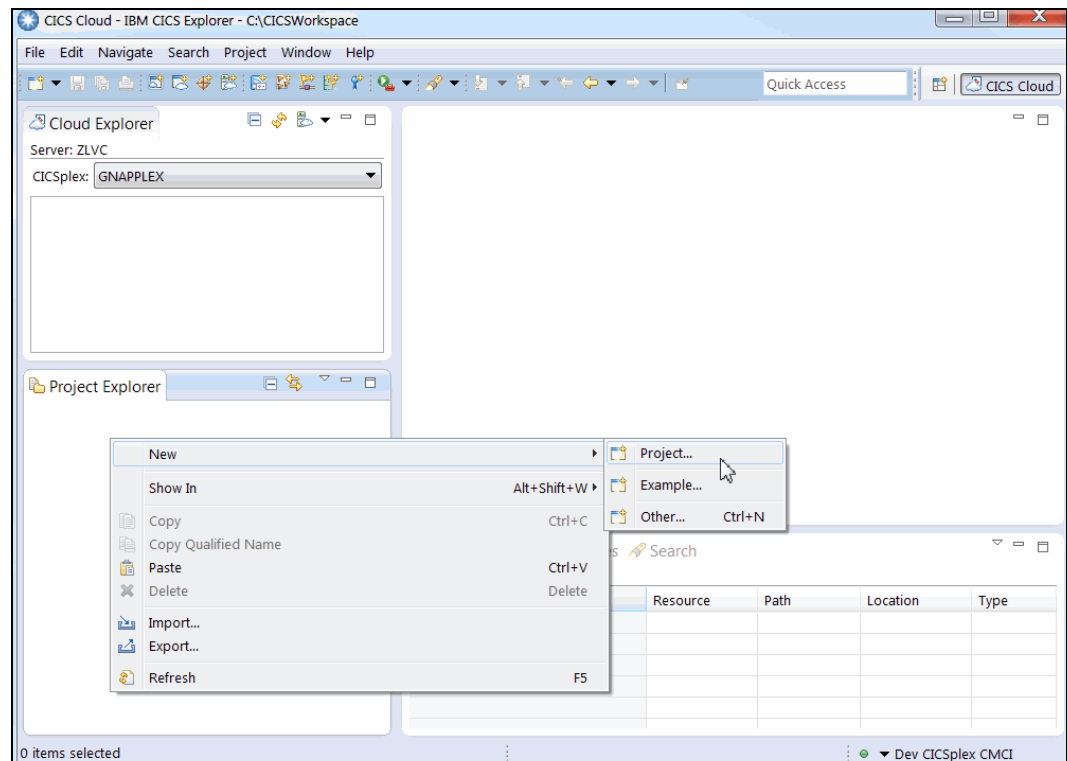


Figure 3-12 Using the Project Explorer view to create a new Project

2. In the New Projects wizard expand CICS Resources, click **CICS Platform Project** (see Figure 3-13), then click **Next**. The CICS Platform Project wizard displays.

Alternatively, to create a project for a CICS TS platform, click the files plus cloud (📁☁️) icon in the tool bar to launch the CICS Platform Project wizard.

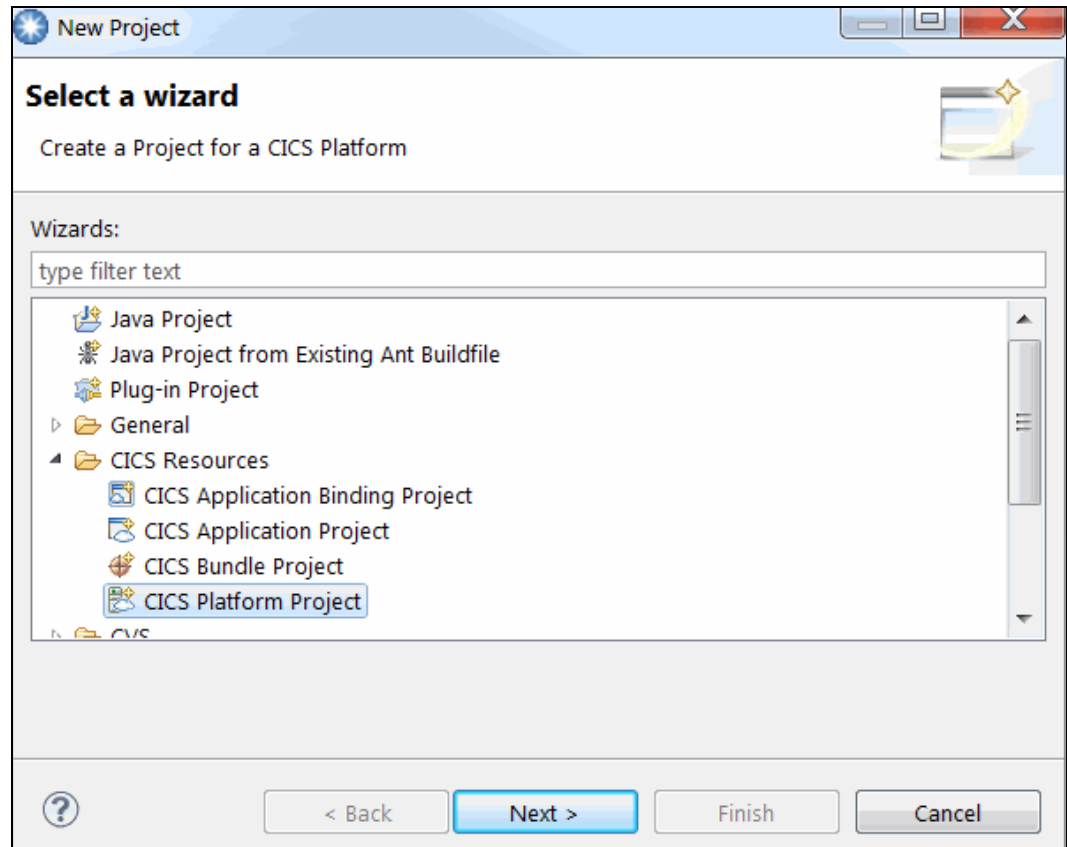
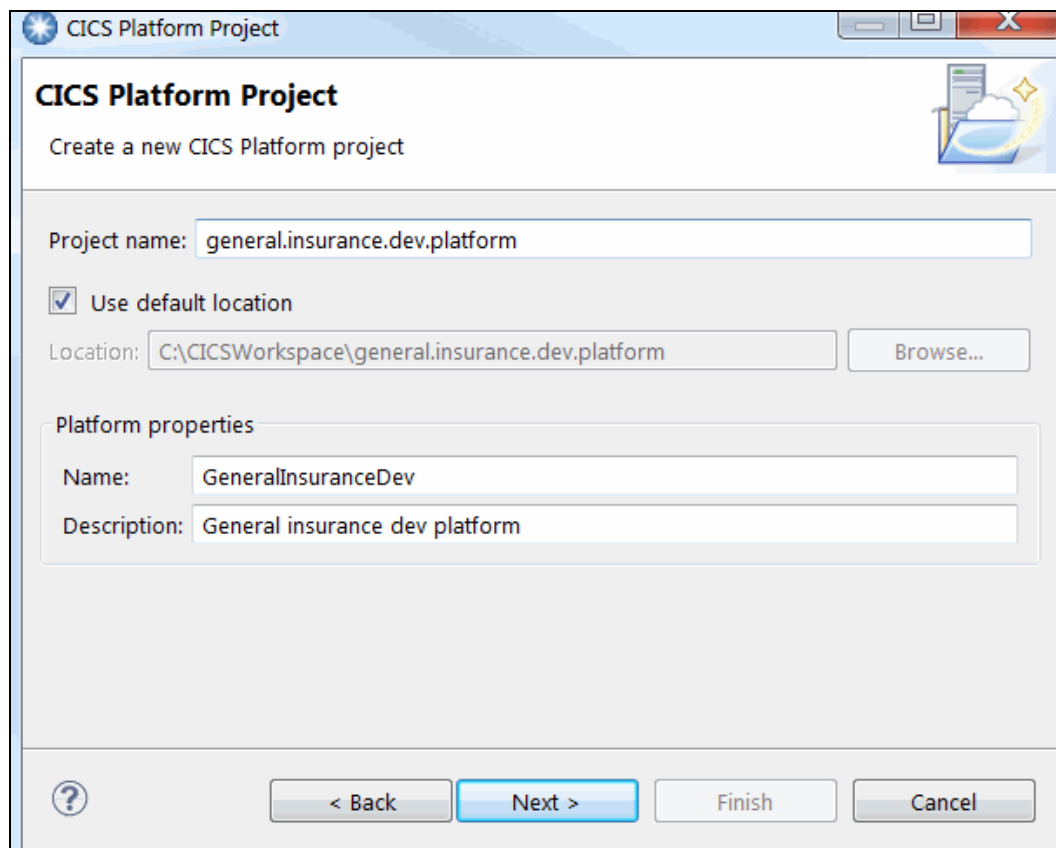


Figure 3-13 Selecting a CICS Platform Project in the New Project wizard

3. Enter `general.insurance.dev.platform` in the Project name field.
4. Enter `GeneralInsuranceDev` in the Name field.

5. Enter General insurance dev platform in the Description field. The New Platform project wizard should look like Figure 3-14.



The screenshot shows a Windows-style dialog box titled "CICS Platform Project". Below the title bar, the text "CICS Platform Project" is displayed in a bold font, followed by "Create a new CICS Platform project" in a smaller font. To the right of the text is a small icon of a folder with a cloud and a star. The main area of the dialog contains several input fields and a checkbox. The "Project name:" field is filled with "general.insurance.dev.platform". Below it, the "Use default location" checkbox is checked. The "Location:" field is filled with "C:\CICSWorkspace\general.insurance.dev.platform", and there is a "Browse..." button to its right. Below these fields is a section titled "Platform properties" which contains two more input fields: "Name:" filled with "GeneralInsuranceDev" and "Description:" filled with "General insurance dev platform". At the bottom of the dialog, there is a row of four buttons: a help button (question mark icon), "< Back", "Next >", and "Cancel". The "Next >" button is highlighted with a blue border.

Figure 3-14 Create New CICS Platform Project wizard

6. Click **Next**.
The wizard now enables you to define the region types that will compose this CICS TS platform.
Next, you define the ApplicationServices region type.
7. Click **Add**. The Add a Region Type dialog shows.
8. Enter ApplicationServices into the Name field.
9. Select the **Adopt an existing system group** radio button.
If required, connect to your CICS TS environment's CICS Management Client Interface (CMCI) connection.
10. Select **GNAPPLEX** in the CICSplex drop-down box.

11. Select **GENAAOR** in the CICS system group drop-down box. The dialog should look as shown in Figure 3-15.

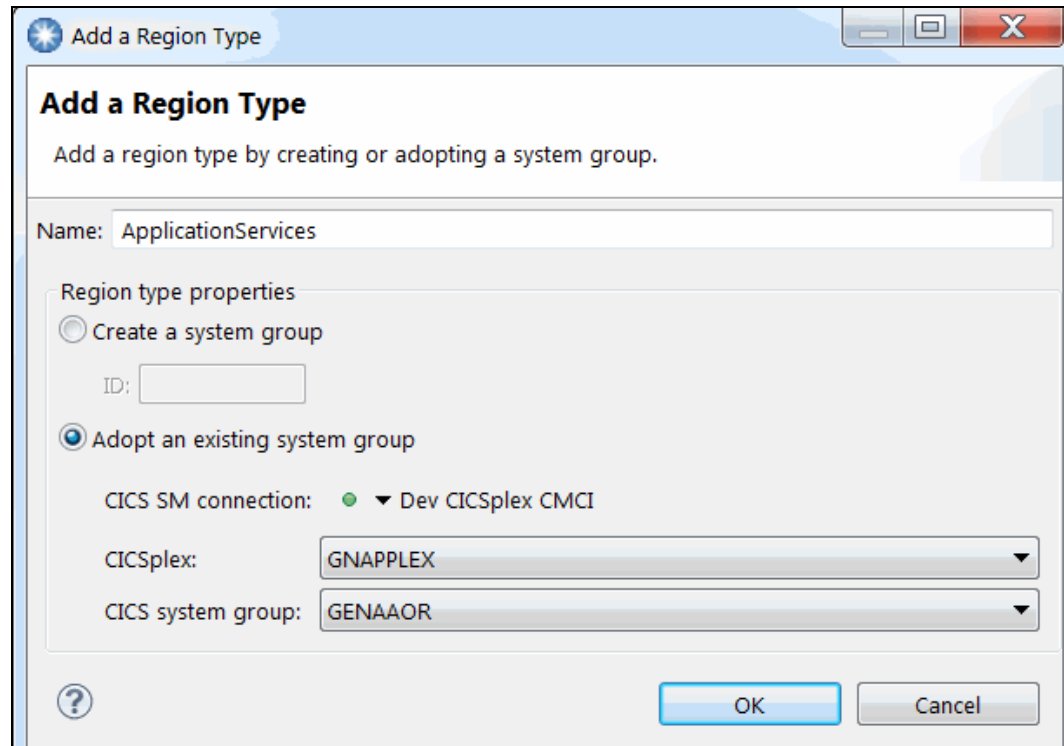


Figure 3-15 Adding the ApplicationServices region type to the platform

12. Click **OK**. The ApplicationServices region type has been defined in the CICS Platform Project.
Next, define the DataAccess region type.
13. Click **Add**.
14. Enter DataAccess into the Name field.
15. Select the **Adopt an existing system group** radio button.
16. Select **GNAPPLEX** in the CICSplex drop-down box.

17. Select **GENADOR** in the CICS system group drop-down box. The dialog should look like Figure 3-16.

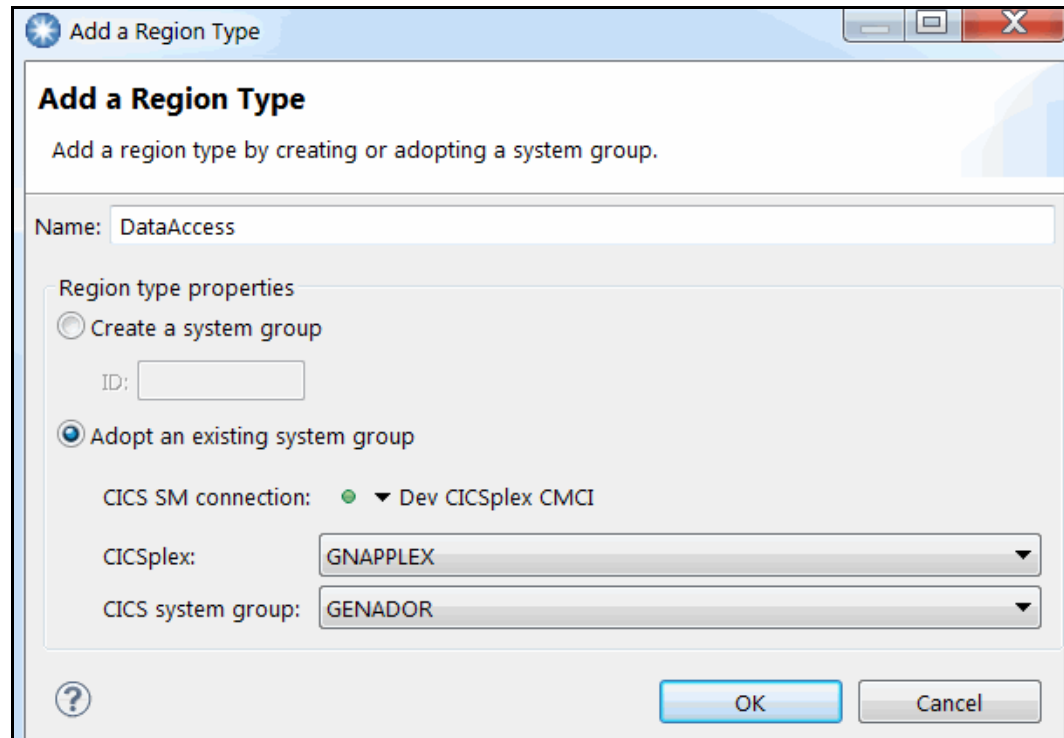


Figure 3-16 Adding the DataAccess region type to the platform

18. Click **OK**. The DataAccess region type has been defined in the CICS Platform Project.
- Next, define the Presentation region type.
19. Click **Add**.
20. Enter Presentation into the Name field.
21. Select **Adopt an existing system group**.
22. Select **GNAPPLEX** in the CICSplex drop-down box.

23. Select **GENATOR** in the CICS system group drop-down box. The dialog should look like Figure 3-17.

Add a Region Type

Add a region type by creating or adopting a system group.

Name:

Region type properties

☐ Create a system group

ID:

☒ Adopt an existing system group

CICS SM connection: ● ▼ Dev CICSplex CMCI

CICSplex:

CICS system group:

Figure 3-17 Adding the Presentation region type to the platform

24. Click **OK**. The Presentation region type has been defined in the CICS Platform Project.

The CICS Platform Project wizard should look like Figure 3-18.

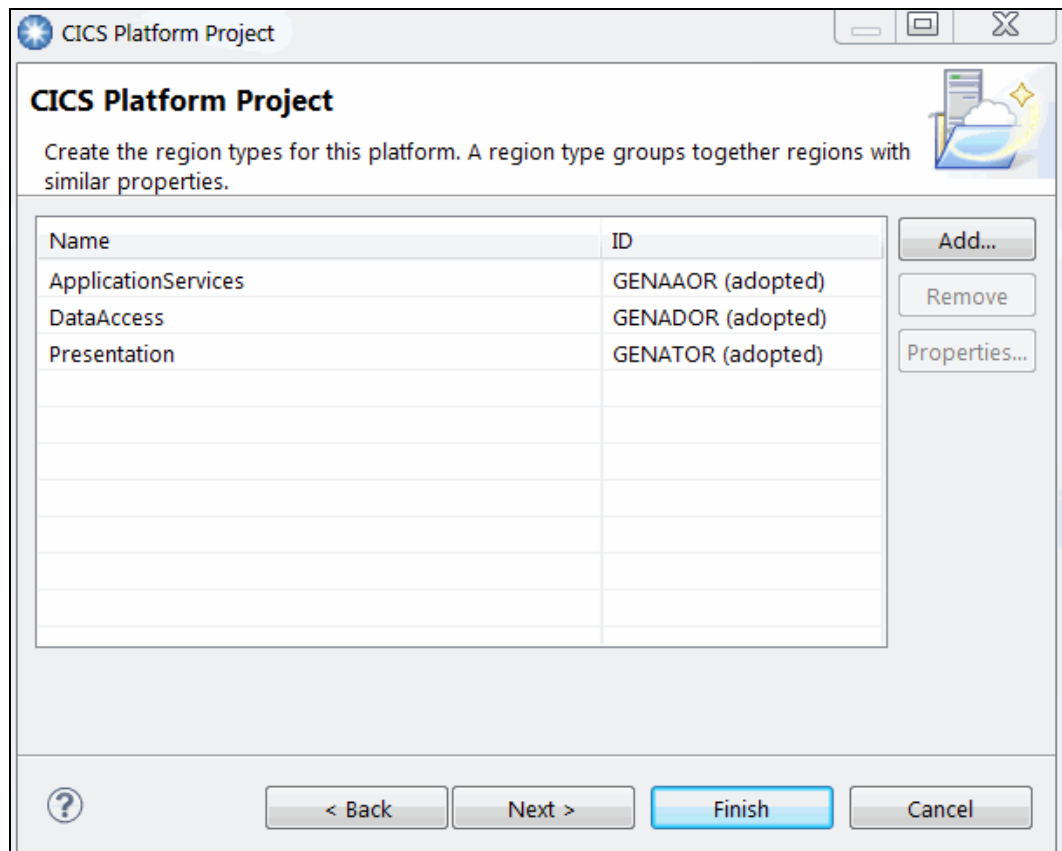


Figure 3-18 Platform GeneralInsuranceDev region types

25. Click **Next**.

The wizard shows the CICS Bundle Projects that have been found in the workspace that can be included into the CICS TS platform.

No CICS bundles are required in this CICS TS platform, as shown in Figure 3-19.

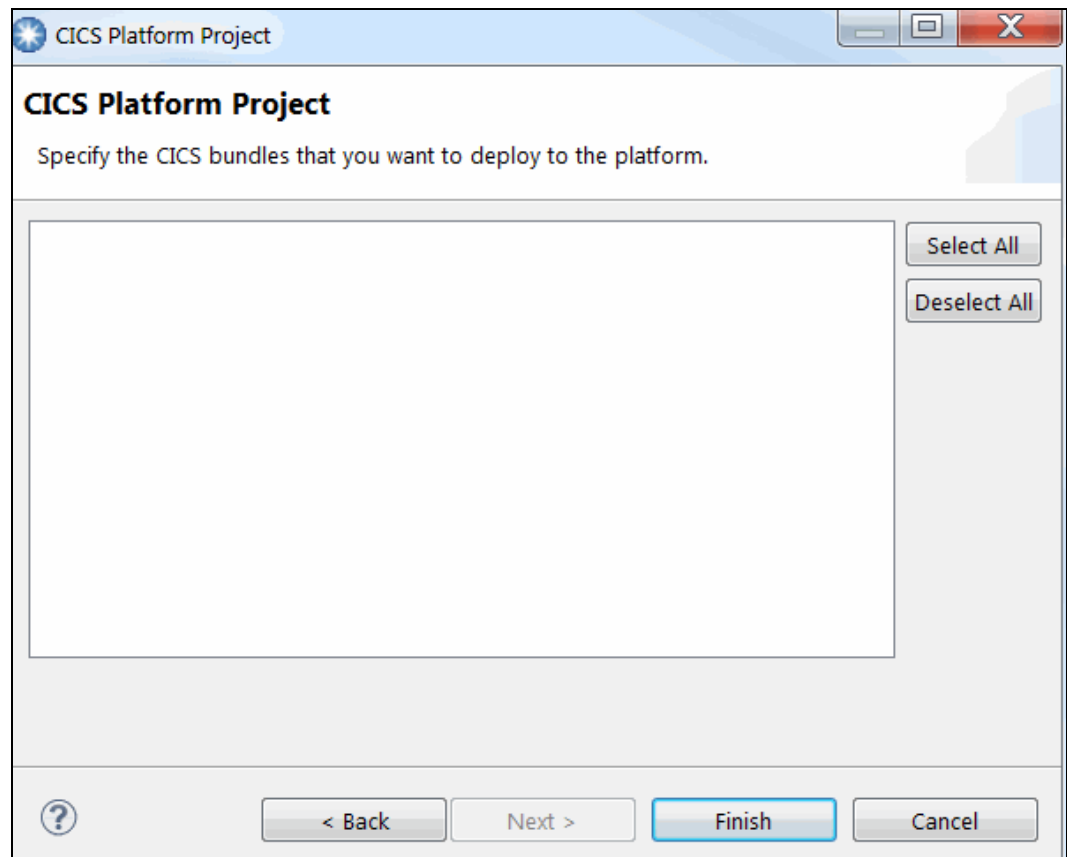


Figure 3-19 List of bundles in the Explorer workspace to include in the platform

26. Click **Finish**. The platform project is created.

The Project Explorer view should look similar to Figure 3-20.

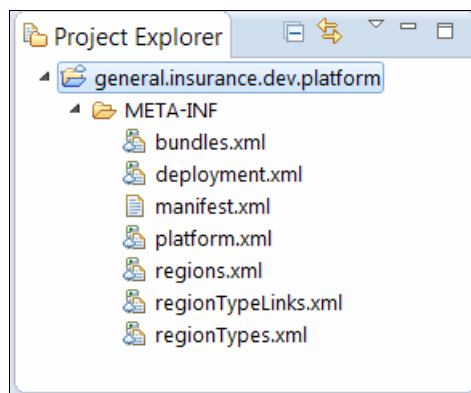


Figure 3-20 Project Explorer view after creating the GeneralInsuranceDev CICS Platform Project

The platform editor for the GeneralInsuranceDev platform launches, and should look similar to Figure 3-21.

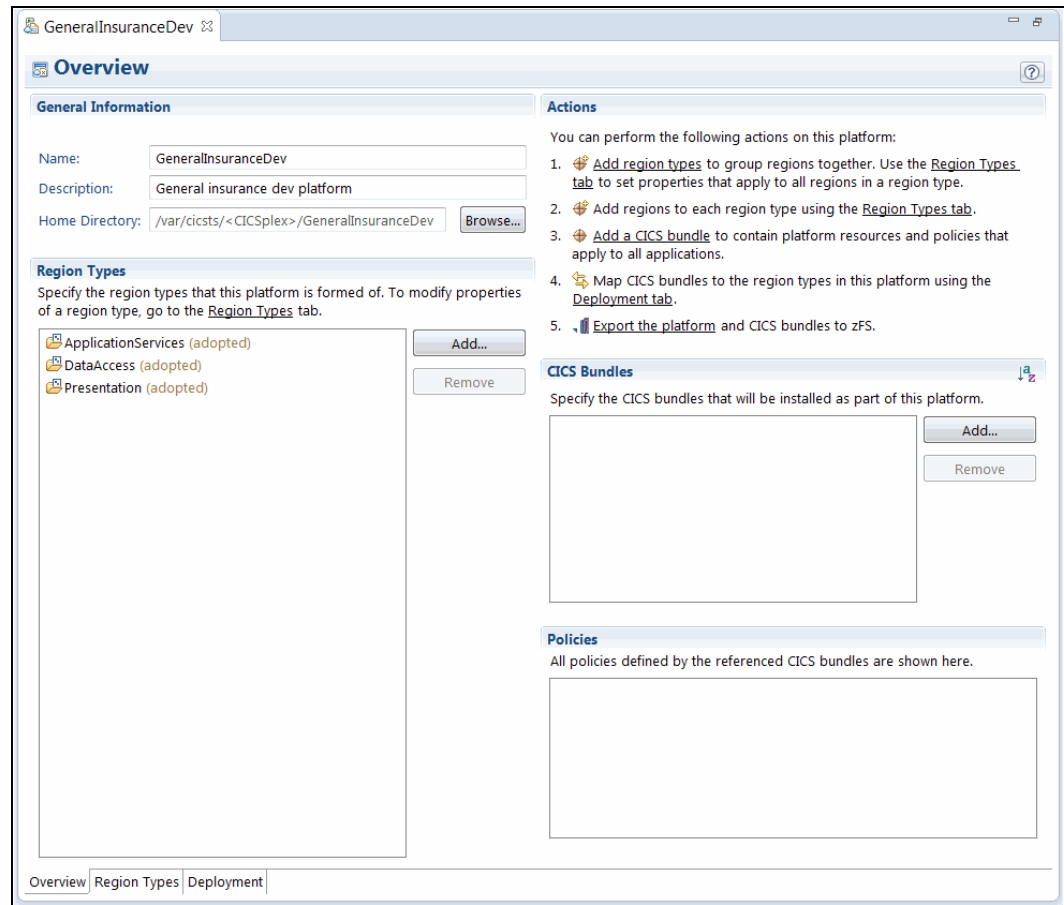


Figure 3-21 GeneralInsuranceDev platform in the platform editor

A CICS Platform Project has now been created. The project defines the name of the CICS platform when it is installed, and it also defines the region types that will be created along with the CICS platform. You have now created the CICS Platform Project for the General Insurance dev platform that represents a development environment ready for deployment to your CICSplex. At this stage, your project only exists locally in the CICS Explorer workspace.

3.4.3 Deploy the platform

This section describes how to deploy your new CICS Platform Project from the local workspace into IBM z/OS Distributed File Service zSeries File System (zFS). The CICS TS platform will then be installed into the CICSplex.

Perform the following steps to deploy the GeneralInsuranceDev CICS TS platform using the platform editor:

1. If the platform editor for the `general.insurance.dev.platform` project is not open, locate the platform files to start the platform editor.

In the Project Explorer view, expand the `general.insurance.dev.platform` project and expand the `META-INF` subdirectory, then double-click the `platform.xml` file to start the platform editor. The platform editor for the GeneralInsuranceDev platform starts, and should look similar to Figure 3-21.

2. On the Overview tab in the platform editor for the GeneralInsuranceDev platform, click **Export the platform** in the Actions section. The Export Platform wizard displays.
If required, connect to your CICS TS environment's CMCI connection.
3. Select **GNAPPLEX** from the CICSplex selection box.
4. Select the **Create CICS Platform Definition after export finishes** check box.

Figure 3-22 shows the completed first page of the Export Platform wizard for the development platform.

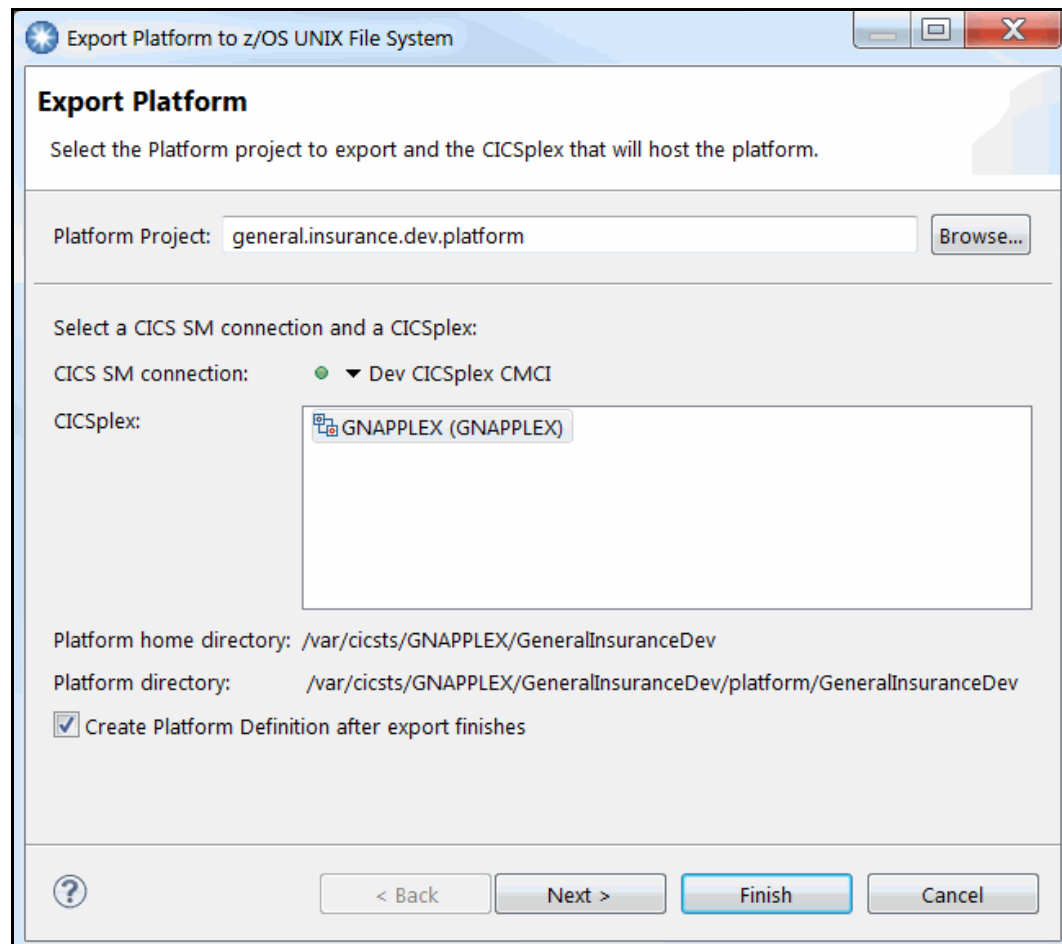


Figure 3-22 Export Platform wizard for development platform GeneralInsuranceDev

5. Click **Next**.

6. Verify that the correct File Transfer Protocol (FTP) connection details are connected. See Figure 3-23.

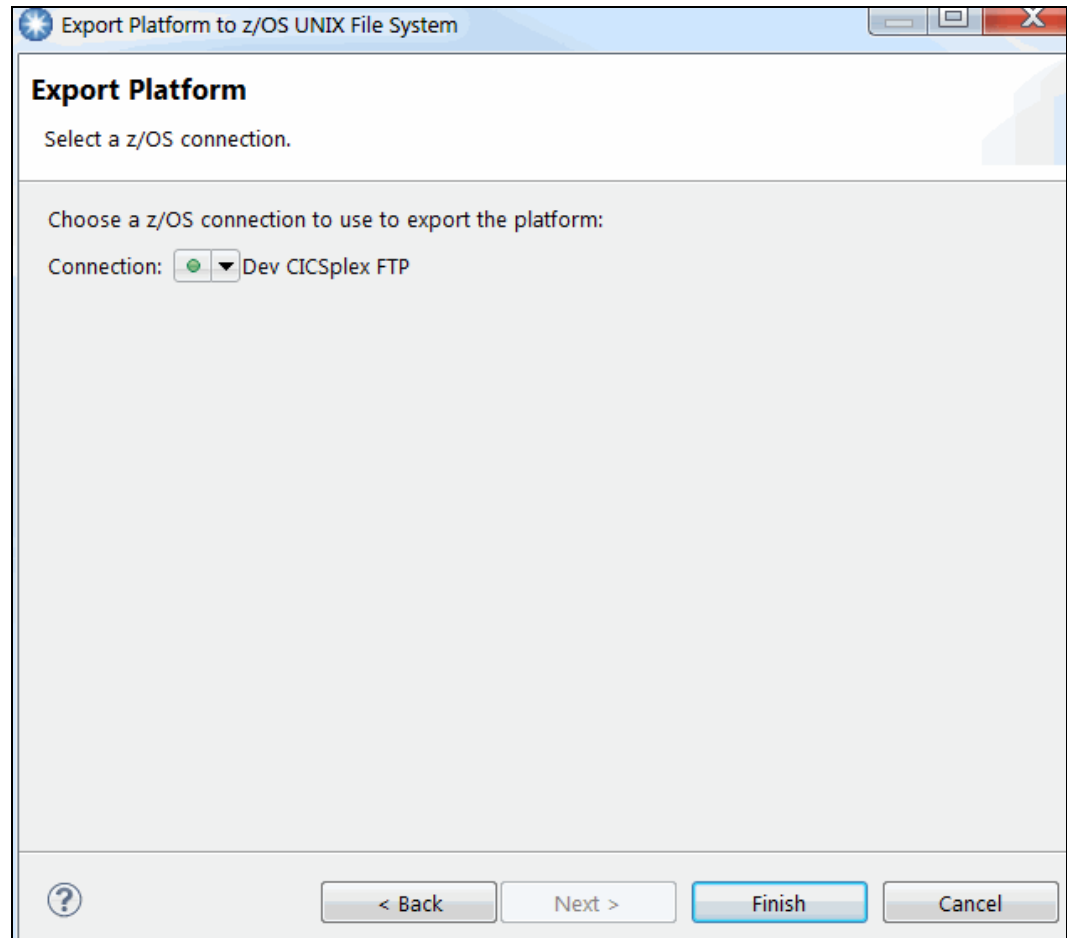


Figure 3-23 An established FTP connection in the platform export wizard

7. Click **Finish**.

The CICS Platform Project begins to export to the z/OS UNIX file system. The CICS Explorer ensures that the correct directory structure is created so that the CICS Platform Project is ready to be used. After the export finishes, the New Platform Definition wizard displays. The CICS Explorer will automatically complete the detail obtained through the export process.

8. Enter GNAPDEV in the Name field.

9. Enter General insurance dev platform in the Description field. Figure 3-24 shows the completed New Platform Definition wizard for the development platform.

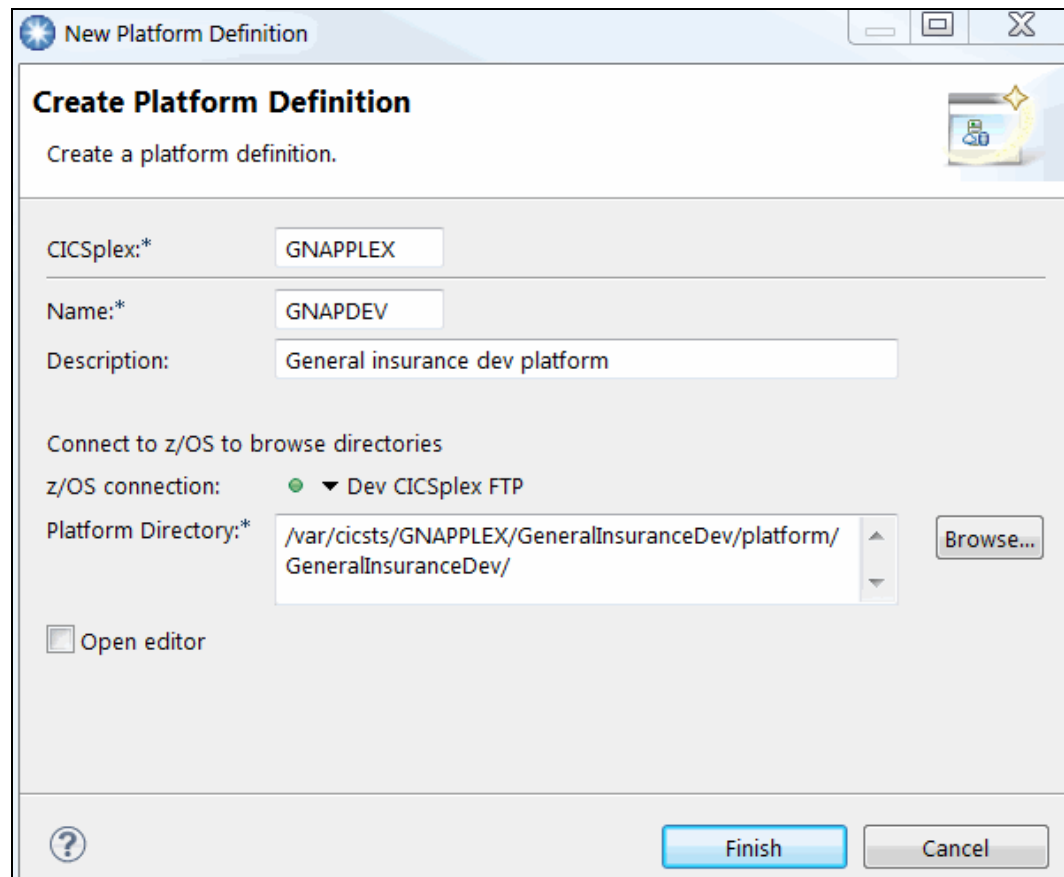


Figure 3-24 New Platform Definition wizard for development platform GeneralInsuranceDev

10. Click **Finish**. The CICS Platform Definition has now been created. The CICS TS cloud view should appear as in Figure 3-25.

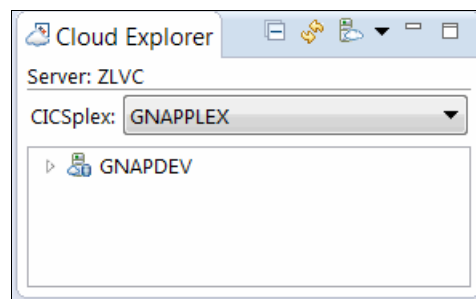


Figure 3-25 Cloud Explorer view after creating the CICS Platform Definition GNAPDEV

11. Right-click **GNAPDEV** in the Cloud Explorer view.

12. Select **Install**, as shown in Figure 3-26. The Perform Operation dialog displays.

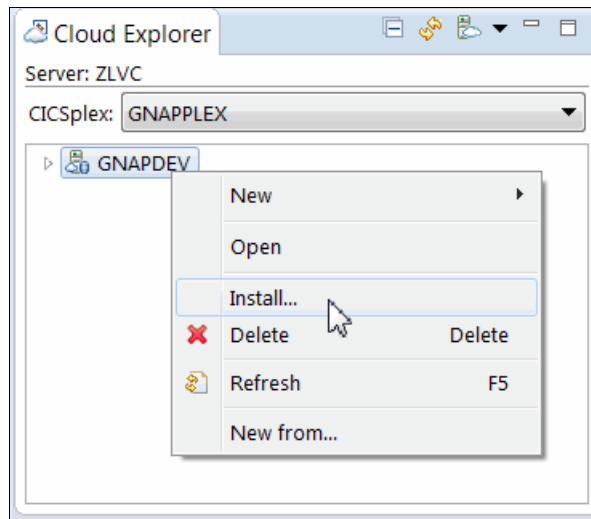


Figure 3-26 Menu to install a platform using the Cloud Explorer View

13. Click **OK** in the Perform Operation dialog, as shown in Figure 3-27. The GeneralInsuranceDev CICS TS platform begins to install.

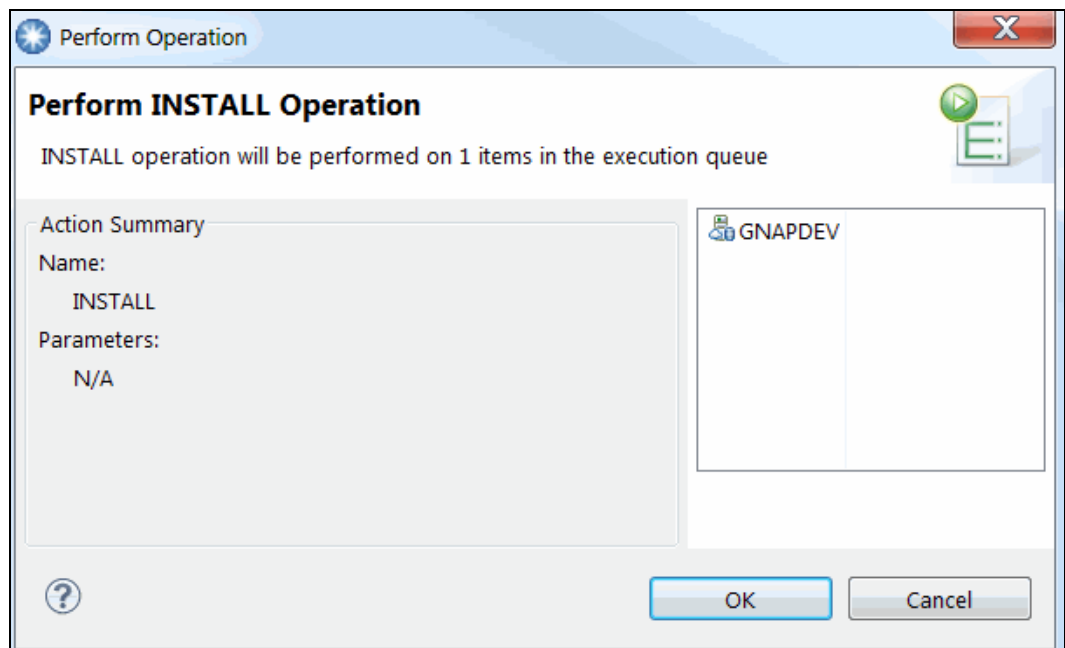


Figure 3-27 Perform operation dialog to install a platform

14. Confirm the contents in the Cloud Explorer view, as shown in Figure 3-28.

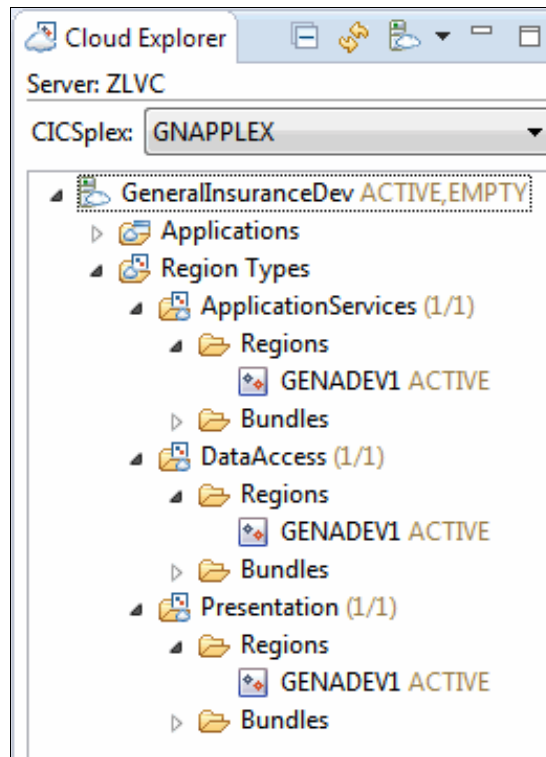


Figure 3-28 Cloud Explorer view after installing the platform

You have now successfully installed the GeneralInsuranceDev platform into your CICS TS environment. The GeneralInsuranceDev CICS TS platform has now been created, deployed, and installed.

3.4.4 Multi-region platform for test or production

9.5, “Promoting GeneralInsuranceCustomer from dev to test” on page 203 describes the technique for promoting your application from this single region development platform into a multi-region platform suitable for test or production environments. The creation of this GeneralInsuranceTest platform is easily accomplished using the same technique described previously.

The only difference is that you should replace all references to “test” with “dev”. In a multi-region platform, the system groups that you adopt each contain one or more regions, rather than the same single region as is the case here.

3.5 Outcome

In this chapter, you have deployed and installed a CICS TS platform (GeneralInsuranceDev) on top of an existing CICSplex topology designed for an existing active application (GENAPP), as described in Chapter 4, “Creating an application” on page 51. In this instance, the topology was a single region. By using the flexibility of platforms, a development environment was created that simplifies creating resource definition while still providing developers with a tiered topology.

From this simple platform, applications can now be deployed and managed. The platform can be extended to provide more services to those applications, or to apply policies that affect all applications installed into this platform. Additionally, for a new CICSplex, the platform itself can define and create the region and system group definitions for you. This enables your complete topology to follow the platform lifecycle.

When test *and* production environments are required, the same platform deployment techniques can be used to deploy platforms to multi-region environments with only minor changes.

For further information about platform design, visit the IBM Knowledge Center on the following website:

http://www.ibm.com/support/knowledgecenter/SSGMCP_5.2.0/com.ibm.cics.ts.doc/eyua7/topics/settingup_platform.html?lang=en



Creating an application

This chapter introduces the IBM Customer Information Control System (CICS) Transaction Server (CICS TS) application resource, highlighting how it can provide business value. We then demonstrate how to create a CICS TS application resource for the General Insurance Application (GENAPP), which was introduced in Chapter 2, “GENAPP introduction” on page 15, and is used for all of the walkthroughs in this book.

The following list includes reasons why a user would create a CICS TS application:

- ▶ Easily determine the status of application resources and dependencies.
- ▶ Measure resource usage. See Chapter 7, “Measurement by application” on page 175.
- ▶ Enable threshold policies to be scoped to an application. See Chapter 5, “Applying a policy” on page 97.
- ▶ Enable multiple versions of an application to be made available on the same CICS TS platform at the same time. See Chapter 6, “Packaging an application for multiversion deployment” on page 127.

This chapter contains the following topics:

- ▶ 4.1, “The CICS TS application” on page 52
- ▶ 4.2, “The components of a CICS TS application” on page 53
- ▶ 4.3, “Developing a CICS TS application” on page 55
- ▶ 4.4, “Example” on page 56
- ▶ 4.5, “The outcome” on page 95

4.1 The CICS TS application

In CICS TS V5.1, CICS TS applications provide business value by simplifying application lifecycle management, and enabling performance measurements and threshold policies to be applied at the application level. In CICS TS V5.2, the value of CICS TS applications was further enhanced by enabling multiple versions of the same CICS TS application to be installed and running at the same time, on the same CICS TS regions. This enables rollout of new application versions without any downtime.

The following list describes the benefits of CICS TS applications in more detail:

- ▶ You can start to measure, at the application level, the cost of your existing applications. By creating a CICS TS application resource that represents an existing application, and enabling CICS TS performance monitoring, you can report on overall application usage for all tasks across all regions where the application is deployed.

This is described in further detail in Chapter 7, “Measurement by application” on page 175.

- ▶ By applying threshold policies, you can protect your CICS TS regions against problematic applications using too many resources. For example, you could issue a message if an update to a CICS TS application resulted in it making too many Structured Query Language (SQL) calls, or using too much processor time.

This is described in further detail in Chapter 8, “Managing by policy” on page 181.

- ▶ You can use a CICS TS application as a single point of reference that indicates whether all of the resources required by an existing CICS TS application are enabled. If any resource gets disabled, the CICS TS application reflects that state. If all resources are enabled, the CICS TS application shows an enabled state.

This is demonstrated later in this chapter.

- ▶ You can simplify application management by using CICS TS applications to deploy and manage the CICS TS resources that your application is composed of, as a single entity, throughout the lifecycle, and across every CICS TS region in which your application runs. By moving the definition of resources from the CICS TS system definition (CSD) data set or IBM CICSplex System Manager (CICSplex SM) Business Application Services (BAS), and into the CICS TS application, they are bundled with the application for its lifecycle.

When you install the application, the resources are installed across all CICS TS regions that the application uses. When you enable the application, the resources are enabled, and so on.

This is described in further detail in Chapter 6, “Packaging an application for multiversion deployment” on page 127.

- ▶ You can deploy multiple versions of the same CICS TS application to the same CICS TS regions, which enables rollout of new application versions without any downtime. CICS TS applications can contain private program and library resources, so each application version can run different versions of programs even when the CICS TS program resources have the same names.

This is described in further detail in Chapter 6, “Packaging an application for multiversion deployment” on page 127.

- ▶ You can deploy different applications to the same CICS TS regions, even if they have clashing program names. This can be especially useful for vendor products over which you have no control of the naming conventions used. CICS TS applications can contain private program and library resources, so each application can run different copies of programs even when the CICS TS program resources have the same names.

- ▶ CICS TS programs can be isolated by making them private to a CICS TS application. By making the programs private, you can ensure that other applications can only use the CICS TS programs that you want them to use. This in turn can make it easier to manage application interdependencies.
- ▶ The CICS TS application, after it is created, can remain unchanged as it is promoted from development, through to test, and into production. Keeping the CICS TS application unchanged provides a greater level of confidence in the promotion of the application, and decreases the risk of errors being introduced during the promotion process.

This is described further in Chapter 9, “Application lifecycle management” on page 193.

4.1.1 Gaining value for existing CICS TS applications

In this chapter, we examine how you can start to gain value from CICS TS applications, by creating a CICS TS application resource that represents an existing CICS TS application. The resources for the existing application are defined on the CICS TS CSD, and this remains unchanged.

Perform the steps described in this chapter to complete the following tasks:

- ▶ Create a CICS TS application resource that enables you to measure the application usage.
- ▶ Apply policies to protect against errors in the application.
- ▶ Use the CICS TS application as a single point of reference for the overall status of all of the resources that make up the application.

4.2 The components of a CICS TS application

A deployed CICS TS application consists of the following components:

- ▶ A CICS TS application that has been exported to IBM z/OS Distributed File Service zSeries File System (zFS)
- ▶ One or more CICS bundles that have been exported to zFS
- ▶ A CICS application binding that has been exported to zFS
- ▶ An APPLDEF resource definition
- ▶ One or more application entry points
- ▶ Optional dependencies declared on resources external to the application
- ▶ Optional threshold policies that ensure that the application runs within defined limits

Figure 4-1 shows how these components fit together.

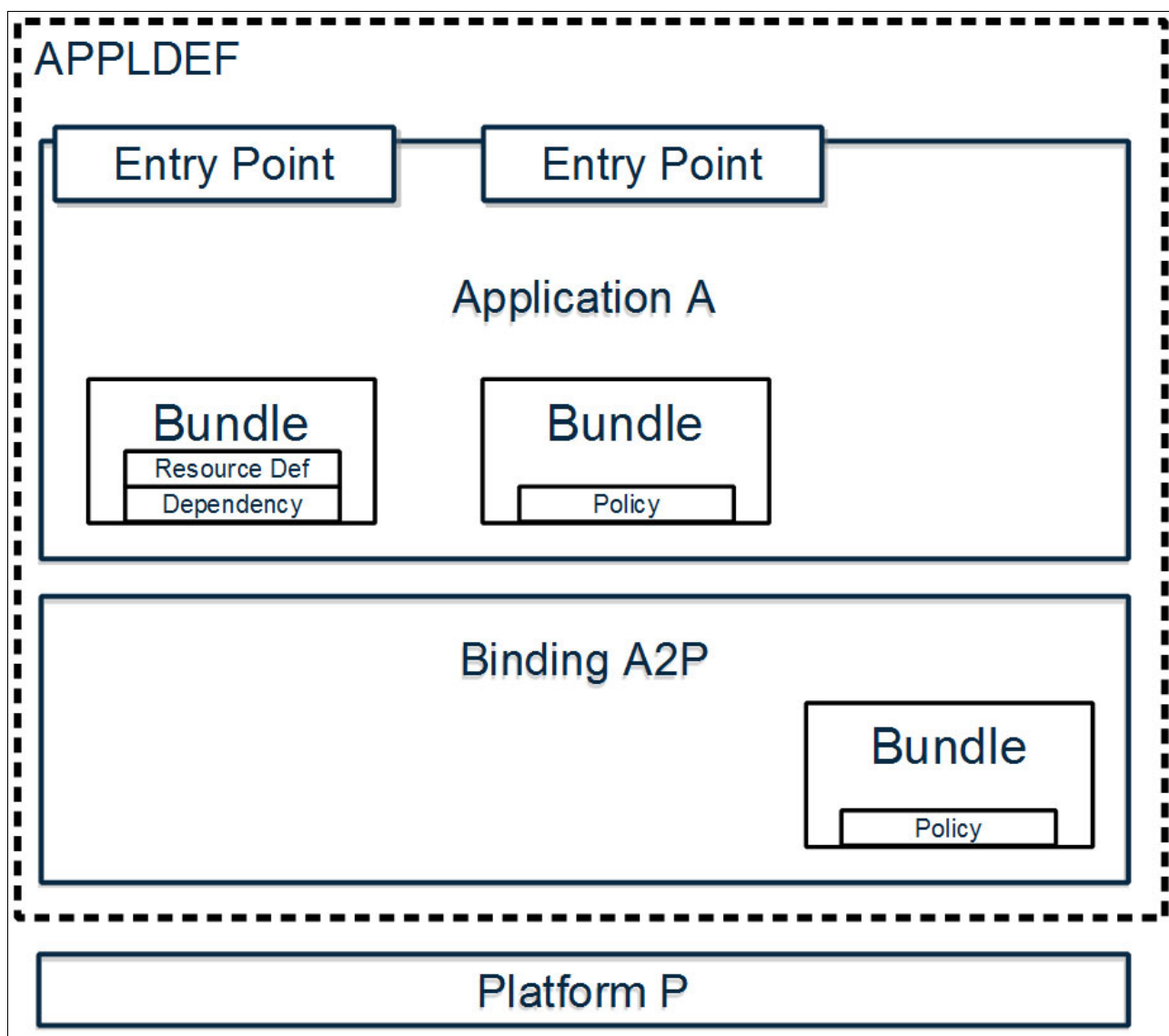


Figure 4-1 How the components of a CICS TS application fit together

The CICS TS application artifacts that have been exported to zFS are read by CICS TS during install of an APPLDEF resource definition. They provide the necessary information to enable the creation of the application resource.

The CICS TS application contains one or more CICS bundles. A CICS bundle is a container for a set of CICS TS resources. When a CICS bundle is installed into a CICS TS region, the set of resources that it contains also get installed.

A CICS TS application might span several different types of CICS TS regions. For example, the application might process requests in a web-owning region (WOR) and then run business logic in an application-owning region (AOR). If this is the case, the CICS TS application can contain a CICS bundle of resources for the WOR, and a second CICS bundle of resources for the AOR. The CICS bundles must be available on zFS at application install time.

A CICS TS application is deployed into a platform. The application should remain unchanged when it is deployed into different platforms (for example, a development platform and a test platform). To enable this, a *CICS application binding* is used. The application binding maps the needs of the application to the specific platform into which it is to be deployed. The application binding can add more CICS bundles to the application.

The *APPLDEF resource definition* tells CICS TS where to find the application and application binding on zFS. On installation of the APPLDEF, the application is installed.

Application entry point declarations in your CICS TS application identify when tasks running in CICS TS are entering into your application. As a task passes through an application entry point, *application context data* is associated with the task. The application context data is propagated forwards to other CICS TS regions that the task uses. The application context data can be made available in monitoring records, to provide a way of measuring how much system resources an application is using.

Threshold policies use a task's application context to identify when a task is in an application, and therefore when to apply policies associated with that application. CICS TS V5.1 enables a CICS TS PROGRAM resource to be set as an entry point. Therefore, when a task runs a CICS TS PROGRAM identified as an entry point, the application context is associated with the task.

In CICS TS V5.2, CICS TS URIMAP resources were added as an extra type of entry point. When an inbound request is mapped by a URIMAP identified as an entry point, the application context is associated with the task processing the inbound request.

Optional dependencies can be expressed within CICS bundles that are part of the CICS TS application. A dependency is checked by CICS TS at installation time and, if it is not available, the application will not enable. For example, if an application has a requirement on a particular CICS TS file being available, a dependency on the CICS TS file could be created. If the file is not installed, the application will not enable.

Optional threshold policies can be defined within CICS bundles that are a part of the CICS TS application. These policies can enforce that the application meets particular requirements. For example, it might need to complete within a specified amount of time. Threshold policies are discussed more in Chapter 5, "Applying a policy" on page 97.

4.3 Developing a CICS TS application

Development of the CICS TS application is done using CICS Explorer. Projects are created in CICS Explorer that represent components of the CICS TS application. These projects can be treated like program source code and shared using a source code management system.

The following list describes the different projects:

- ▶ Application Project
- ▶ Bundle Project
- ▶ Platform Project
- ▶ CICS Application Binding Project

An **Application Project** describes the application itself, including its name and version. The application project also describes the CICS bundles that make up an application. When the application is ready for testing, it is exported to zFS, where it is used by CICS TS at run time.

A **Bundle Project** represents a CICS bundle resource. In it, various CICS TS resources can be defined, application entry points can be created, and dependencies can be expressed on other CICS TS resources.

A **Platform Project** describes a CICS TS platform and the region types that it consists of. Chapter 3, “Creating a platform” on page 25 demonstrated the creation of a Platform project. The platform project does not need to be changed during creation of a CICS TS application, but it is required within the CICS Explorer environment for the creation of the application binding that maps that application to the platform.

A *CICS Application Binding Project* describes which CICS bundles within an Application map to particular region types. It can also be used to specify additional CICS bundles that need to be installed during installation of the application. For example, a resource needed by the application that requires different attributes in a test platform and a production platform, could be defined in the application binding.

4.4 Example

This section provides information about how we can create a CICS TS application that represents the existing GENAPP application.

After completing the procedures in this section, you will have created the following components:

- ▶ Two defined entry points in your GENAPP application that enable you to monitor the resource usage of the application across your environment, or to apply threshold policies at the application level
- ▶ A running CICS TS application resource that encompasses dependencies on the GENAPP application, and enables you to see the running status of the application as a whole

4.4.1 Prerequisites

The following sections build on previous instructions described in Chapter 2, “GENAPP introduction” on page 15 and Chapter 3, “Creating a platform” on page 25. To proceed, you need the following software:

- ▶ CICS TS regions running V5.1 or later. The walkthrough is done using CICS TS V5.2.
- ▶ CICS Explorer V5.2, or CICS Explorer V5.2 software development kit (SDK).

Tip: It is possible to follow the steps in this chapter if you are using CICS TS V5.1. If this is the case, you can skip the instructions on making an application Available or Unavailable, because this capability was added in CICS TS V5.2.

- ▶ The GeneralInsuranceDev platform defined and installed from Chapter 3, “Creating a platform” on page 25.
- ▶ The GENAPP application, available and installed into a single managed CICS TS region, as described in Chapter 2, “GENAPP introduction” on page 15 and Appendix A, “Setup and environment” on page 207.

4.4.2 Creating application entry points

Our initial goal is to create a CICS TS application resource, containing the application entry points for the GENAPP application. On completion of this initial goal, the application resource can be installed, and could be used for monitoring or for applying threshold policies.

Our secondary goal is to enhance the CICS TS application to reflect the status of the CICS TS resources that the application relies upon, by adding dependencies. These enable us to determine, at installation, whether the resources needed by the application are available. Checking dependencies is not required for monitoring or applying threshold policies.

Identifying application entry points

There are some factors to consider when identifying entry points:

- ▶ The earlier in the transaction flow that you identify that a task is part of an application, the sooner you can apply policies and start monitoring for the cost of the application. This can be of particular importance if the application spans multiple CICS TS regions. To monitor the application across all of the CICS TS regions that it touches, the entry points should be defined on the CICS TS region that first accepts the inbound request.
- ▶ An application can have different functions that it performs. These are referred to as the *operations* of the application. For example, GENAPP can inquire on a customer, or add a customer. There can be value in identifying these operations, and monitoring or applying policies to individual operations.

Remember: In CICS TS V5.1, an entry point can be a CICS TS PROGRAM resource. In CICS TS V5.2, URIMAPs were added as an additional type of entry point.

Figure 2-6 on page 20 describes how the GENAPP customer application artifacts are separated between presentation logic, application services logic, and data access logic. In our GeneralInsuranceDev platform created in Chapter 3, “Creating a platform” on page 25, all three of these roles are performed by a single CICS TS region.

However, if we were to promote the application to the GeneralInsuranceTest platform, which we also created in Chapter 3, “Creating a platform” on page 25, the application would then have separate CICS TS regions for presentation logic, application services logic, and data access logic.

In terms of identifying our GENAPP entry points, we have three possible options:

1. The first program resource run for GENAPP is LGTESTC1. This is part of the presentation layer. If we identify this as our entry point, we can ensure that tasks in the CICS TS regions, running the presentation layer of our application, are monitored as a part of the application, and that threshold policies can apply in these CICS TS regions.
2. If we wanted to differentiate between adding a customer and inquiring on existing customers, we could use the programs LGICUS01 (inquire) and LGACUS01 (add) as our application entry points. These programs run in the application services logic layer.
Therefore, if we choose these entry points, we can apply more selective threshold policies, and can separately account for the cost of the inquire customer and add customer operations through monitoring. However, the cost of the presentation logic would not be included.
3. We could consider an architectural change to GENAPP, such that different CICS TS programs are driven in the presentation logic layer for the inquire customer and add customer operations. We could then define these new programs as the application entry points, which would give us the benefits of both option 1 and option 2.

In this example, we decided to proceed with option 2, and to create two application entry points in the application services logic layer of the application. This enables us to demonstrate applying policies to a single operation of an application, as shown in Chapter 5, “Applying a policy” on page 97.

GeneralInsuranceCustomerServices Bundle

Before creating the CICS TS application project, we create a CICS bundle to contain the application entry points. The CICS bundle represents the application services layer of the GENAPP application.

Use the following steps to create the GeneralInsuranceCustomerServices CICS Bundle Project:

1. In the CICS Explorer, ensure that you are in the CICS Cloud perspective. (If the top bar of the CICS Explorer window starts with the words CICS Cloud, you are in the CICS Cloud perspective.) If not, click the drop-down menu called **Window**. Select **Open Perspective** → **Other**. Select **CICS Cloud** and click **OK**.
2. Right-click the Project Explorer view, then select **New** → **Project**. The New Project wizard displays.
3. Expand CICS Resource, click **CICS Bundle Project** (see Figure 4-2), then click **Next**. The CICS Bundle Project wizard displays.

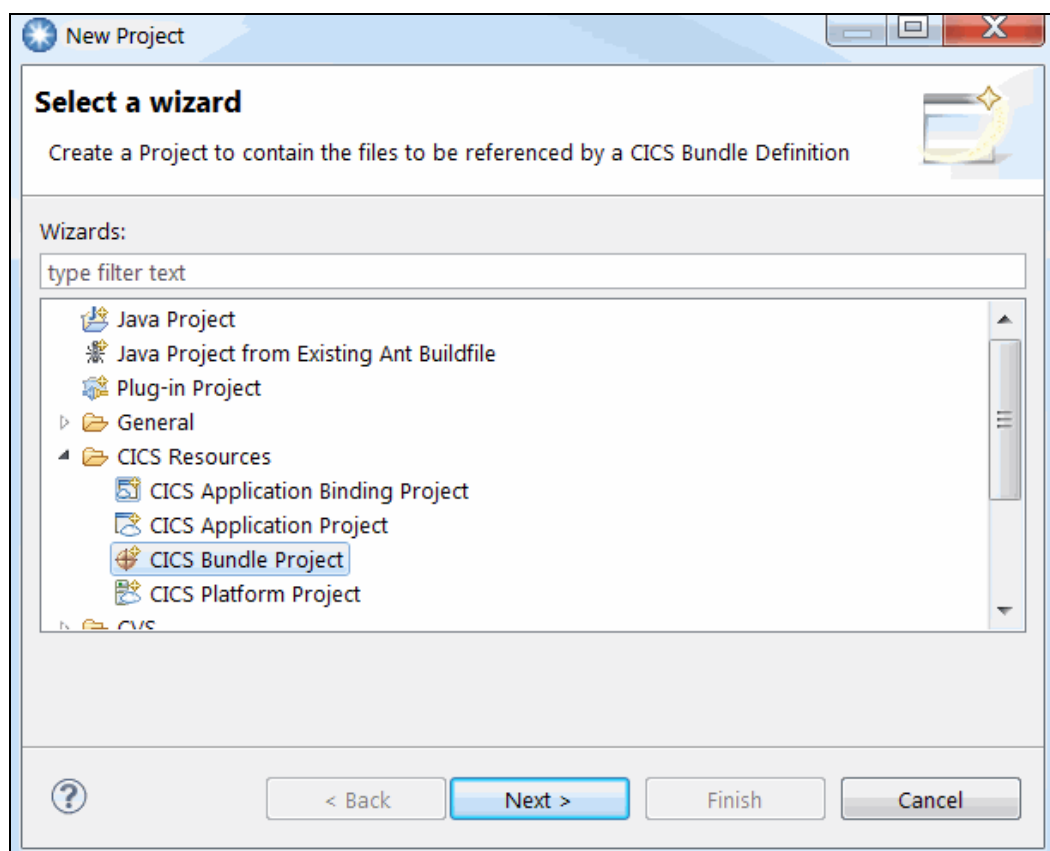


Figure 4-2 Selecting a CICS Bundle Project in the New Project wizard

Alternatively, to create a project to contain the files to be referenced by a CICS bundle definition, click the bundle plus (📦+) icon in the toolbar to start the CICS Bundle Project wizard.

4. Enter `general.insurance.customer.services.bundle` as the Project name.
5. Enter `GeneralInsuranceCustomerServices` as the ID.
6. Enter `1.0.0` in Version. See Figure 4-3 for an example of the completed CICS Bundle Project wizard. Click **Finish**. The bundle project is created and the CICS bundle Manifest editor displays.

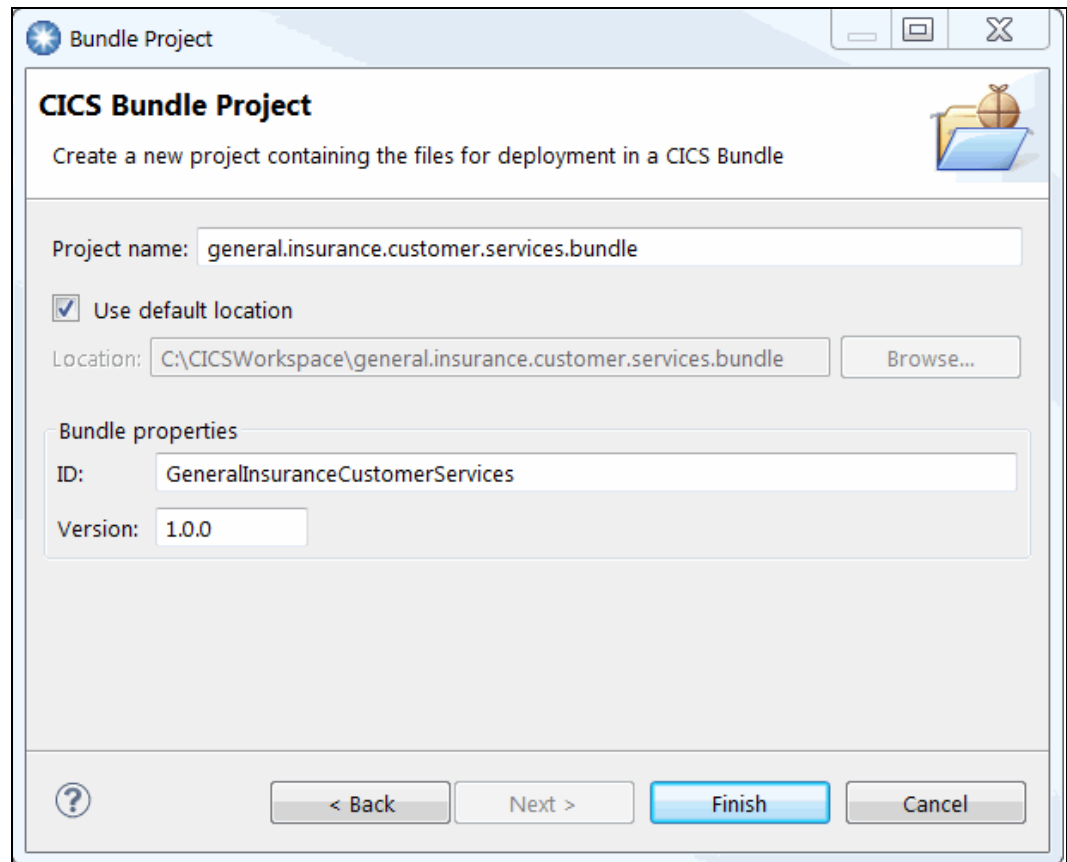


Figure 4-3 Completed CICS Bundle Project wizard for GeneralInsuranceCustomerServices bundle

7. The CICS bundle Manifest editor will automatically start after the creation of the CICS Bundle Project in the previous step, as shown in Figure 4-4. If the editor is not open, locate the bundle manifest file to start the editor. In the Project Explorer view, expand the `general.insurance.customer.services.bundle` project, and expand the **META-INF** subdirectory, then double-click the `cics.xml` manifest file to start the CICS bundle Manifest editor.

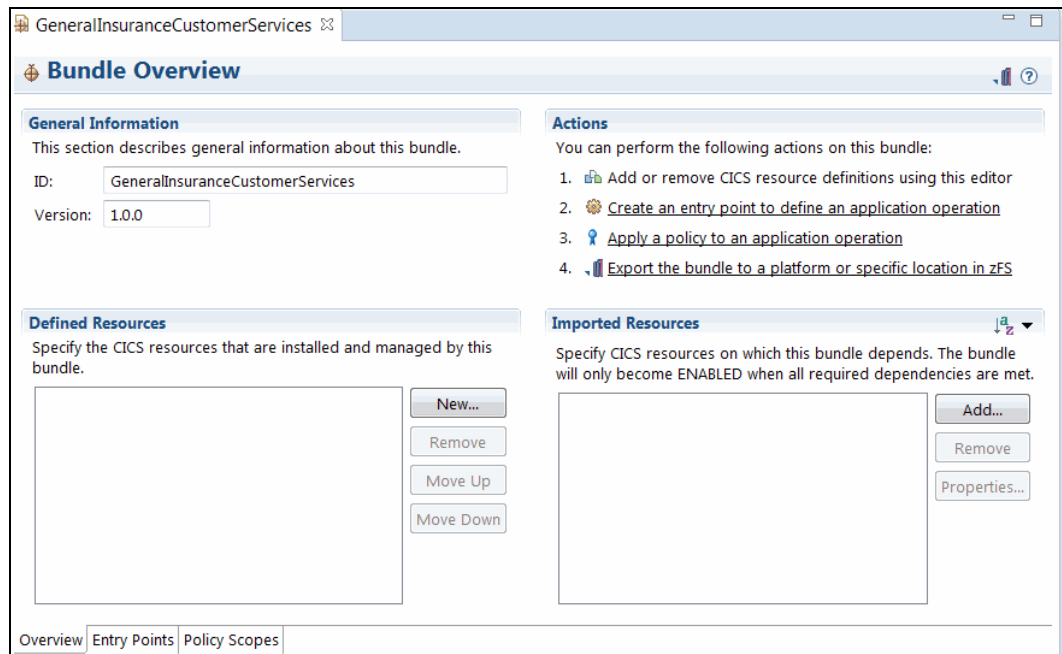


Figure 4-4 CICS bundle Manifest Editor for GeneralInsuranceCustomerServices bundle

8. In the GeneralInsuranceCustomerServices CICS Bundle Manifest Editor, select the **Entry Points** tab.
9. In the Application Entry Points tab, click **Add**. The Create Application Entry Point dialog displays.
10. Enter `addCustomer` in the Operation field.
11. Select the Resource type of **PROGRAM** (for example, `http://www.ibm.com/xmlns/prod/cics/bundle/PROGRAM`) in the drop-down box.
12. Enter `LGACUS01` in the Resource Name field.

13. The Create Application Entry Point dialog should look like Figure 4-5. Click **OK**.

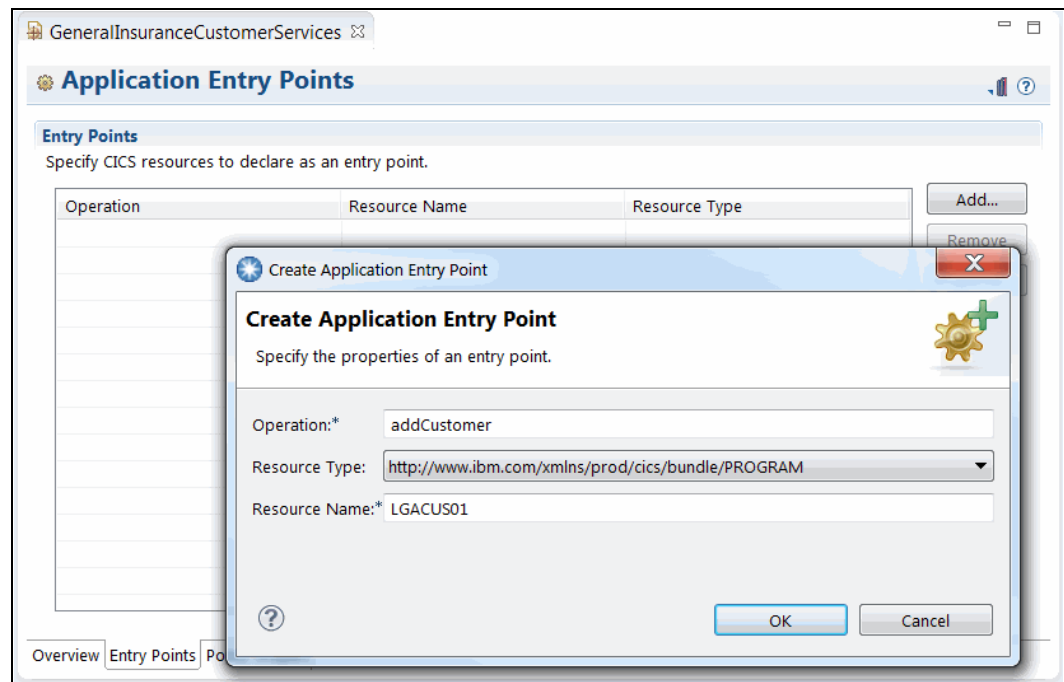


Figure 4-5 Create Application Entry Point dialog for addCustomer operation

14. In the Application Entry Points tab, click **Add**.

15. Enter inquireCustomer in the Operation field.

16. Select the Resource type of PROGRAM (for example, <http://www.ibm.com/xmlns/prod/cics/bundle/PROGRAM>) in the drop-down box.

17. Enter LGICUS01 in the Resource Name field. The Create Application Entry Point dialog should look like Figure 4-6.

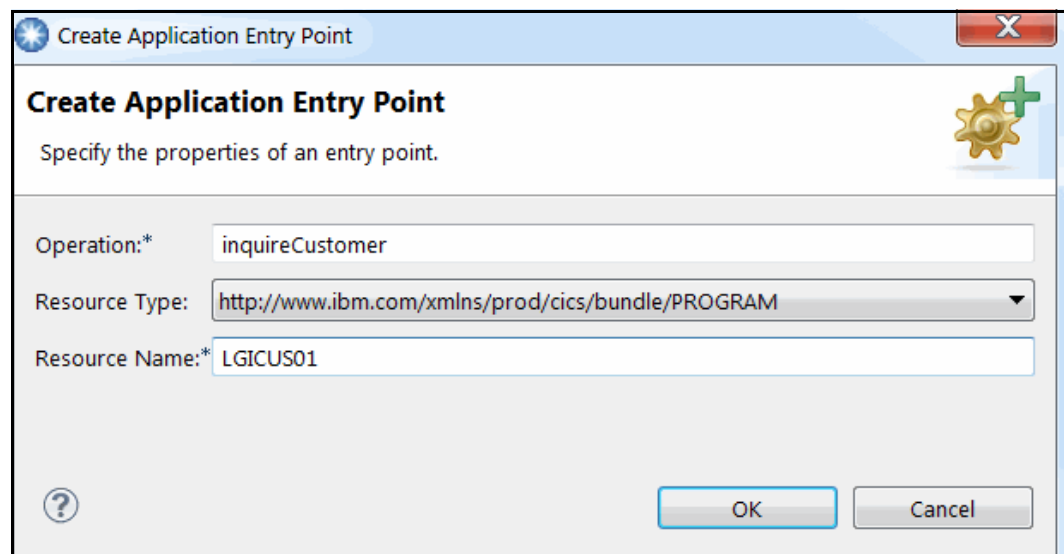


Figure 4-6 Create Application Entry Point dialog for inquireCustomer operation

18. Click **OK**.

19. Press **Ctrl+S** to save the changes.

The GeneralInsuranceCustomerServices CICS Bundle Manifest Editor should look like Figure 4-7.

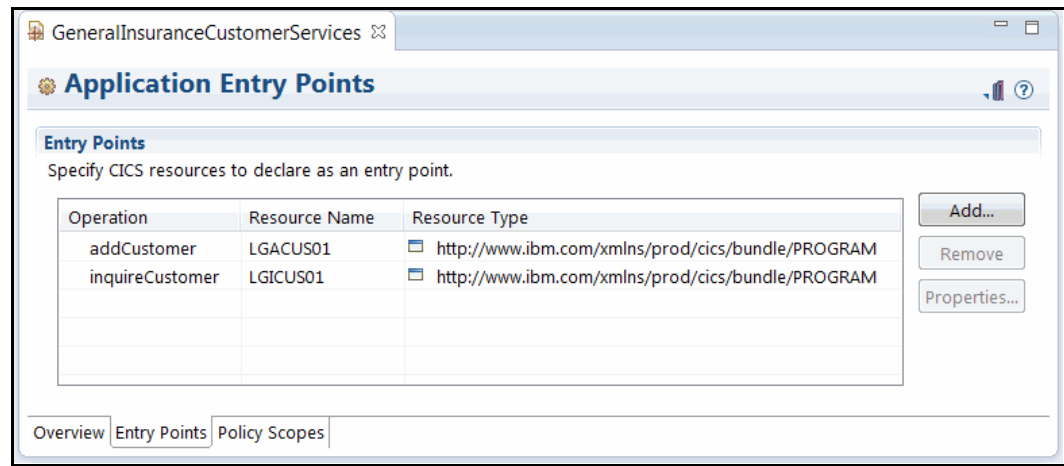


Figure 4-7 GeneralInsuranceCustomerServices CICS Bundle Manifest Editors Entry Points view

The bundle creation is complete.

4.4.3 CICS Application Project

This section describes how to create a CICS Application Project that associates the CICS bundle we have created into a CICS TS application.

Use the following steps to create the GeneralInsuranceCustomer CICS Application Project using the CICS Cloud perspective in CICS Explorer:

1. Right-click in the Project Explorer view, then select **New** → **Project**. The New Project wizard displays.
2. Expand CICS Resource, click **CICS Application Project** (see Figure 4-8 on page 63), then click **Next**. The new Application Project wizard appears.

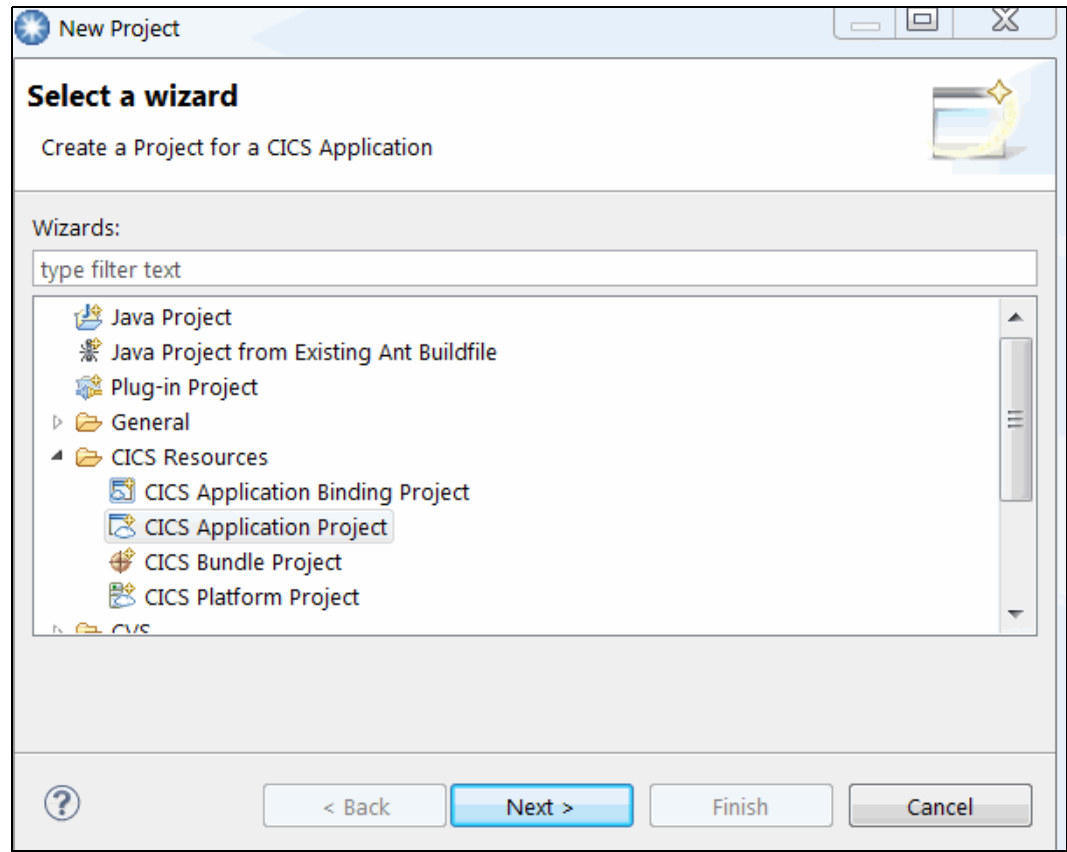


Figure 4-8 Selecting a CICS Application Project in the New Project wizard

Alternatively, to create a project for a CICS TS application, click the window plus cloud (🌩) icon in the toolbar to start the new Application Project wizard.

3. Enter `general.insurance.customer.application` in the Project name field.
4. Enter `GeneralInsuranceCustomer` in the Name field.
5. Enter `General insurance customer application` in the Description field.

6. Enter 1.0.0 in the Version field.

The New Application Project wizard should look like Figure 4-9.

New Application Project

Create a new CICS Application project

Project name:

☒ Use default location

Location:

Application properties

Name:

Description:

Version:

Figure 4-9 New Application Project wizard for GeneralInsuranceCustomer application

7. Click **Next**.

The wizard shows the CICS Bundle Projects that have been found in the workspace that can be included by the CICS TS application.

8. Select the GeneralInsuranceCustomerServices CICS bundle.
The new Application Project wizard should look like Figure 4-10.

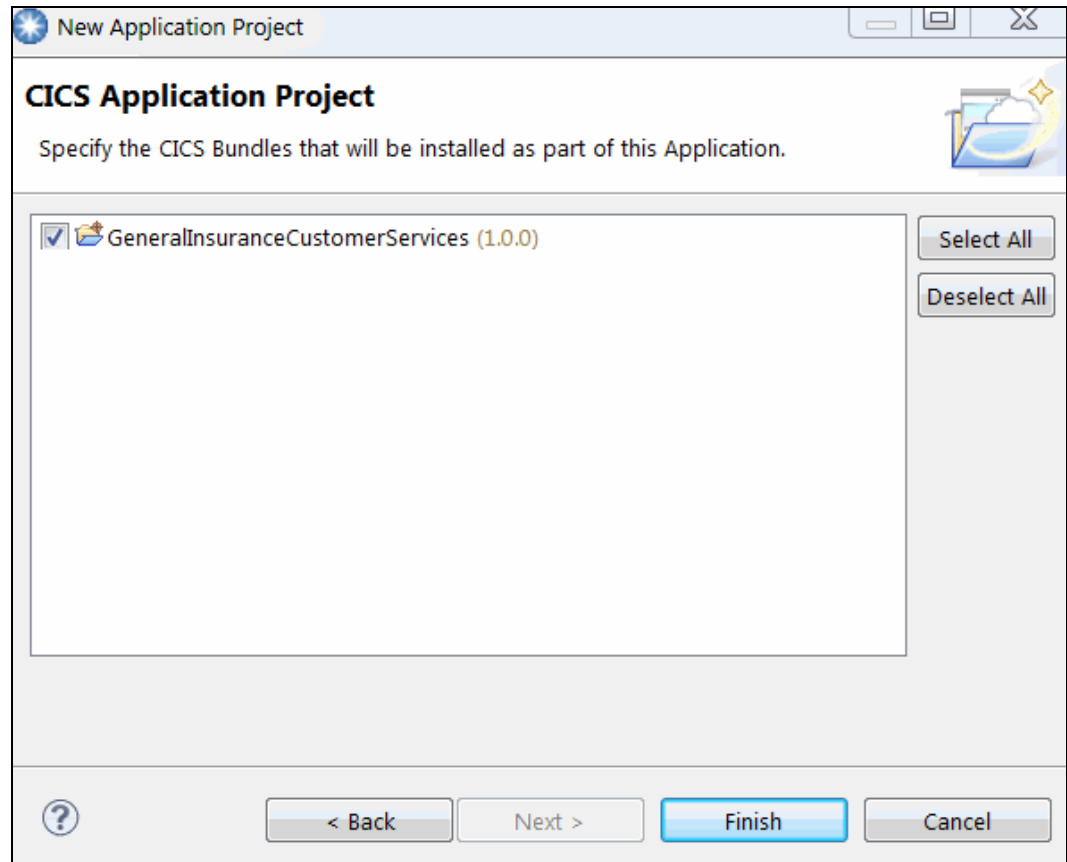


Figure 4-10 New Application Project CICS bundle selection dialog

9. Click **Finish**. The application project is created.
The Project Explorer view should look similar to Figure 4-11.

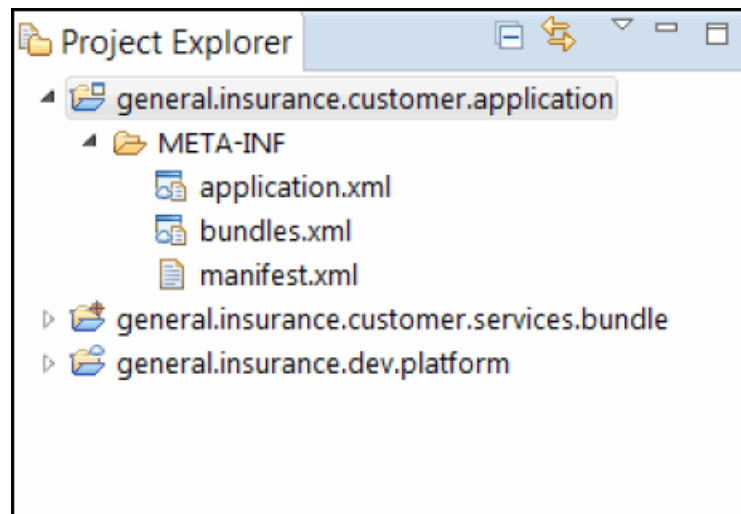


Figure 4-11 Project Explorer view after creating the CICS Application Project

10. The Application editor for the GeneralInsuranceCustomer application will start and should look similar to Figure 4-12.

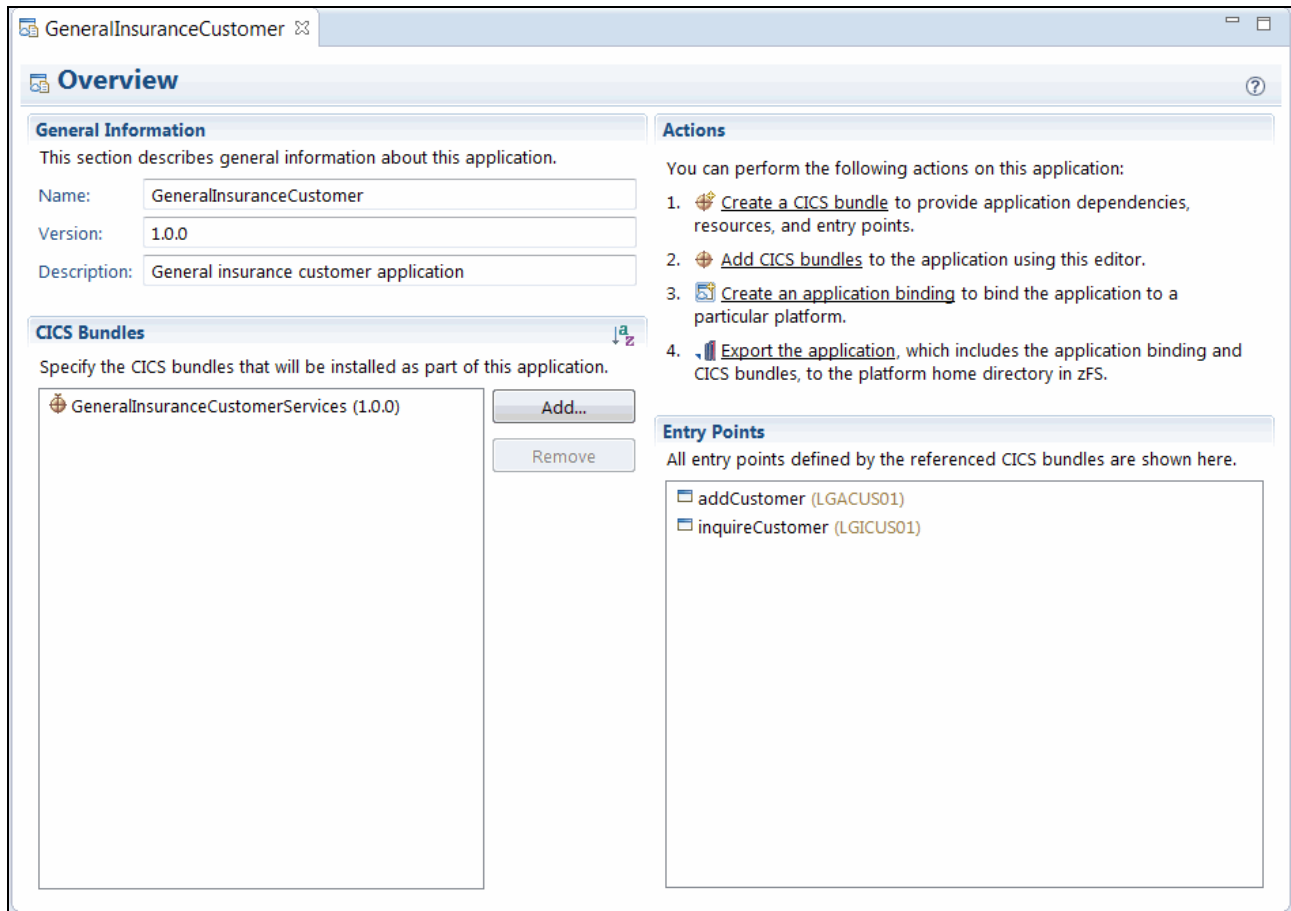


Figure 4-12 GeneralInsuranceCustomer application in the Application Editor

A CICS Application Project has now been created.

4.4.4 CICS Application Binding Project

The CICS application binding is the link between a CICS TS application and a CICS TS platform. It is the mechanism for defining how the associated CICS bundles in a CICS TS application should be installed into the region types of a CICS TS platform. CICS bundles can also be incorporated into CICS application binding, as discussed in Chapter 6, “Packaging an application for multiversion deployment” on page 127.

Use the following steps to create the `general.insurance.customer.to.dev.platform.binding` project using the CICS Cloud perspective in CICS Explorer:

1. Right-click the Project Explorer view, then select **New** → **Project**. The New wizard displays.
2. Expand **CICS Resource**, click **CICS Application Binding Project**, then click **Next**. The New Application Binding Project wizard displays.

Alternatively, to create a project for a CICS application binding, click the cloud plus window (🌥️) icon in the toolbar to start the New Application Binding Project wizard.

3. Enter `general.insurance.customer.to.dev.platform.binding` in the Project name field.
4. Enter `GeneralInsuranceCustomerToDev` in the Name field.
5. Enter `General insurance customer application to dev platform binding` in the Description field.
6. Enter `1.0.0` in the Version field.
7. Select `GeneralInsuranceCustomer (1.0.0)` in the Application field.
8. Select `GeneralInsuranceDev` in the Platform field.

The New Application Binding Project wizard should look like Figure 4-13.

New Application Binding Project

Create an application binding project to map application components to different region types in the platform.

Project name:

☒ Use default location

Location:

Application Binding properties

Name:

Description:

Version:

Application:

Platform:

Figure 4-13 New Application Binding Project wizard

9. Click **Next**.

The wizard shows the CICS Bundle Projects that have been found in the workspace that can be included into the application binding. At this point, you do not want to add any additional CICS bundles to the CICS application binding that you are creating.

10. Click **Next**.

The wizard now enables you to bind the CICS bundles that have been associated to the CICS TS application, or incorporated in this binding, to the region types that have been defined in the CICS TS platform.

11. In CICS bundles, select GeneralInsuranceCustomerServices (1.0.0). In Region Types, select ApplicationServices. The wizard should look similar to Figure 4-14.

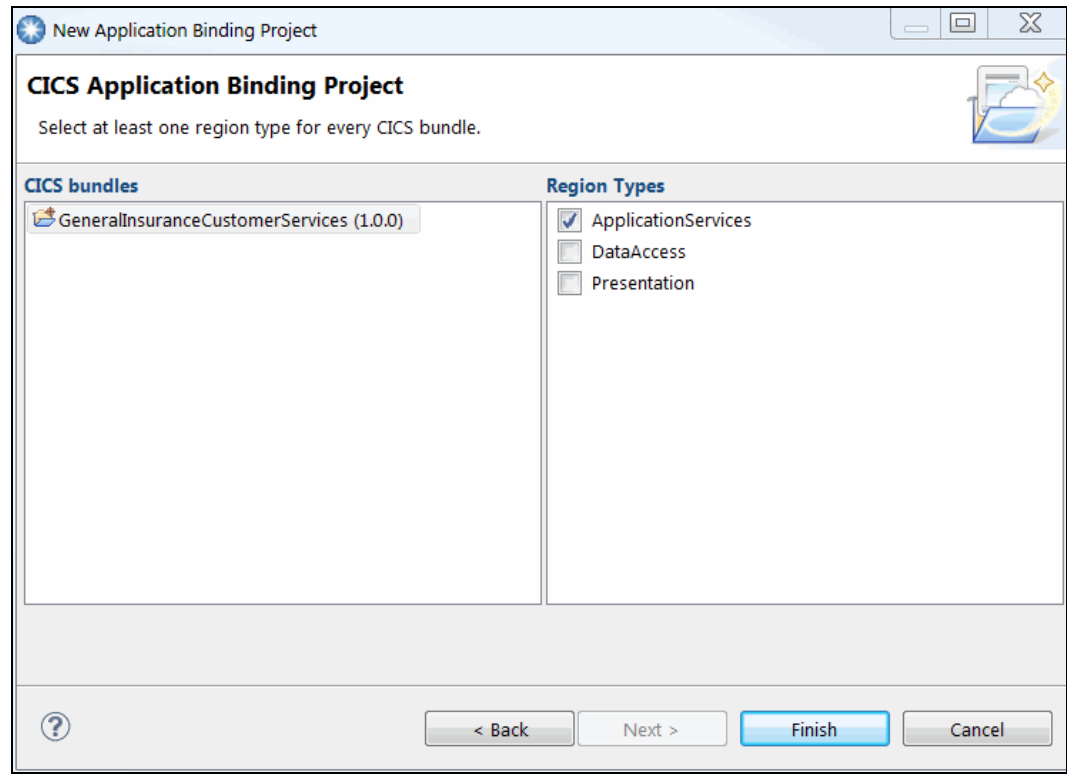


Figure 4-14 Deployment bindings of CICS bundles to region types

12. Click **Finish**. The application binding project is created.
The Project Explorer view should look similar to Figure 4-15.

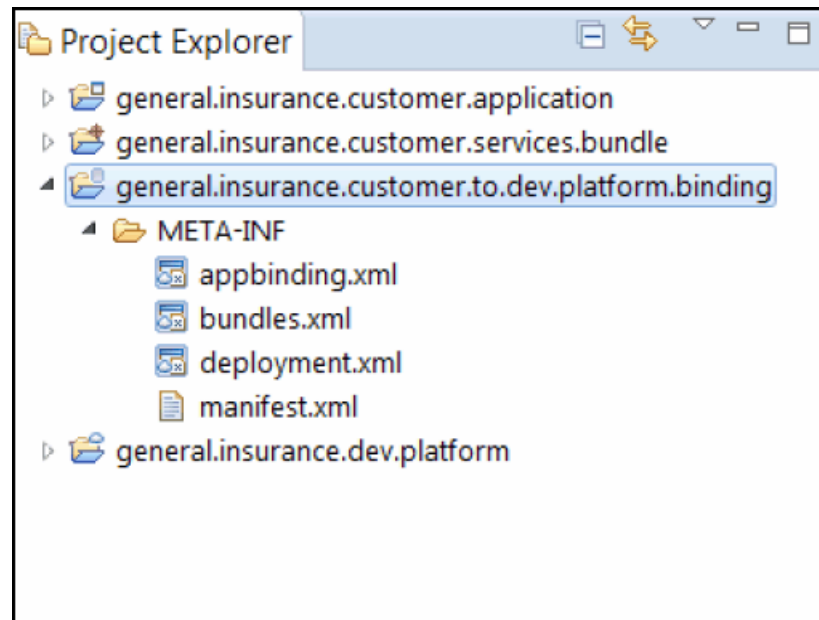


Figure 4-15 Project Explorer view after creating the GeneralInsuranceCustomerToTest CICS Application Binding Project

13. The application binding editor for the GeneralInsuranceCustomerToDev binding will start and should look similar to Figure 4-16.

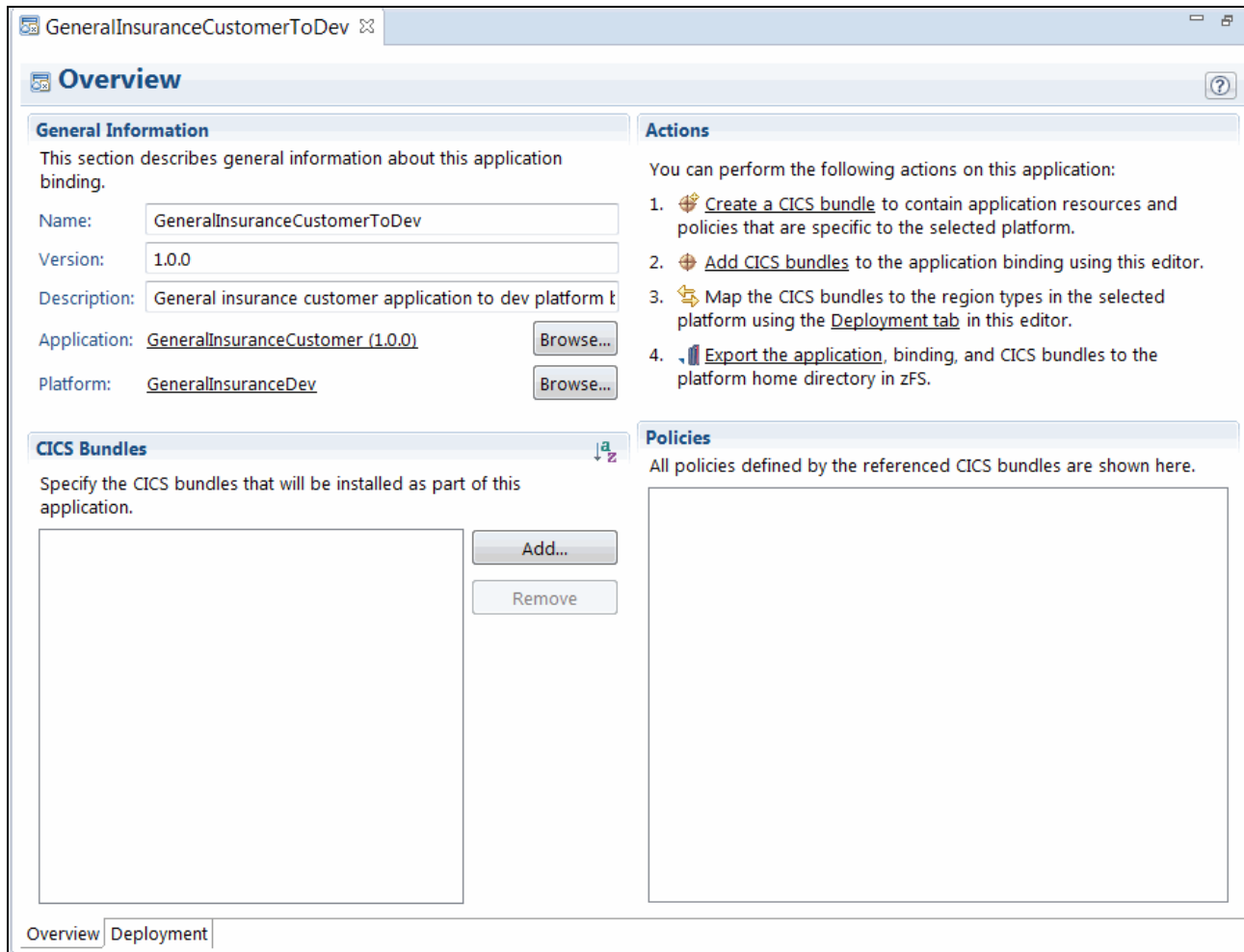


Figure 4-16 GeneralInsuranceCustomerToTest application binding in the application binding editor

The CICS Application Binding Project has now been created. This project defines how the CICS TS application, GeneralInsuranceCustomer, is bound to the CICS TS platform, GeneralInsuranceDev.

4.4.5 Deploying the CICS TS application and creating the application definition

The CICS Application Project, CICS Application Binding Project, and all associated CICS Bundle Projects can be deployed to the z/OS UNIX file system in a single export step. When deployed, the CICS TS application can then be installed and managed in your CICS TS environment.

Use the following steps to deploy the CICS TS application, CICS application binding and CICS Bundle Projects using the application editor:

1. If the Application editor for project general.insurance.customer.application is not open, locate the application files to start the application editor.

In the Project Explorer view, expand project `general.insurance.customer.application` and expand the subdirectory `META-INF`. Next, double-click the `application.xml` file to start the application editor, as shown in Figure 4-12 on page 66.

2. In the Actions section of the Application editor, click **Export the application**.
The Export Application to the home directory of a Platform wizard displays. This wizard enables you to specify the CICSplex and CICS TS platform to which you want to deploy your CICS TS application. This action identifies the CICS application binding that is used for deployment of the application.
3. If required, connect to your CICS TS environment's CMCI connection.
4. In GNAPPLEX, under `GeneralInsuranceDev`, select `GeneralInsuranceCustomerToDev`, as shown in Figure 4-17.
5. Select **Create Application Definition after export finishes**.

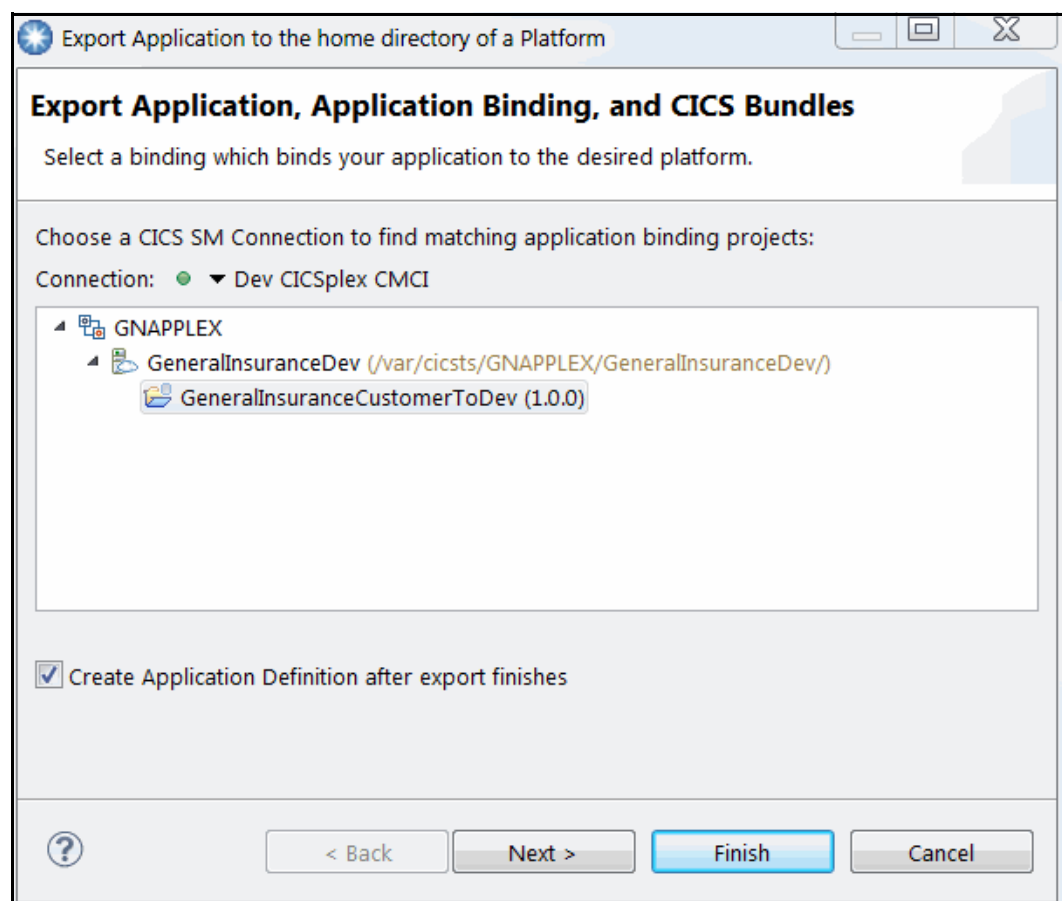


Figure 4-17 Export Application to the home directory of a platform

6. Click **Next**.

7. If required, connect to your CICS TS environment's z/OS FTP connection, as shown in Figure 4-18. Click **Finish**.

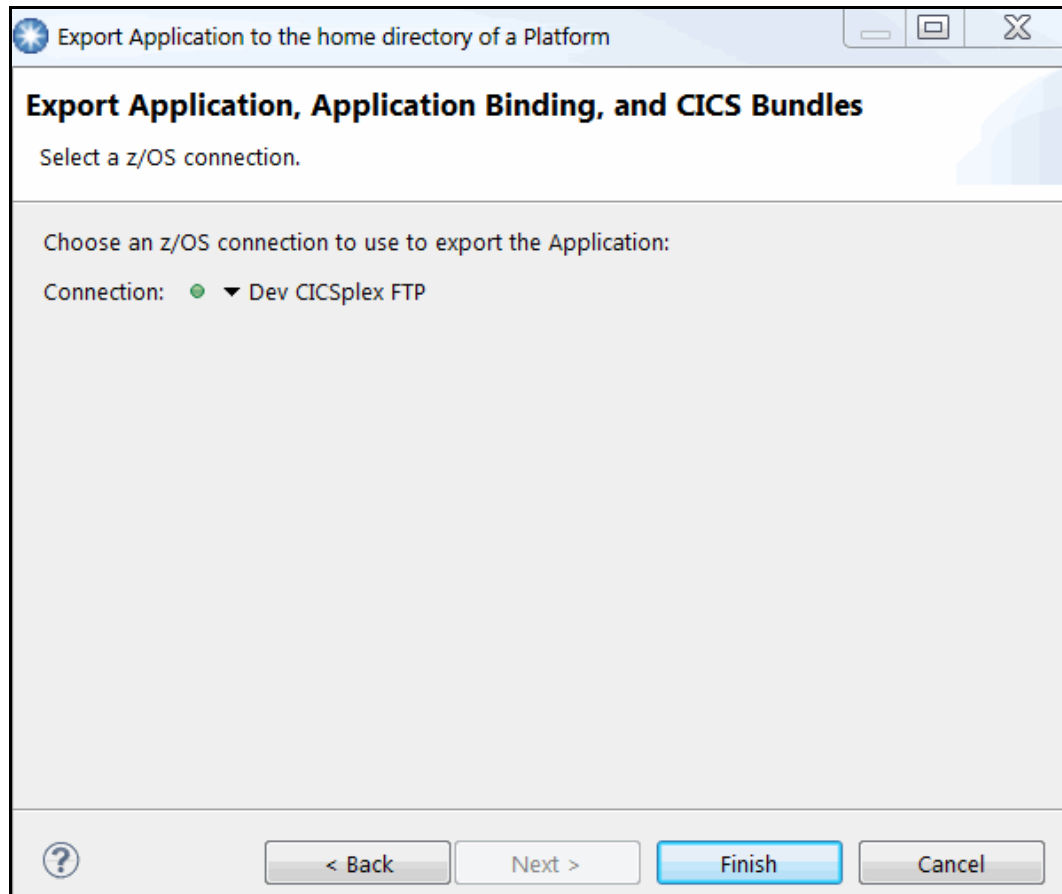


Figure 4-18 Established FTP connection for exporting files to the z/OS UNIX file system

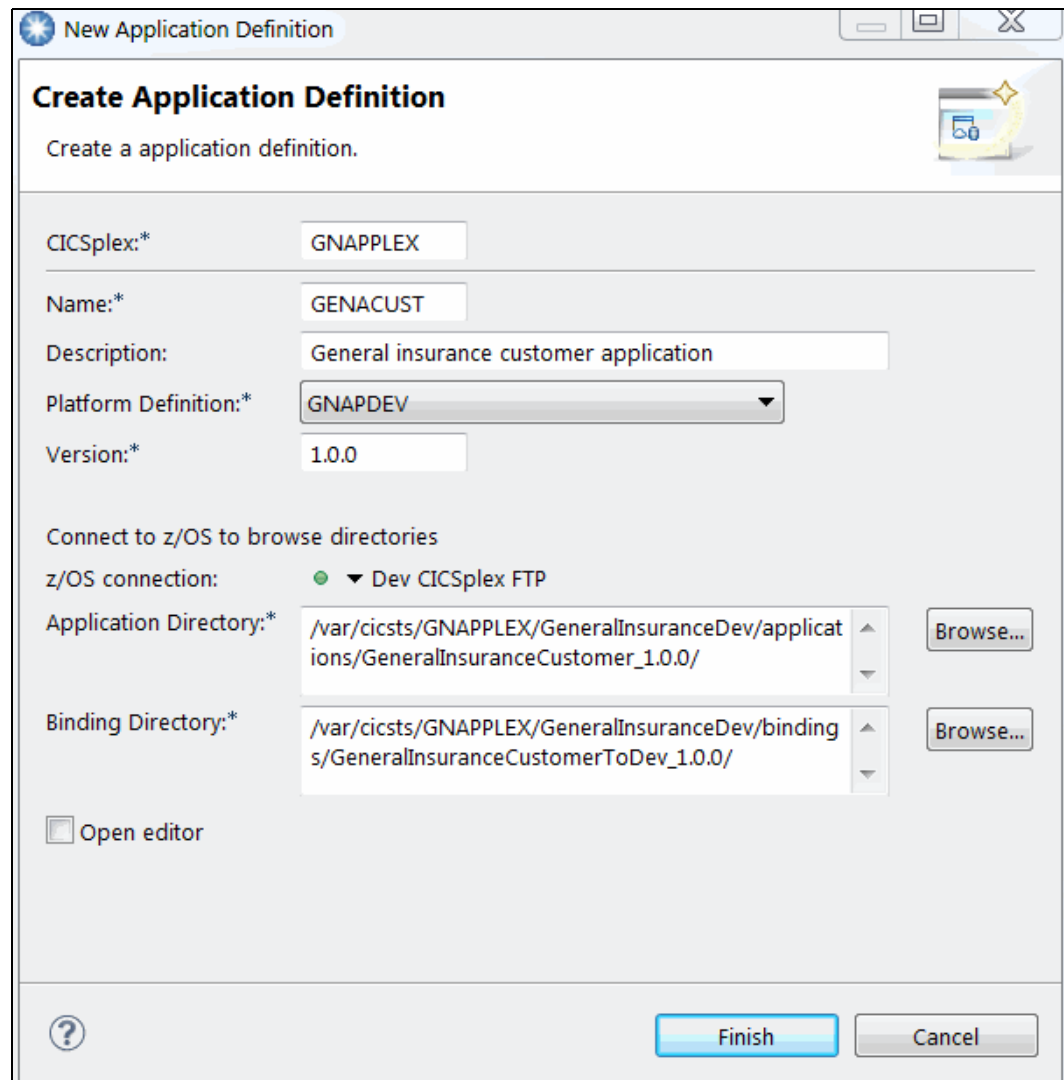
The CICS Explorer begins to export the CICS TS application, CICS application binding, and CICS Bundle Projects to the z/OS UNIX file system. The projects are deployed to the platform home directory of the CICS TS platform in the applications, bindings, and bundles subfolders, where appropriate.

When the export has completed, the New Application Definition wizard displays. This wizard can be used to define the CICS TS application to your CICS TS environment. All fields except the Name field have been completed using information found during the export process.

Use the following steps to define the CICS TS application definition:

1. Enter GENACUST in the Name field.
2. Enter General insurance customer application in the Description field.

The New Application Definition wizard should look like Figure 4-19.



The screenshot shows the 'New Application Definition' wizard window. The title bar says 'New Application Definition'. The main heading is 'Create Application Definition' with a subtext 'Create a application definition.' and a small icon of a document with a star. The form contains the following fields and values:

- CICSplex:* GNAPPLEX
- Name:* GENACUST
- Description: General insurance customer application
- Platform Definition:* GNAPDEV (selected in a dropdown)
- Version:* 1.0.0
- Connect to z/OS to browse directories
- z/OS connection: Dev CICSplex FTP (selected with a green dot and dropdown arrow)
- Application Directory:* /var/cicsts/GNAPPLEX/GeneralInsuranceDev/applications/GeneralInsuranceCustomer_1.0.0/ (with a 'Browse...' button)
- Binding Directory:* /var/cicsts/GNAPPLEX/GeneralInsuranceDev/bindings/GeneralInsuranceCustomerToDev_1.0.0/ (with a 'Browse...' button)
- ☐ Open editor

At the bottom, there is a question mark icon, a 'Finish' button, and a 'Cancel' button.

Figure 4-19 Completed New Application Definition wizard

3. Click **Finish**. The CICS TS application definition has now been created. The Cloud Explorer view should now look similar to Figure 4-20.

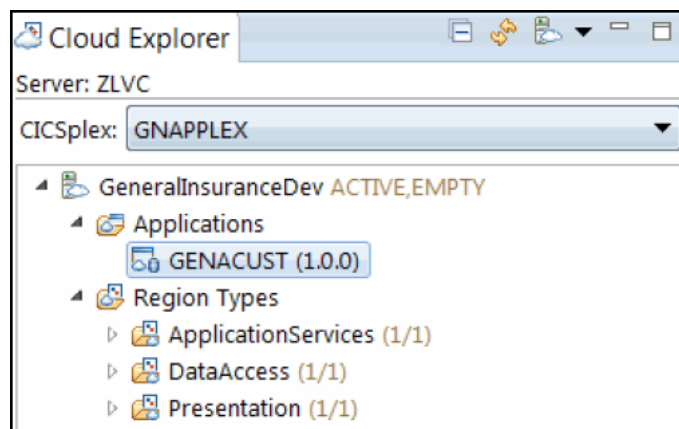


Figure 4-20 Cloud Explorer view after the CICS TS application definition has been created

The CICS TS application, CICS application binding and CICS bundle that comprises the GeneralInsuranceCustomer application have been created and exported to the GeneralInsuranceDev platform. The application definition has been created and is ready to be installed into the GeneralInsuranceDev platform.

4.4.6 Installing and managing the CICS TS application

This section shows how easy it is to install a CICS TS application, and therefore all CICS bundles associated with your CICS TS application across all CICS TS regions in the CICS TS platform's region types.

Use the following steps to install the GeneralInsuranceCustomer application using the CICS Cloud perspective in CICS Explorer:

1. In the Cloud Explorer view, expand the GeneralInsuranceDev platform and expand the Applications subdirectory. Right-click the **GENACUST (1.0.0)** CICS TS application definition and click **Install**, as shown in Figure 4-21. The Perform INSTALL Operation dialog displays.

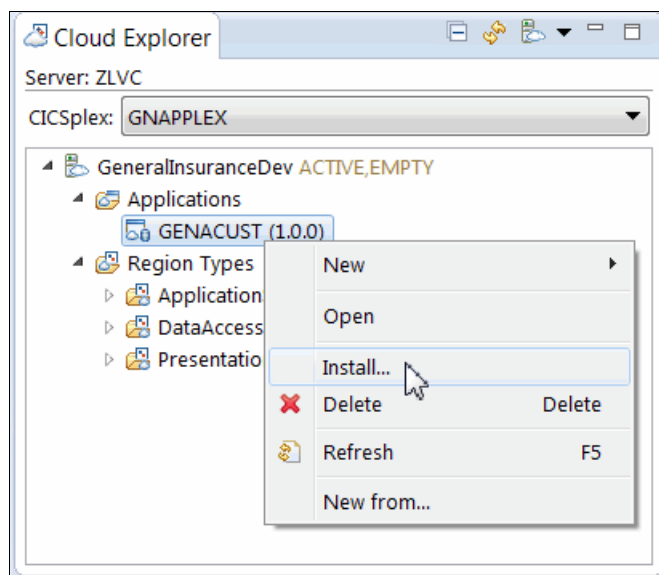


Figure 4-21 Menu to install an application using the Cloud Explorer view

2. In the Perform INSTALL Operation dialog, click **OK**.

The CICS TS application and its associated CICS bundle will now install into the CICS TS regions in the CICS TS platform. The GeneralInsuranceCustomer application is visible in the Cloud Explorer, as shown in Figure 4-22 on page 74.

Tip: The installation action on the CICS TS application installs the CICS bundles that comprise the application, into all of the CICS TS regions in the platform region types, as defined by the deployment bindings. If the CICS TS application state does not immediately display as DISABLED, then wait for 15 seconds and refresh the Cloud Explorer view.

The installed application is visible, as shown in Figure 4-22.

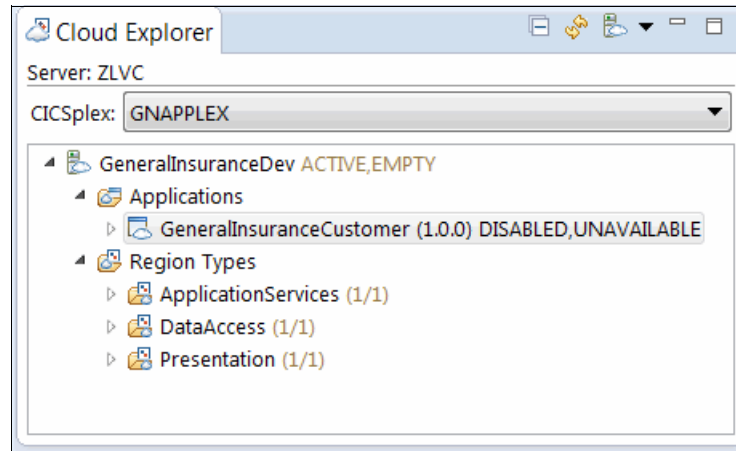


Figure 4-22 Application GeneralInsuranceCustomer visible in the Cloud Explorer view after being installed

The CICS TS application installs all associated bundles into a disabled and unavailable state. This enables the application to install across all of the CICS TS regions before any of its resources are enabled.

Use the following steps to enable the GeneralInsuranceCustomer CICS TS application:

1. Double-click the GeneralInsuranceCustomer (1.0.0) application in the Cloud Explorer view. The Online Application editor for the GeneralInsuranceCustomer application appears, as shown in Figure 4-23.

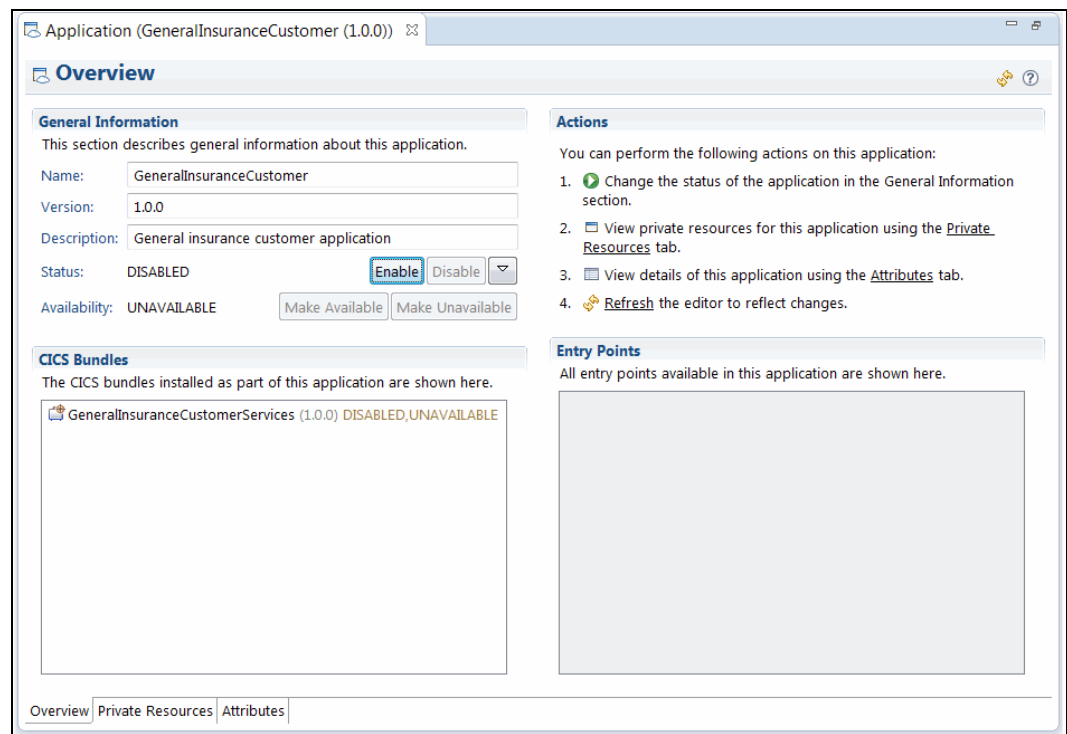


Figure 4-23 Online Application editor for the Installed GeneralInsuranceCustomer application

2. On the Overview tab of the Online Application editor for the GeneralInsuranceCustomer application, in the General Information section, click the **Enable** button. The Perform ENABLE Operation dialog displays, as shown in Figure 4-24.

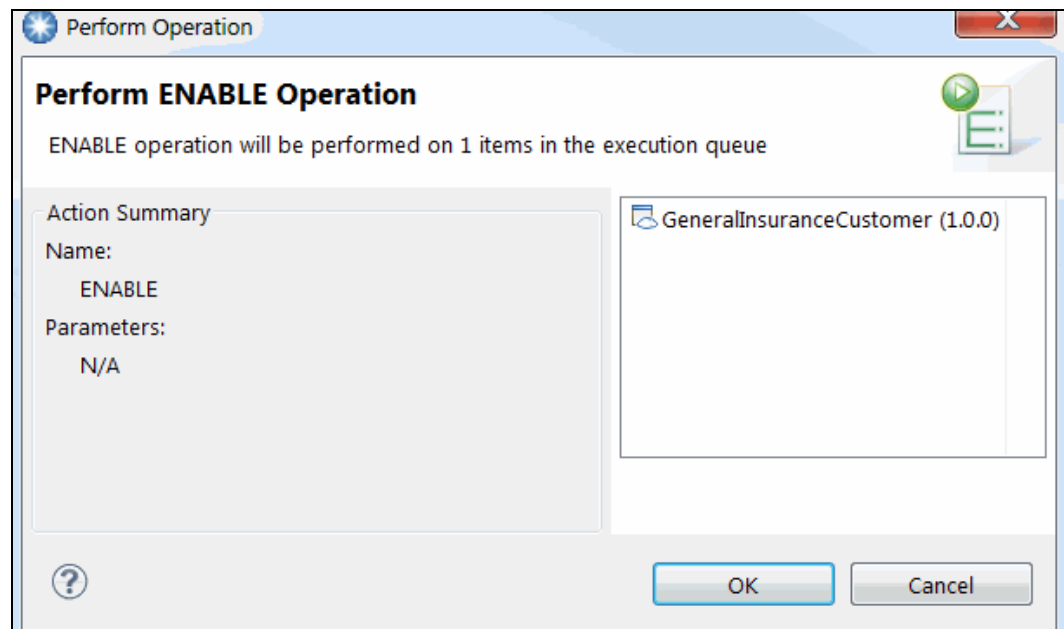


Figure 4-24 perform Operation dialog to enable application GeneralInsuranceCustomer

3. Click **OK**.

The CICS TS application will now attempt to become enabled.

The application is now enabled in an unavailable state. The unavailable state means that the application's entry points are not yet applied.

Important: The value of the additional Make Available/Unavailable step is demonstrated in Chapter 6, "Packaging an application for multiversion deployment" on page 127, where application multi-versioning is demonstrated.

Next, Make Available the GeneralInsuranceCustomer CICS TS application using the Online Application editor.

4. On the Overview tab of the Online Application editor for the GeneralInsuranceCustomer application, in the General Information section, click **Make Available**. The Perform AVAILABLE Operation dialog displays.

5. Click **OK**. The application is made available, and the CICS Cloud perspective looks similar to Figure 4-25.

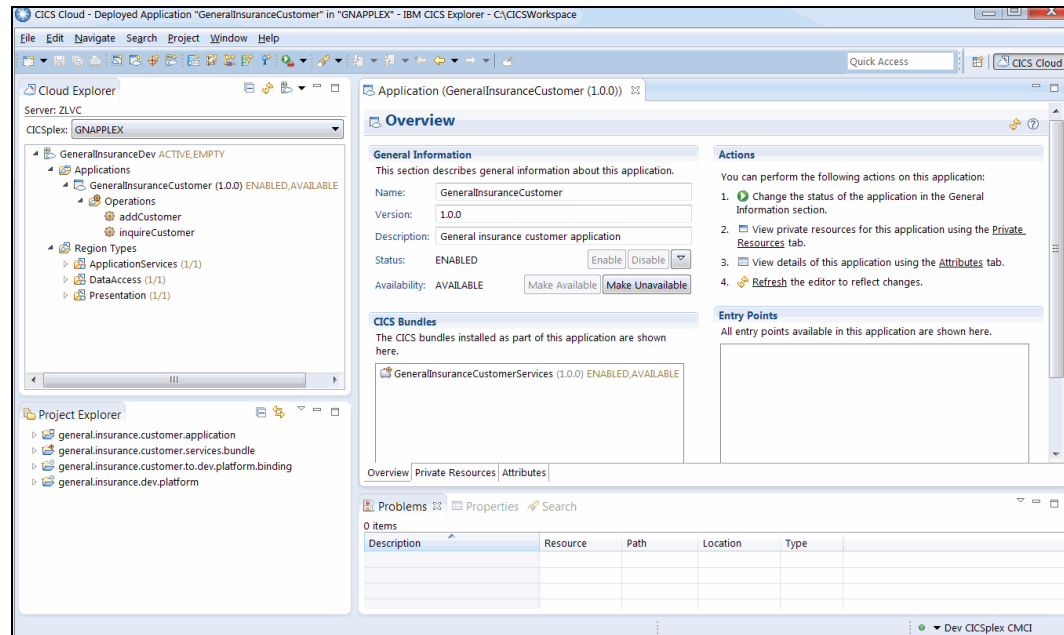


Figure 4-25 The CICS Cloud perspective showing a provisioned CICS TS application

The GeneralInsuranceCustomer application is now available for users to start through the identified application entry points. The application entry points set the application context on tasks, which supports monitoring of resource consumption and policing of threshold policies for the application.

4.4.7 Adding dependencies into the CICS TS application

An additional optional step that can be performed is to configure the CICS TS application to include dependencies on CICS TS resources that the application relies upon. We can add dependencies on the GENAPP resources, so that our application's enabled state reflects the state of the underlying resources.

Before updating the CICS Application Project, we create additional CICS bundles that reflect the state of resources in the GENAPP presentation layer and the data access layer:

- ▶ general.insurance.customer.presentation.bundle
- ▶ general.insurance.customer.data.bundle

GeneralInsuranceCustomerPresentation Bundle

The first CICS bundle represents the presentation layer of the GENAPP application. To accurately reflect the status of this layer of the application, the CICS bundle has a dependency on the following CICS TS resources:

- ▶ TRANSACTION SSC1
- ▶ PROGRAM LGTESTC1
- ▶ PROGRAM SSMAP
- ▶ LIBRARY GENALIB

You can add bundle dependencies as imports in the Bundle manifest file (`cics.xml`). Bundle dependencies are CICS TS resources that must be present in the CICS TS region for the CICS bundle to become enabled.

Use the following steps to create the `GeneralInsuranceCustomerPresentation` CICS Bundle Project using the CICS Cloud perspective in CICS Explorer:

1. Right-click the Project Explorer view, then select **New** → **Project**. The New Project wizard displays.
2. Expand **CICS Resource**, click **CICS Bundle Project**, then click **Next**. The CICS Bundle Project wizard displays.

Alternatively, to create a project to contain the files to be referenced by a CICS bundle definition, click the bundle plus (📁) icon in the toolbar to start the CICS Bundle Project wizard.

3. Enter `general.insurance.customer.presentation.bundle` as the Project name.
4. Enter `GeneralInsuranceCustomerPresentation` as the ID.
5. Enter `1.0.0` as the Version. Figure 4-26 shows the completed Bundle Project wizard for the `GeneralInsuranceCustomerPresentation` bundle.

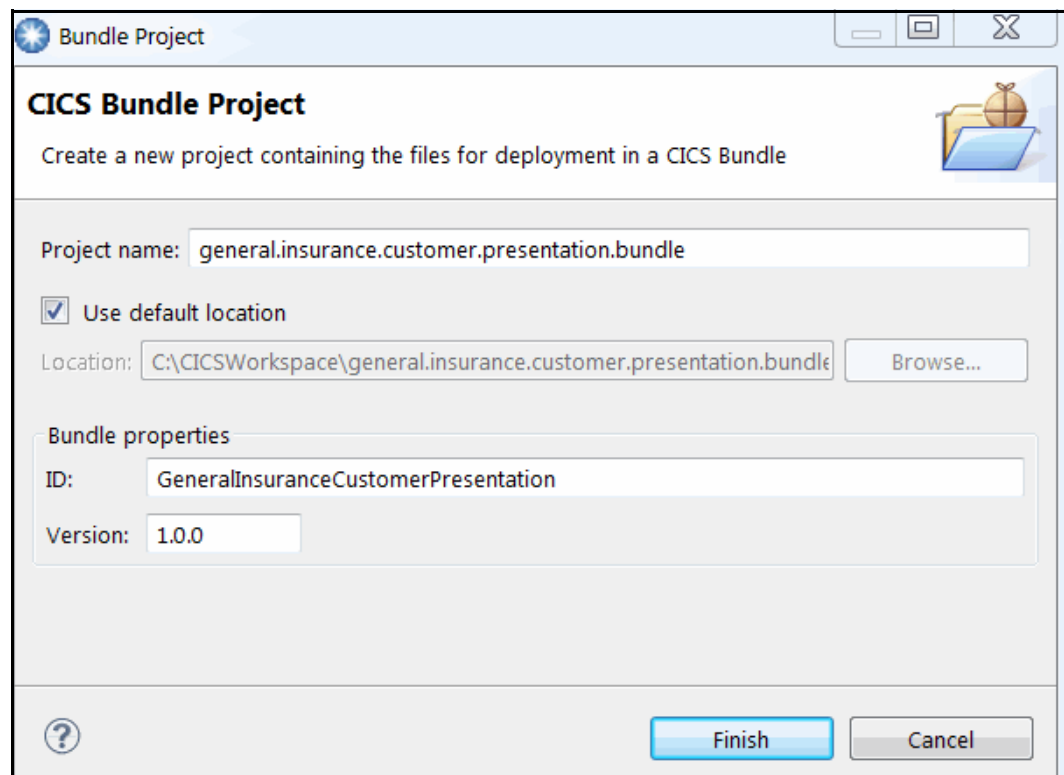
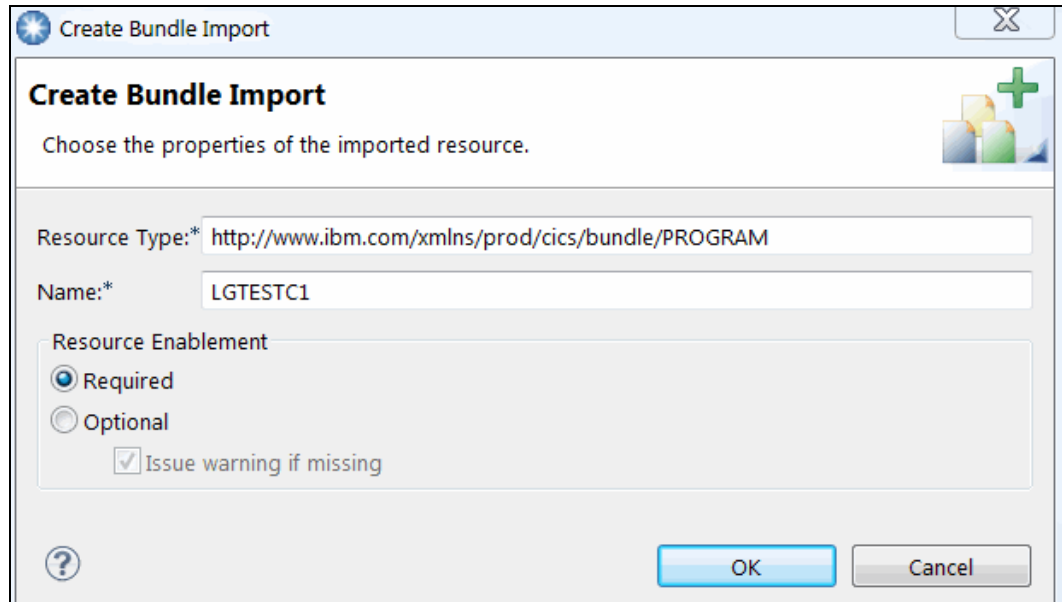


Figure 4-26 New CICS Bundle Project wizard for `GeneralInsuranceCustomerPresentation` bundle

6. Click **Finish**. The bundle project is created and the CICS Bundle Manifest Editor appears.
7. The CICS Bundle Manifest Editor will automatically start after the creation of the CICS Bundle Project in the previous step. If the editor is not open, locate the bundle manifest file to start the editor. In the Project Explorer view, expand the `general.insurance.customer.presentation.bundle` project and expand the `META-INF` subdirectory, then double-click the `cics.xml` manifest file to start the CICS Bundle Manifest Editor.

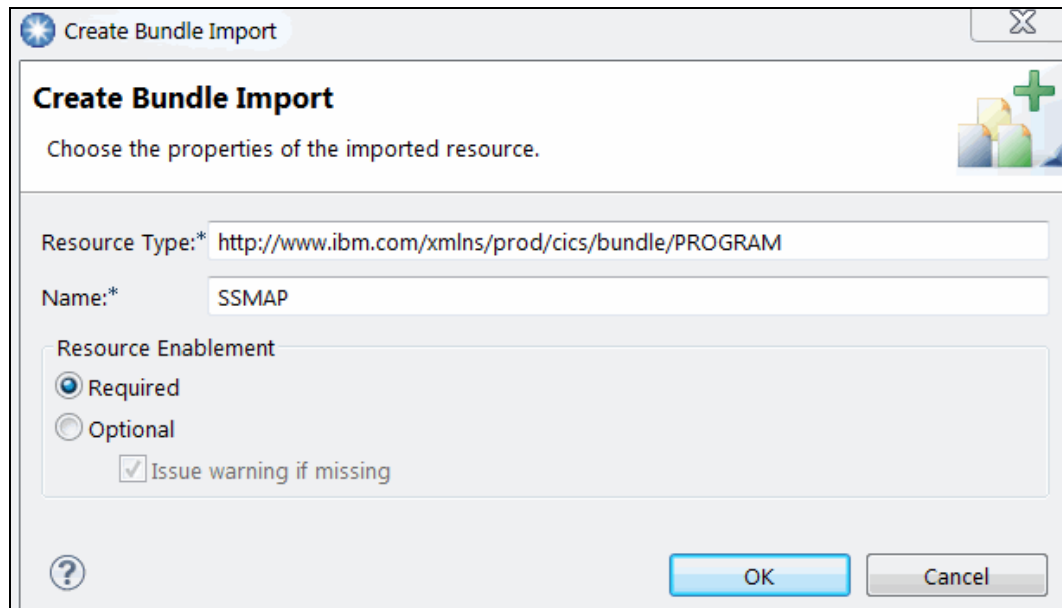
8. On the Overview tab in the CICS Bundle Manifest Editor for the GeneralInsuranceCustomerPresentation bundle, in the Imported Resources section, click **Add**.
9. Select PROGRAM in Resource Type, then enter LGTESTC1 as the Name.
10. Figure 4-27 shows how the Create Bundle Import dialog should look. Click **OK**.



The image shows a 'Create Bundle Import' dialog box. The title bar says 'Create Bundle Import' with a close button. The main title is 'Create Bundle Import' and the subtitle is 'Choose the properties of the imported resource.' Below this, there are two text input fields: 'Resource Type:*' with the value 'http://www.ibm.com/xmlns/prod/cics/bundle/PROGRAM' and 'Name:*' with the value 'LGTESTC1'. Under the 'Name:*' field is a 'Resource Enablement' section with two radio buttons: 'Required' (selected) and 'Optional'. Below these is a checkbox labeled 'Issue warning if missing' which is checked. At the bottom right are 'OK' and 'Cancel' buttons. A help icon (?) is at the bottom left.

Figure 4-27 Create Bundle Import dialog for program LGTESTC1

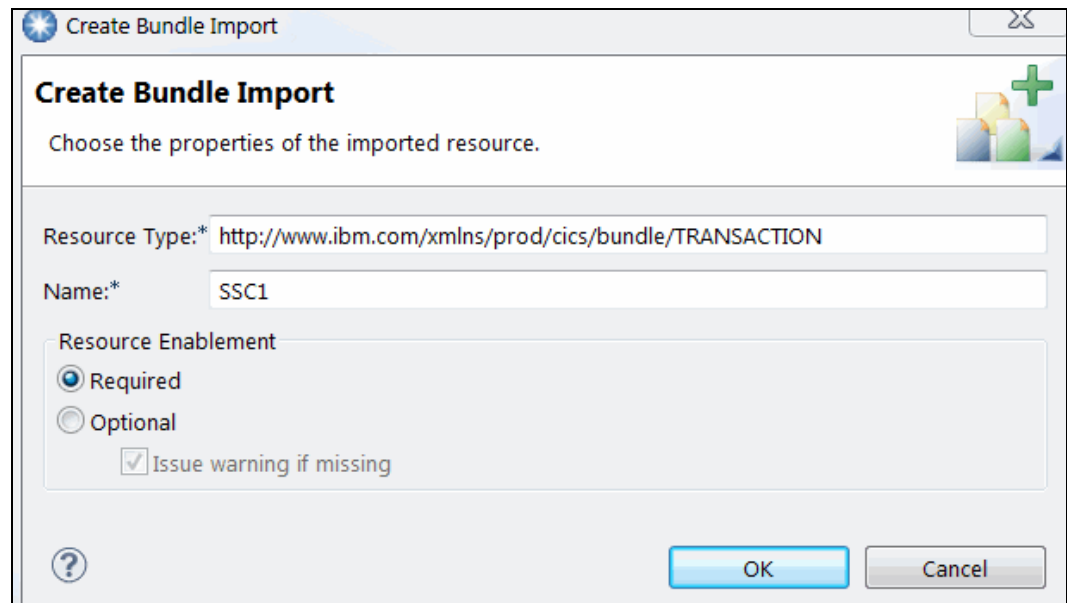
11. In Imported Resources, click **Add**. Select PROGRAM as the Resource Type, then enter SSMAP in Name. Figure 4-28 shows how the Create Bundle Import dialog should look. Click **OK**.



The image shows a 'Create Bundle Import' dialog box, similar to Figure 4-27. The title bar says 'Create Bundle Import' with a close button. The main title is 'Create Bundle Import' and the subtitle is 'Choose the properties of the imported resource.' Below this, there are two text input fields: 'Resource Type:*' with the value 'http://www.ibm.com/xmlns/prod/cics/bundle/PROGRAM' and 'Name:*' with the value 'SSMAP'. Under the 'Name:*' field is a 'Resource Enablement' section with two radio buttons: 'Required' (selected) and 'Optional'. Below these is a checkbox labeled 'Issue warning if missing' which is checked. At the bottom right are 'OK' and 'Cancel' buttons. A help icon (?) is at the bottom left.

Figure 4-28 Create Bundle Import dialog for program SSMAP

12. In Imported Resources, click **Add**. Select TRANSACTION as the Resource Type, then enter SSC1. Figure 4-29 shows how the Create Bundle Import dialog should look. Click **OK**.

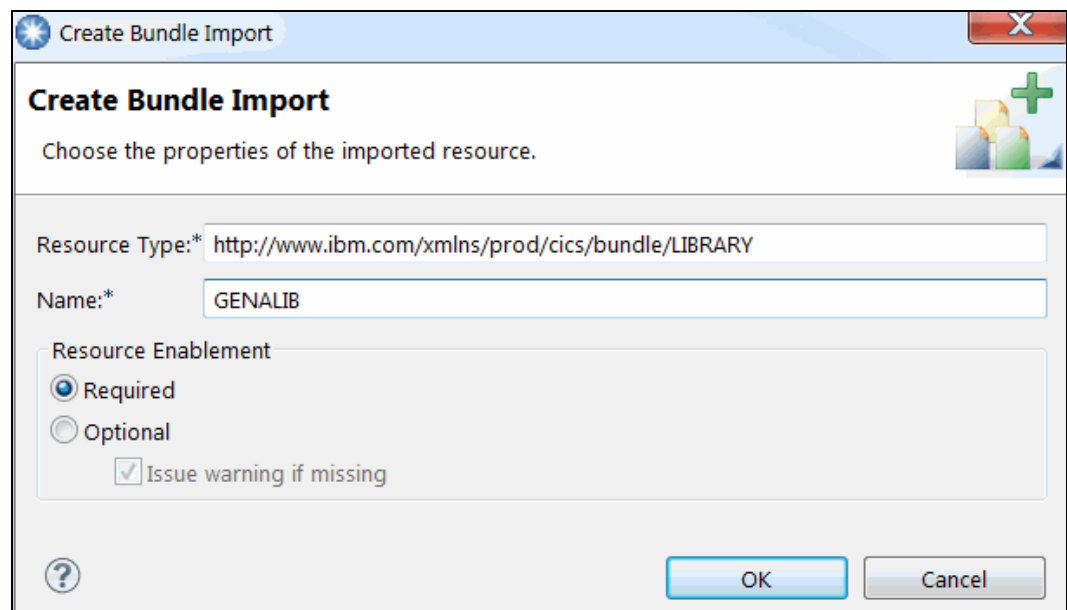


The dialog box is titled "Create Bundle Import" and contains the instruction "Choose the properties of the imported resource." It features two text input fields: "Resource Type:*" with the value "http://www.ibm.com/xmlns/prod/cics/bundle/TRANSACTION" and "Name:*" with the value "SSC1". Below these is a "Resource Enablement" section with two radio buttons: "Required" (selected) and "Optional". A checkbox labeled "Issue warning if missing" is checked. The dialog includes a help icon (question mark), an "OK" button, and a "Cancel" button.

Figure 4-29 Create Bundle Import dialog for transaction SSC1

13. In Imported Resources, click **Add**. Select LIBRARY as the Resource Type, then enter GENALIB.

14. Figure 4-30 shows how the Create Bundle Import dialog should look. Click **OK**.



The dialog box is titled "Create Bundle Import" and contains the instruction "Choose the properties of the imported resource." It features two text input fields: "Resource Type:*" with the value "http://www.ibm.com/xmlns/prod/cics/bundle/LIBRARY" and "Name:*" with the value "GENALIB". Below these is a "Resource Enablement" section with two radio buttons: "Required" (selected) and "Optional". A checkbox labeled "Issue warning if missing" is checked. The dialog includes a help icon (question mark), an "OK" button, and a "Cancel" button.

Figure 4-30 Create Bundle Import dialog for LIBRARY GENALIB

15. Press **Ctrl+S** to save changes.

The GeneralInsuranceCustomerPresentation CICS Bundle Manifest Editor should look as shown in Figure 4-31.

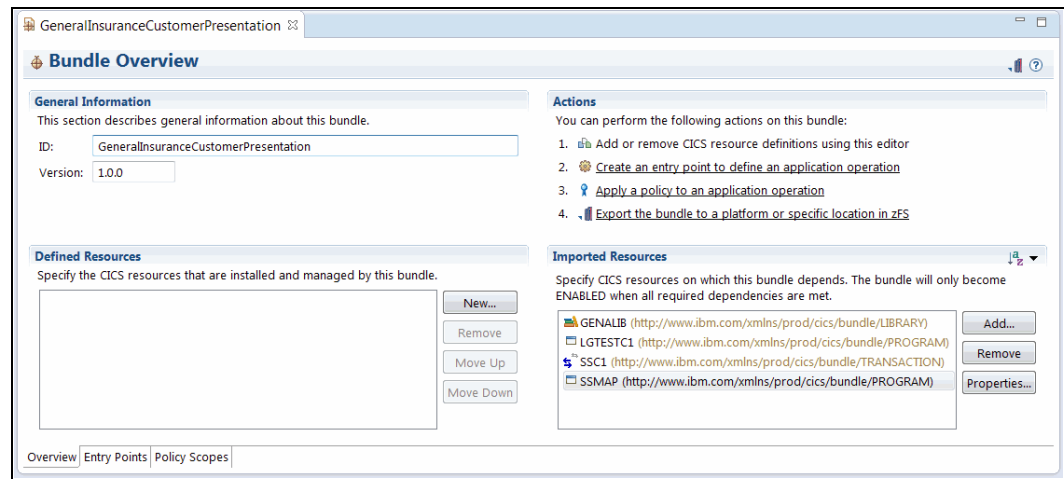


Figure 4-31 GeneralInsuranceCustomerPresentation CICS Bundle Manifest Editor

GeneralInsuranceCustomerData Bundle

The second CICS bundle represents the data layer of the GENAPP application. The GeneralInsuranceCustomerData Bundle has dependencies on the following CICS TS resources:

- ▶ LIBRARY GENALIB
- ▶ FILE KSDSCUST
- ▶ PROGRAM LGICDB01
- ▶ PROGRAM LGACDB01
- ▶ PROGRAM LGSTSQ

Tip: We could also add a dependency on a DB2ENTRY resource used by GENAPP. However, the name of that resource varies depending on the GENAPP installation options, so for simplicity it has been left out here.

Use the following steps to create the GeneralInsuranceCustomerData CICS Bundle Project:

1. Right-click the Project Explorer view, then select **New** → **Project**. The New Project wizard displays.
2. Expand CICS Resource, click **CICS Bundle Project**, then click **Next**. The CICS Bundle Project wizard displays.

Alternatively, to create a project to contain the files to be referenced by a CICS bundle definition, click the bundle plus (📁) icon in the toolbar to start the CICS Bundle Project wizard.

3. Enter `general.insurance.customer.data.bundle` as the Project name.
4. Enter `GeneralInsuranceCustomerData` as the ID.
5. Enter `1.0.0` in Version.

6. Figure 4-32 shows the completed Bundle Project wizard for the GeneralInsuranceCustomerData bundle. Click **Finish**. The bundle project is created and the CICS Bundle Manifest Editor appears.

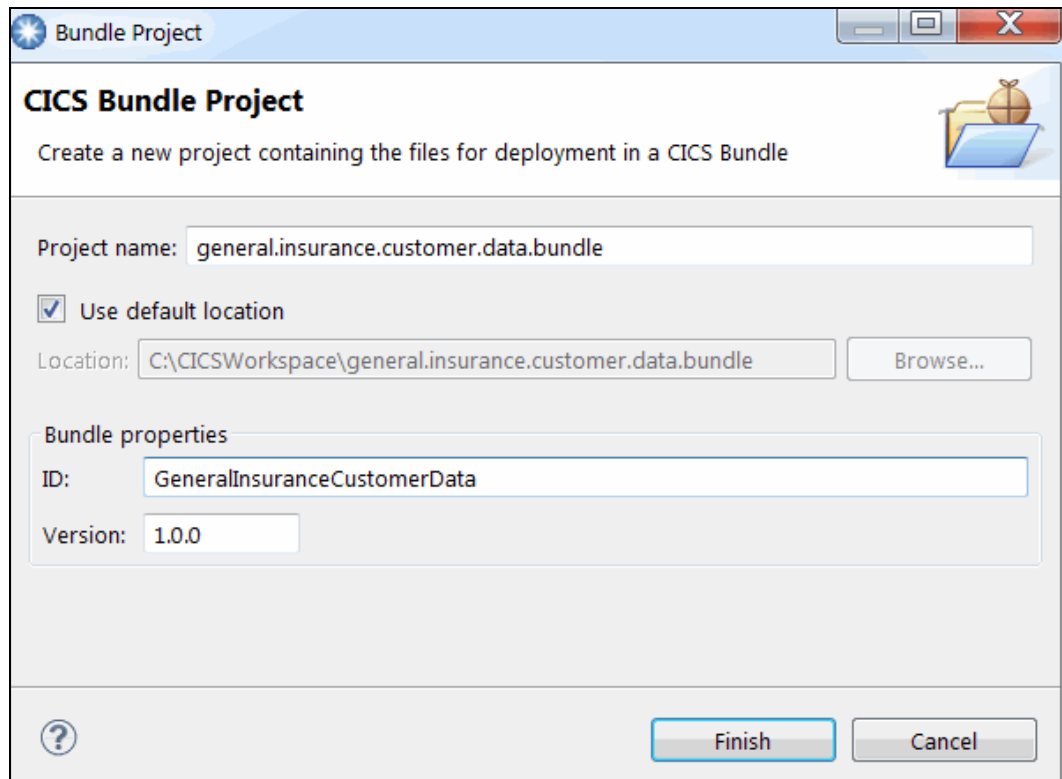


Figure 4-32 New CICS Bundle Project wizard for GeneralInsuranceCustomerData bundle

7. The CICS Bundle Manifest Editor will automatically start after the creation of the CICS Bundle Project in the previous step. If the editor is not open, locate the bundle manifest file to start the editor. In the Project Explorer view, expand the `general.insurance.customer.data.bundle` project and expand the META-INF subdirectory, then double-click the `cics.xml` manifest file to start the CICS Bundle Manifest Editor.
8. In Imported Resources, click **Add**. Select LIBRARY as the Resource Type, then enter GENALIB as the Name. Click **OK**.
9. In Imported Resources, click **Add**. Select FILE as the Resource Type, then enter KSDSCUST as the Name.

10. Figure 4-33 shows how the Create Bundle Import dialog should look. Click **OK**.

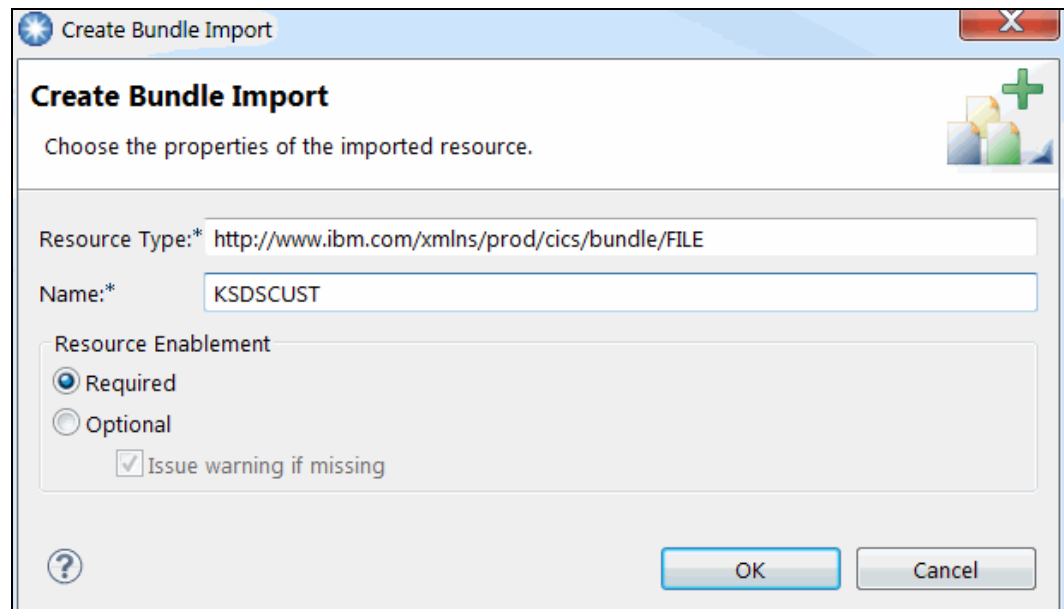


Figure 4-33 Create Bundle Import dialog for file KSDSCUST77

11. In Imported Resources, click **Add**. Select PROGRAM as the Resource Type, then enter LGICDB01 in Name. Click **OK**.
12. In Imported Resources, click **Add**. Select PROGRAM as the Resource Type, then enter LGACDB01 in Name. Click **OK**.
13. In Imported Resources, click **Add**. Select PROGRAM as the Resource Type, then enter LGSTSQ in Name. Click **OK**.
14. Press **Ctrl+S** to save changes.

The GeneralInsuranceCustomerData CICS Bundle Manifest Editor should look like Figure 4-34.

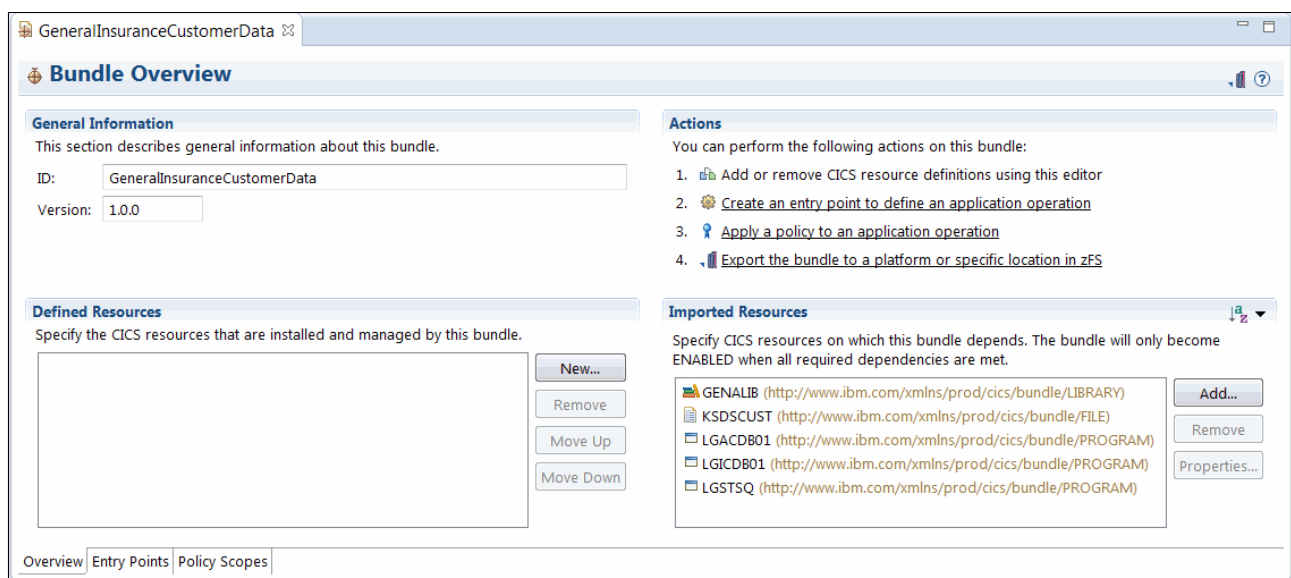


Figure 4-34 GeneralInsuranceCustomerData CICS Bundle Manifest Editor

GeneralInsuranceCustomerServices Bundle

We can also add dependencies into the GeneralInsuranceCustomerServices bundle that we created to contain the following application entry points:

- ▶ PROGRAM LGSTSQ
- ▶ PROGRAM LGICUS01
- ▶ PROGRAM LGACUS01
- ▶ LIBRARY GENALIB

Perform the following steps to add dependencies:

1. Locate the GeneralInsuranceCustomerServices bundle manifest file to start the editor. In the Project Explorer view, expand the general.insurance.customer.services.bundle project, expand the META-INF subdirectory, then double-click the cics.xml manifest file to start the CICS Bundle Manifest Editor.
2. Select the Overview tab.
3. In Imported Resources, click **Add**. Select PROGRAM as the Resource Type, then enter LGSTSQ as the Name. Click **OK**.
4. In Imported Resources, click **Add**. Select PROGRAM as the Resource Type, then enter LGICUS01 in Name. Click **OK**.
5. In Imported Resources, click **Add**. Select PROGRAM as the Resource Type, then enter LGACUS01 in Name. Click **OK**.
6. In Imported Resources, click **Add**. Select LIBRARY as the Resource Type, then enter GENALIB as the Name. Click **OK**.
7. Press **Ctrl+S** to save changes.

The GeneralInsuranceCustomerServices CICS Bundle Manifest Editor should look like Figure 4-35.

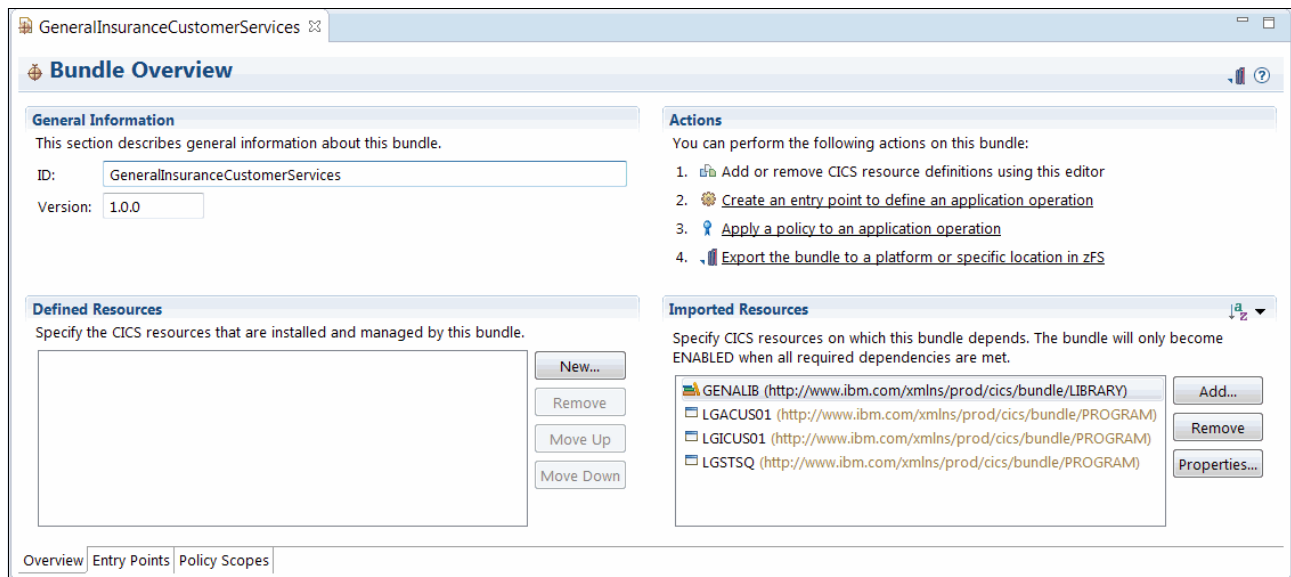


Figure 4-35 GeneralInsuranceCustomerServices CICS Bundle Manifest Editor

After following the steps in this section, you now have three CICS Bundle Projects in your workspace.

Your workspace should look similar to Figure 4-36. These CICS bundles represent the three logical layers of the GENAPP application:

- ▶ Presentation
- ▶ Application services
- ▶ Data manipulation and access

The GeneralInsuranceCustomerServices Bundle also creates two application entry points, so that the resource usage can be monitored with the use of application context data.

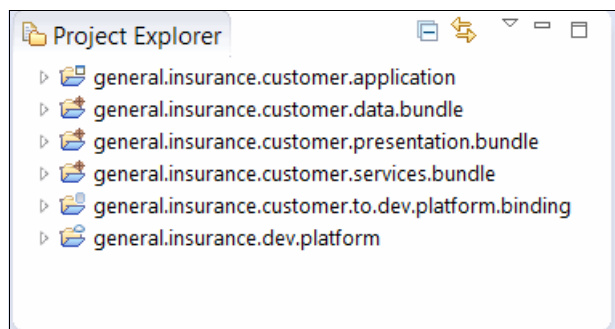


Figure 4-36 Project Explorer view after creating the three CICS Bundle Projects, along with the platform, application, and application binding projects

Chapter 6, “Packaging an application for multiversion deployment” on page 127 describes the reasons for and the methods to define CICS TS resources in the CICS bundles associated to your CICS TS applications and CICS TS platforms.

4.4.8 Updating the CICS Application Project

This section describes the methods involved in updating the CICS Application Project to include the two extra CICS bundles for the presentation and data access layers of the GENAPP application.

Use the following steps to update the GeneralInsuranceCustomer CICS Application Project using the CICS Cloud perspective in CICS Explorer:

1. Locate the GeneralInsuranceCustomer application manifest file to start the application editor. In the Project Explorer view, expand the `general.insurance.customer.application` project, expand the `META-INF` subdirectory, then double-click `application.xml` manifest file to start the CICS TS application editor.
2. In the CICS bundles section of the editor, click the **Add** button to add a new CICS bundle.

3. In the CICS Bundle Selection dialog, select bundle GeneralInsuranceCustomerData (1.0.0) as shown in Figure 4-37. Click **OK**.

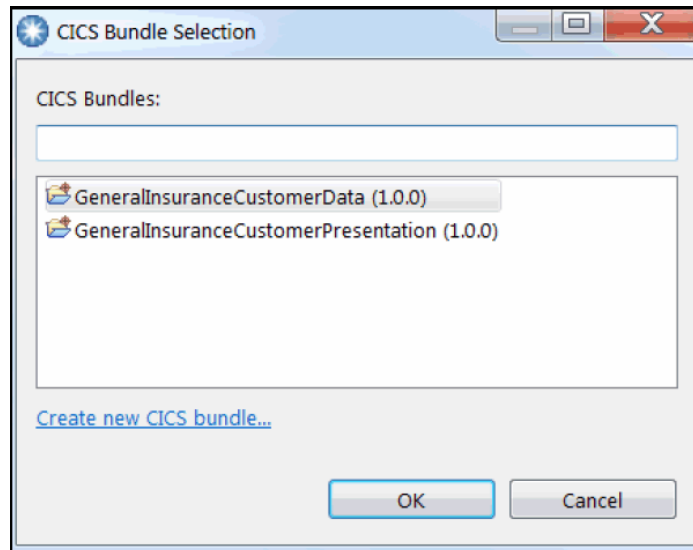


Figure 4-37 CICS Bundle Selection dialog to add bundle GeneralInsuranceCustomerData to the CICS TS application

Repeat this process to add the presentation layer CICS bundle.

4. In the CICS bundles section of the editor, click **Add** to add a new CICS bundle.
5. In the CICS Bundle Selection dialog, select the GeneralInsuranceCustomerPresentation (1.0.0) bundle, as shown in Figure 4-38.

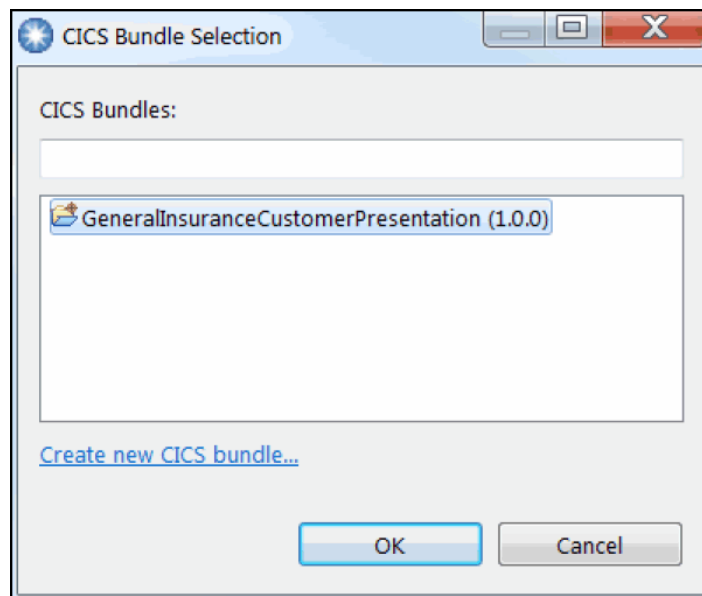


Figure 4-38 CICS Bundle Selection dialog to add the GeneralInsuranceCustomerPresentation bundle to the CICS TS application

6. Click **OK**. The application project is updated.
7. Press **Ctrl+S** to save changes.

Note: At this point, you might notice that the application binding for the GeneralInsuranceCustomer application is reporting an error. This is because the new CICS bundles that we have added to the application are not mapped to region types in the Application Binding. We will resolve this issue in the next section.

4.4.9 Update the CICS Application Binding Project

This section describes the methods to update the CICS Application Binding Project, so that the two new CICS bundles are mapped to the correct region types on the platform.

Use the following steps to update the `general.insurance.customer.to.dev.platform.binding` project using the CICS CloudCICS Cloud perspective in CICS Explorer:

1. Locate the `general.insurance.customer.to.dev.platform.binding` deployment file to start the application binding editor. In the Project Explorer view, expand project `general.insurance.customer.to.dev.platform.binding` and expand the META-INF subdirectory. Then double-click the `deployment.xml` file to start the CICS TS application binding editor.
2. In the Region Types section of the editor, select `DataAccess`. In the CICS bundles section, select `GeneralInsuranceCustomerData (1.0.0)`.
3. In the Region Types section of the editor, select `Presentation`. In the CICS bundles section, select `GeneralInsuranceCustomerPresentation (1.0.0)`.
4. Press **Ctrl+S** to save changes. Error messages resulting from the changes made in 4.4.8, “Updating the CICS Application Project” on page 84 should now disappear.

The application binding editor Deployment tab should look similar to Figure 4-39.

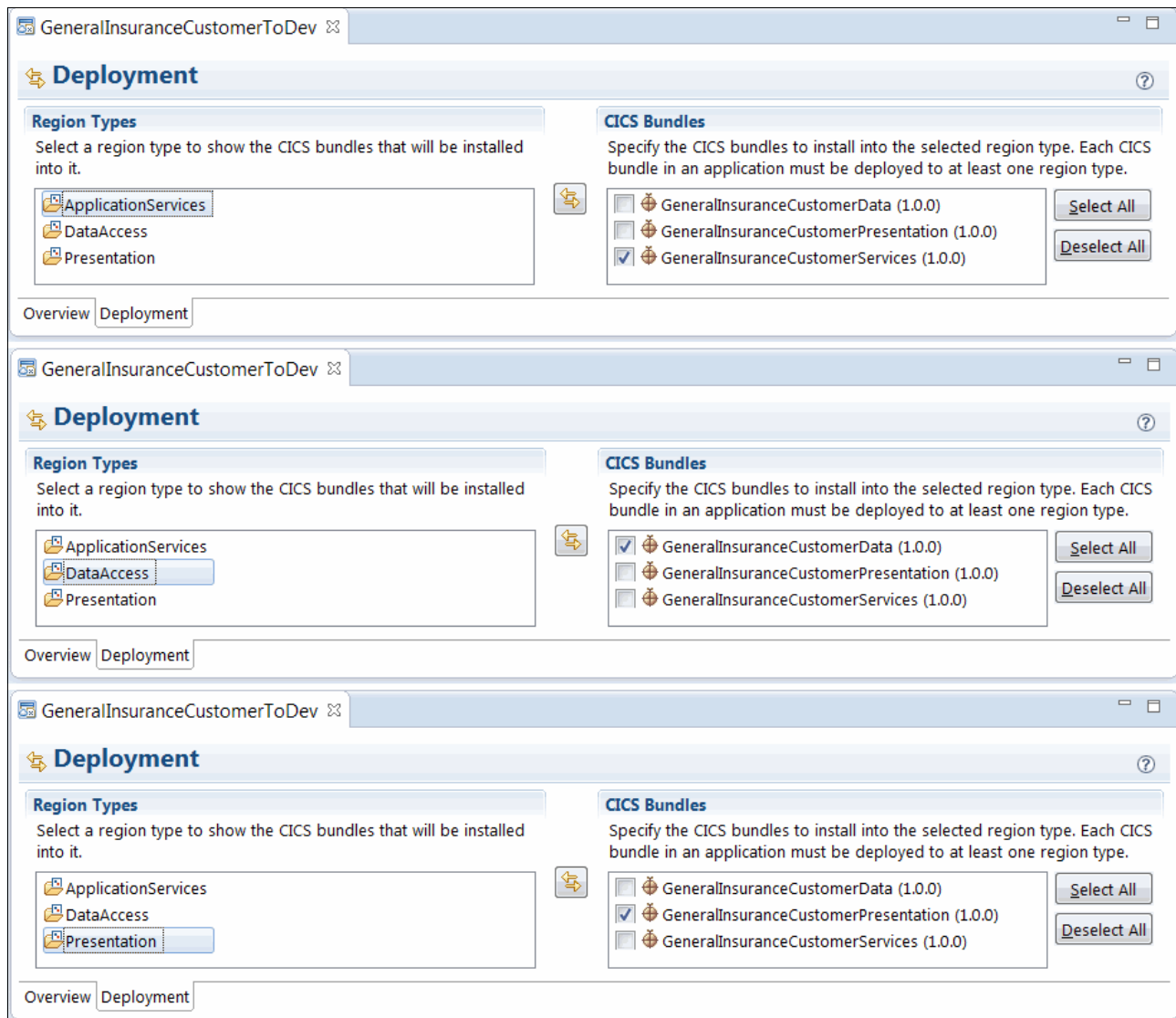


Figure 4-39 Three images of the GeneralInsuranceCustomerToDev application binding to show the CICS bundle deployment for each region type

The CICS Application Binding Project has now been updated. This project defines how the CICS TS application, GeneralInsuranceCustomer, is bound to the CICS TS platform, GeneralInsuranceDev.

4.4.10 Redeploying the CICS TS application

Before we can redeploy the CICS Application Project, CICS Application Binding Project, and all associated CICS Bundle Projects to the z/OS UNIX file system, we must first discard the installed GeneralInsuranceCustomer application from our GeneralInsuranceDev platform. This is because the CICS Explorer blocks any attempt to export files to the z/OS UNIX file system if they will overwrite files currently in use by an installed application.

Remember: We could have chosen to increase the version number of the GeneralInsuranceCustomer CICS TS application and the GeneralInsuranceCustomerServices bundle as we updated them.

This is a good practice for promoting application changes into production, and is a requirement for running multiple versions of the application simultaneously on the same CICS TS regions, which we describe in Chapter 6, “Packaging an application for multiversion deployment” on page 127.

However, while an application is under development, it is likely that the developer will want to make changes to the application without changing the version, until it is ready for promotion to test, and later into production.

Make unavailable, disable, and discard the installed application

Use the following steps to discard the CICS TS application installed in 4.4.6, “Installing and managing the CICS TS application” on page 73:

1. In the Cloud Explorer view, double-click the GeneralInsuranceCustomer (1.0.0) application. The Online Application editor for the GeneralInsuranceCustomer application displays.
2. On the Overview tab of the Online Application editor for the GeneralInsuranceCustomer application, in the General Information section, click **Make Unavailable**. The Perform UNAVAILABLE Operation dialog displays, as shown in Figure 4-40.

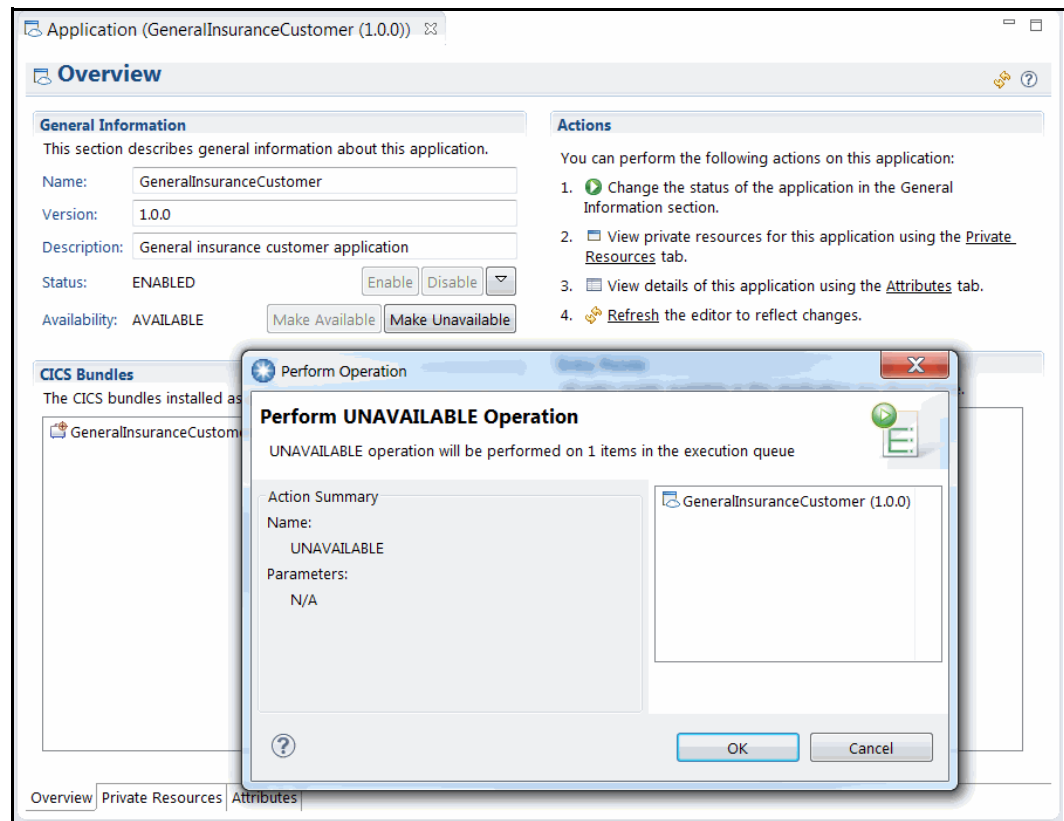


Figure 4-40 Using the Online Application Editor to make a CICS TS application unavailable

3. Click **OK**.

The application is now enabled in an unavailable state. The CICS TS programs named as entry points by the application are no longer entry points, so application-level monitoring and threshold policies will not be applied.

4. Access the Overview tab of the Online Application editor for the GeneralInsuranceCustomer application. In the General Information section, click **Disable**. The Perform DISABLE Operation dialog displays.
5. Click **OK**. The application will be disabled.
6. Access the Overview tab of the Online Application Editor for the GeneralInsuranceCustomer application. In the General Information section, expand the drop-down menu next to the **Disable** button and select **Discard**, as shown in Figure 4-41. The Perform DISCARD Operation dialog displays.

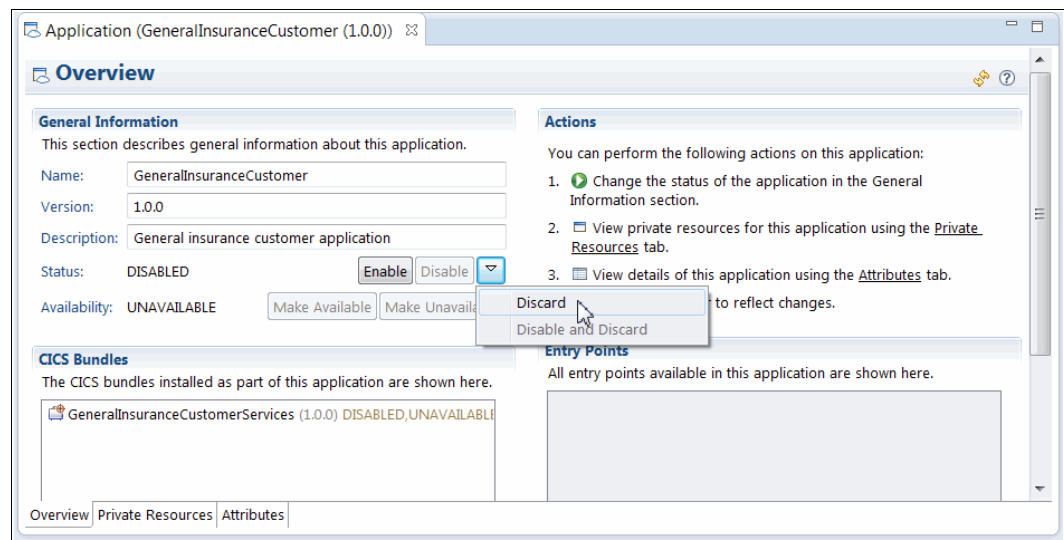


Figure 4-41 Drop-down menu to discard the application in the Online Application Editor

7. Click **OK**. The application is discarded.

Exporting the updated application

Use the following steps to export the updated CICS TS application, CICS application binding, and CICS Bundle Projects to the z/OS UNIX file system using the Application editor:

1. If the Application editor for the general.insurance.customer.application project is not open, locate the application files to start the Application editor.
In the Project Explorer view, expand the general.insurance.customer.application project, expand the META-INF subdirectory, and then double-click the application.xml file to start the Application editor.
2. In the Actions section of the Application editor, click **Export the application**.
The Export Application to the home directory of a Platform wizard displays. This wizard enables you to specify the CICSplex and CICS TS platform to which you want to deploy your CICS TS application. This action identifies the CICS application binding that will be used for deployment of the application.
3. If required, connect to the CICS TS environment's CICS Management Client Interface (CMCI) connection.

4. In GNAPPLEX, under GeneralInsuranceDev, select GeneralInsuranceCustomerToDev (1.0.0), as shown in Figure 4-42.

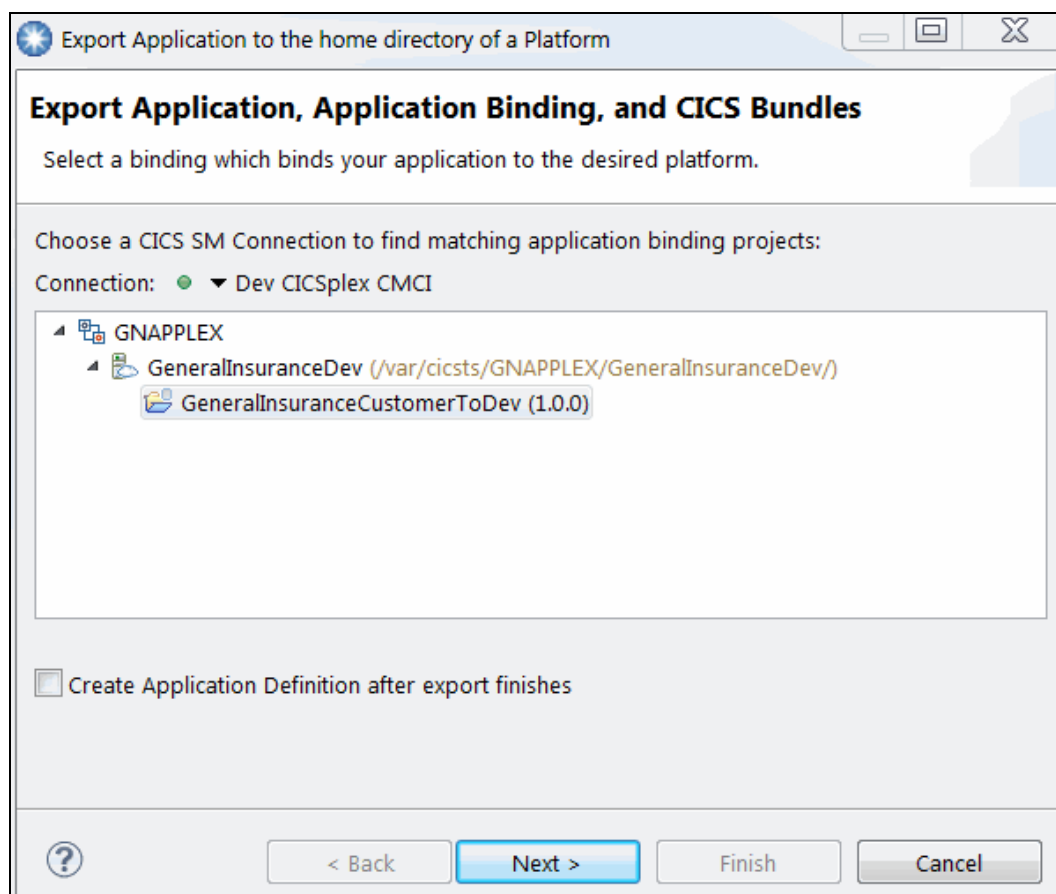


Figure 4-42 Export Application to the home directory of a platform

5. The application definition (APPLDEF) from the prior deployment can be reused. Therefore, leave **Create Application Definition after export finishes** cleared.
6. Click **Next**.
7. If required, connect to your CICS TS environment's z/OS File Transfer Protocol (FTP) connection. Click **Finish**.

The CICS Explorer begins to export the CICS TS application, CICS application binding, and CICS Bundle Projects to the z/OS UNIX file system. The projects are deployed to the platform home directory of the CICS TS platform in the applications, bindings, and bundles subfolders, where appropriate.

4.4.11 Installing and managing the updated CICS TS application

The CICS TS application, CICS application binding, and CICS Bundle Projects have now been redeployed to the z/OS UNIX file system. Use the following steps to install the GeneralInsuranceCustomer application using the CICS Cloud perspective in CICS Explorer:

1. In the Cloud Explorer view, expand the GeneralInsuranceDev platform and expand the Applications subdirectory. Right-click the GENACUST (1.0.0) CICS TS application definition and click **Install**, as shown in Figure 4-43 on page 91. The Perform INSTALL Operation dialog displays.

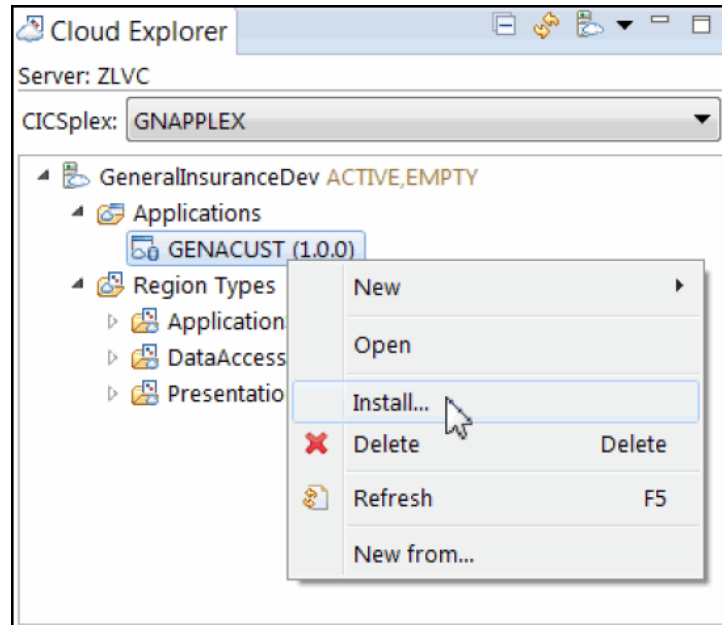


Figure 4-43 installing an application definition using the Cloud Explorer view

2. In the Perform INSTALL Operation dialog for the installation action, click **OK**.

The CICS TS application and all associated CICS bundles install into the CICS TS regions in the CICS TS platform. The GeneralInsuranceCustomer application is visible in the Cloud Explorer, as shown in Figure 4-44.

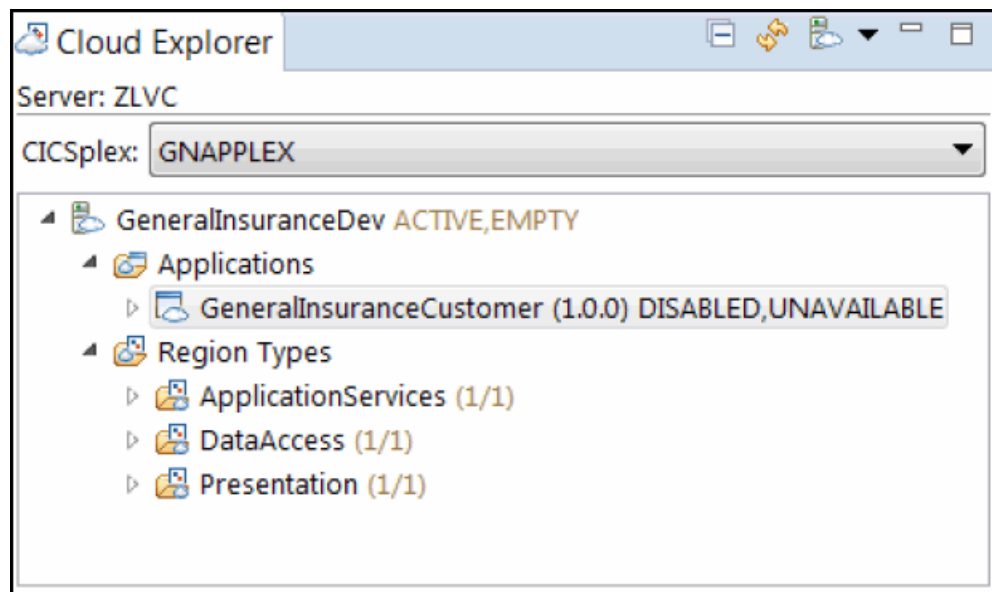


Figure 4-44 GeneralInsuranceCustomer visible in the Cloud Explorer after being installed

Tip: The installation action on the CICS TS application installs the CICS bundles that compose the application into the CICS TS regions in the platform region types, as defined by the deployment bindings. If the CICS TS application state does not immediately display as DISABLED, then wait for 15 seconds and refresh the Cloud Explorer view.

The CICS TS application installs all associated bundles into a disabled and unavailable state. Use the following steps to enable the GeneralInsuranceCustomer CICS TS application:

1. Double-click the GeneralInsuranceCustomer (1.0.0) application in the Cloud Explorer view. The Online Application editor for the GeneralInsuranceCustomer application displays, as shown in Figure 4-45.

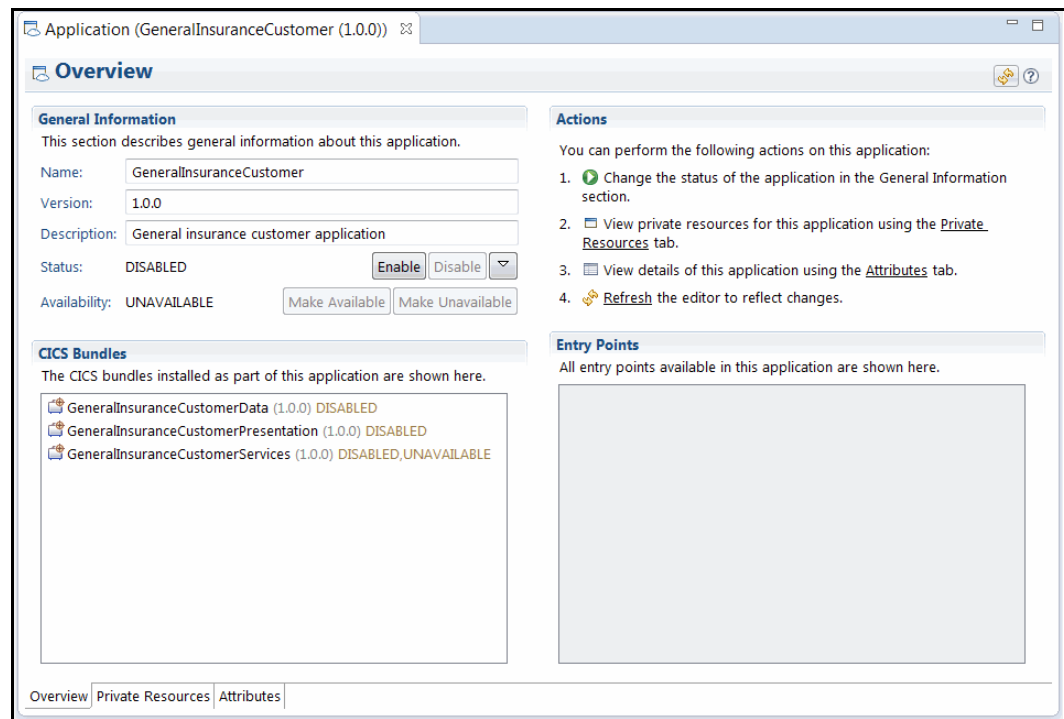


Figure 4-45 Online Application editor for GeneralInsuranceCustomer application

2. Access the Overview tab of the Online Application editor for the GeneralInsuranceCustomer application. In the General Information section, click **Enable**. The Perform ENABLE Operation dialog displays.
3. Click **OK**.

The CICS TS application will now attempt to become enabled. After a short amount of time, the status of the CICS TS application will reflect the status of the dependencies that were declared in the CICS bundles. Therefore, the status of the CICS TS application reflects the status of the GENAPP application in the CICS TS environment.

The application is now enabled in an unavailable state. The unavailable state means that the application's entry points are not yet applied.

Next, Make Available the GeneralInsuranceCustomer CICS TS application using the Online Application editor.

4. On the Overview tab of the Online Application editor for the GeneralInsuranceCustomer application, in the General Information section, click **Make Available**. The Perform AVAILABLE Operation dialog displays.
5. Click **OK**. The application is made available.

The GeneralInsuranceCustomer application is now available again for users to start through the identified application entry points. The application entry points set the application context on tasks, which enables monitoring of resource consumption for the application. In addition, the added dependencies mean that the applications enabled state reflects the state of the resources on which the application relies.

If the application shows a state of Enabled, all of the resources on which it has dependencies are installed and enabled. If the application shows a different state, or is unable to Enable, this could be because a resource on which it relies is not available, and further diagnosis might be required.

4.4.12 Diagnosing dependency errors

If your application does fail to enable, or at some point after enabling successfully, its state changes to SOMEDISABLED, or DISABLED, use the CICS Explorer to identify the cause of the error. In the example that follows, we first Made Unavailable and Disabled the GeneralInsuranceCustomer application. We then discarded the LGICUS01 CICS TS program from the CICS TS region on which the application is running, to force a dependency error.

To correct the error, perform the following steps:

1. Attempt to enable the GeneralInsuranceCustomer application again. The enable step highlighted that there was an error, and only some of the application's bundles were enabled. Figure 4-46 shows that the state of the GeneralInsuranceCustomer application is set to SOMEDISABLED.

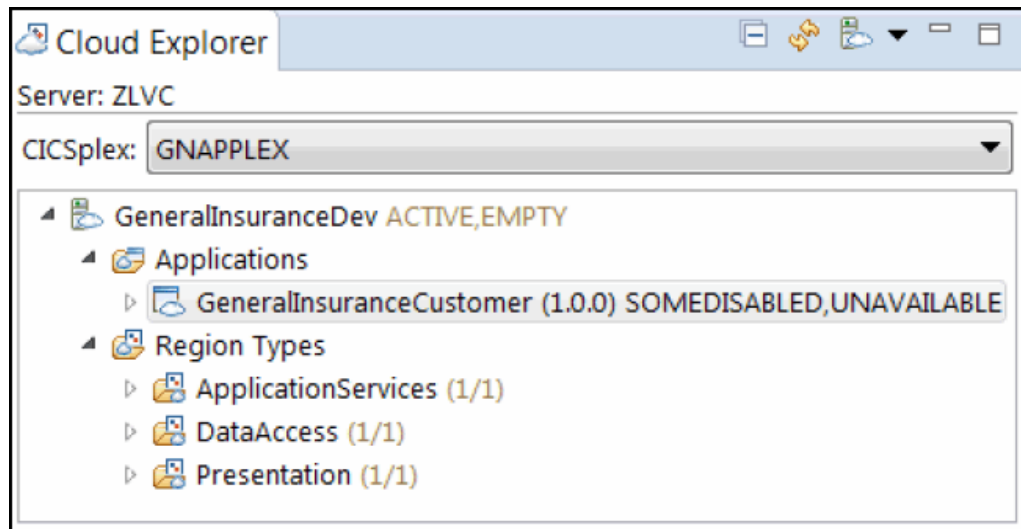


Figure 4-46 Cloud explorer view after enabling an application with a missing dependency

- One way to perform further diagnosis into the cause of the error, is to start from the Online Application Editor shown in Figure 4-47. In the CICS bundles section of the online editor, you can see the combined state of each bundle for all of the CICS TS regions that the bundle is deployed in. In this example, the GeneralInsuranceCustomerServices bundle appears to be in error. It has a state of SOMEDISABLED.

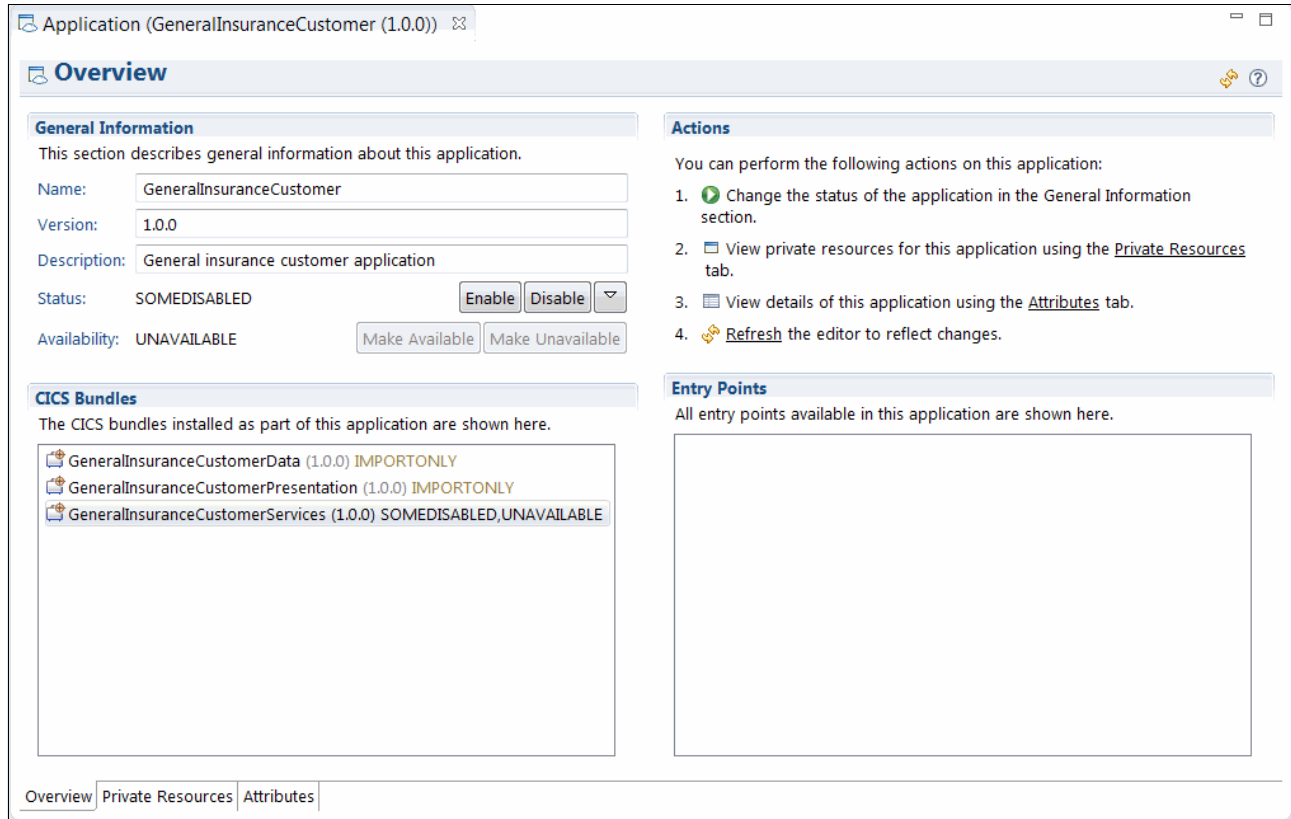


Figure 4-47 Application editor showing the GeneralInsuranceCustomerServices bundle has a state of SOMEDISABLED

- Double-clicking the name of the CICS bundle. A new Bundles view opens, showing the state of the Bundle on each CICS TS region where it is installed. In this example, the new view is called *Bundles for Management Part 'GeneralInsuranceCustomerServices (1.0.0)'*. Because the bundle is only installed into a region type consisting of a single CICS TS region, there is only one row in the returned view, as shown in Figure 4-48. The view shows the APPLID of the CICS TS region in which the bundle is disabled.

The screenshot shows the 'Bundles for Management Part 'GeneralInsuranceCustomerServices (1.0.0)'' view. The context is 'CNX0211I Context: GENADEVA. Resource: BUNDLE. 1 (filtered) records collected at 22 Sep 2014 14:53:02'. The table below shows the bundle's status across different regions.

Region	Name	Bundle ID	Major Version	Minor Version	Micro Version	Status	Availability
IVK2ZLVG	\$0667181	GeneralInsuranceCustomerServices	1	0	0	DISABLED	UNAVAILABLE

Figure 4-48 Bundles view for GeneralInsuranceCustomerServices (1.0.0)

4. In the new Bundles view, right-click one of the Bundles and select Show Bundle Parts to show the status of each resource associated with the Bundle on a particular CICS TS region, and identify which resources are not in an Enabled state.

In our example, it opens a new view called *Bundle Parts for Bundle 'GeneralInsuranceCustomerServices (1.0.0)'*. We can see that both the import for LGICUS01 and the entry point for LGICUS01 are disabled, as shown in Figure 4-49.

Region	Bundle	Bundle Part	Enable Status	Availability	Part Class	Part Type
IYK2ZLVG	\$0667181	GENALIB	✓ ENABLED	NONE	IMPORT	http://www.ibm.com/xmlns/prod/cics/bundle/LIBRARY
IYK2ZLVG	\$0667181	LGACUS01	✓ ENABLED	NONE	IMPORT	http://www.ibm.com/xmlns/prod/cics/bundle/PROGRAM
IYK2ZLVG	\$0667181	LGACUS01	✓ ENABLED	UNAVAILABLE	ENTRYPOINT	http://www.ibm.com/xmlns/prod/cics/bundle/PROGRAM
IYK2ZLVG	\$0667181	LGICUS01	✗ DISABLED	NONE	IMPORT	http://www.ibm.com/xmlns/prod/cics/bundle/PROGRAM
IYK2ZLVG	\$0667181	LGICUS01	✗ DISABLED	UNAVAILABLE	ENTRYPOINT	http://www.ibm.com/xmlns/prod/cics/bundle/PROGRAM
IYK2ZLVG	\$0667181	LGSTSQ	✓ ENABLED	NONE	IMPORT	http://www.ibm.com/xmlns/prod/cics/bundle/PROGRAM

Figure 4-49 Bundle Parts for GeneralInsuranceCustomerServices (1.0.0)

The dependencies have highlighted that a component of the application has a problem. We have demonstrated how to determine the resources that are in error, and the CICS TS region on which those resources are in error. Finally, we could examine the MSGUSR output from the CICS TS region for error messages to help us understand why the problem occurred. In this example, some useful error messages are emitted:

- ▶ DFHRL0126 The import of resource LGICUS01 of type <http://www.ibm.com/xmlns/prod/cics/bundle/PROGRAM> for BUNDLE resource \$0667181 has changed to a DISABLED state.
- ▶ DFHPG0306 BUNDLE \$0667181 unable to enable PROGRAM LGICUS01 as an entry point for operation inquireCustomer because the PROGRAM does not exist.

4.5 The outcome

In this chapter, you have deployed a CICS TS application (GeneralInsuranceCustomer) based on an existing active application (GENAPP), as described in Chapter 2, “GENAPP introduction” on page 15.

Two operations of the application, addCustomer and inquireCustomer, were identified. Corresponding program resources in the application were identified as the application entry points for these operations, and declared in the application.

Using the CICS Explorer, you created a CICS bundle to contain the application entry points. A CICS TS application, which included the CICS bundle, was then created. An application binding was created to describe the deployment rules for the mapping of the CICS bundle to the correct region type of the GeneralInsuranceDev platform created in Chapter 3, “Creating a platform” on page 25.

After the application-related projects are created, the application, binding, and CICS bundles are managed as a single unit. The application was exported to the z/OS UNIX file system, and an application definition was created for the application. As a single point of control, the application was installed, enabled, and made available to users. These actions are controlled by CICS TS to provision resources across the CICS TS platform.

Two additional CICS bundles were then created to express dependencies on CICS TS resources used by GENAPP, and to reflect their state into the overall CICS TS application state. The CICS TS application was updated, as was the application binding. The installed application was discarded, and the updated application was redeployed and reinstalled.

You can create more application bindings to deploy the application to other platforms without modifying the application resources. By doing this, you can confidently promote an application through development, test, and production platforms.

The CICS TS application supports a single point of control for the resources in the application. The application entry points add application context data to tasks that run in CICS TS. Application context data is available in monitoring records to provide a way of measuring how much system resources an application is using. The application context data also supports the delivery of automated control when resource usage exceeds defined thresholds, which is further discussed in Chapter 5, “Applying a policy” on page 97.



Applying a policy

This chapter introduces IBM Customer Information Control System (CICS) policies. It explains the benefits and capabilities that policies can provide to IBM CICS Transaction Server (CICS TS) platforms and CICS TS applications. By using working examples, this chapter guides you through the creation, deployment, and effects of adding policies into a CICS TS cloud environment.

For further information and design considerations for CICS Policy, see Chapter 8, “Managing by policy” on page 181.

The following list describes reasons that a user would create a CICS Policy:

- ▶ Protect the CICS TS platform from erroneous applications.
- ▶ Identify early in the development lifecycle if an application meets its service level agreements (SLAs).
- ▶ Enforce rules that applications must adhere to at run time.

This chapter contains the following topics:

- ▶ 5.1, “CICS Policy” on page 98
- ▶ 5.2, “Example CICS policies” on page 100
- ▶ 5.3, “Results of adding a policy” on page 126

5.1 CICS Policy

Policies support the management of operations by the control of critical resource thresholds. You can deploy policies to monitor the resource usage of user tasks, and to deliver automated control when resource usage exceeds defined thresholds.

Policy definitions can include one or more policy rules. Each policy rule consists of a defined threshold and an associated action that is taken if the threshold is breached, as shown in Figure 5-1.

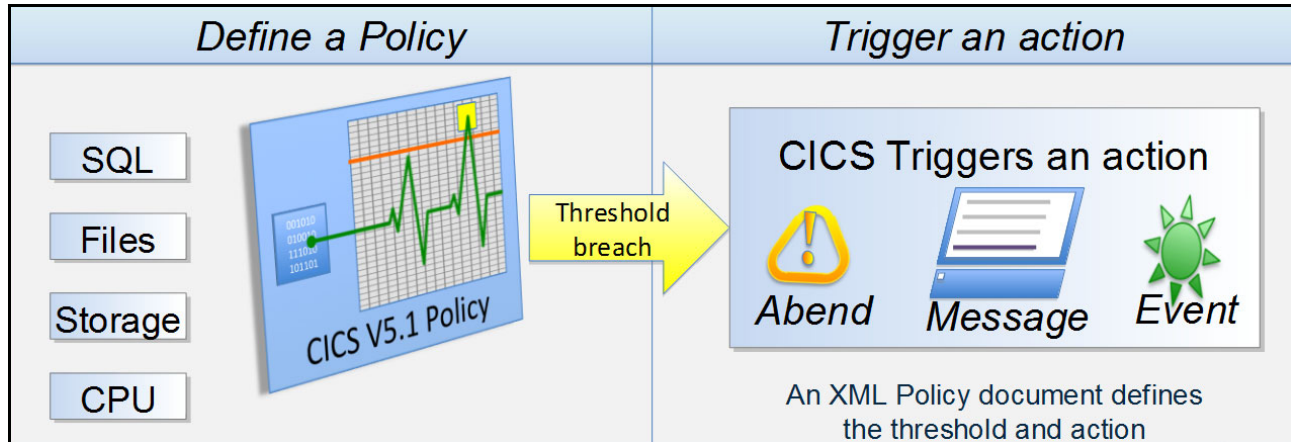


Figure 5-1 A breach of a CICS TS policy triggers an action to be taken

You can create, deploy, and search for deployed CICS policies using the Cloud Explorer view in the CICS Explorer or CICS Explorer software development kit (SDK).

5.1.1 Policy rule thresholds

Policy rules define the threshold conditions for which an action can be taken. Policy rules were introduced in CICS TS V5.1 and extended in CICS TS V5.2. In CICS TS V5.2, you can define threshold conditions to limit the following types of resources:

- ▶ Databases
 - The number of database requests (Structured Query Language (SQL) commands)
- ▶ Files
 - The number of file access requests, such as Read, Read update, Write, Rewrite, Delete, Start browse, Read next, or Read previous requests
- ▶ Programs
 - The number of program link requests
- ▶ Start requests
 - The number of start requests
- ▶ Storage
 - The amount of task or user storage below the line (24-bit), above the line (31-bit), or above the bar (64-bit)
 - The number of requests for task or user storage below the line (24-bit), above the line (31-bit), or above the bar (64-bit)

- ▶ Sync points
 - The number of sync point or sync point rollback requests
- ▶ Time
 - The amount of processor time that is used by a task
 - The elapsed time on a user task
- ▶ Temporary Storage
 - The number of temporary storage requests, such as ReadQ or WriteQ requests
 - The amount of data written to temporary storage
- ▶ Transient Data
 - The number of transient data requests, such as ReadQ or WriteQ requests

5.1.2 Policy rule actions

Policy rules can perform the following actions when a threshold is exceeded:

- ▶ Issue message DFHMP3001, which is the default policy action.
- ▶ End the task by issuing either the default abnormal end (abend) code AMPB, or a user-specified abend code.
- ▶ Emit an event.

5.1.3 Policy rule scope

Policy definitions are created using the CICS Explorer, and are defined within CICS bundles. These CICS bundles are then deployed to a CICS TS cloud environment by one of two deployments:

- ▶ Deployed as part of a CICS TS platform or added into a deployed platform
- ▶ Deployed along with a CICS TS application

The method of deployment used affects the scope of the policy.

Policies deployed into CICS TS platforms

CICS bundles containing policy can be deployed as part of a CICS TS platform. This deployment is achieved by including the CICS bundle in the list of bundles to be deployed along with the platform during the creation of the platform project. Alternatively, a CICS bundle containing policy can be deployed into an installed platform using the **ADDBUNDLE** function.

Policies deployed in a platform apply to all workloads where the application context of the running user tasks specifies the named platform. Therefore, platform policies will scope to all applications deployed to a specific platform. An example of a policy being deployed into an installed platform can be seen in section 5.2.1, “Platform policy example” on page 100.

Policies deployed into CICS TS applications

CICS bundles containing policies can be deployed as part of a CICS TS application. This is achieved by including, in either the application or application binding project, the CICS bundle in the list of bundles to be deployed along with the application. A CICS bundle cannot be installed dynamically into an installed application.

Important: One of the premises of a CICS TS application is that it remains unchanged throughout its promotion from the development platform, through various test platforms, and into one or more production platforms. Therefore, when adding a policy to a CICS TS application, there is a choice of whether to define it within the CICS TS application or within the CICS application binding.

The decision is based on whether the policy remains the same in all of the different platforms on which it runs. If it remains the same, define it in the application. If it needs to change between different platforms, define it in the application binding.

Policies deployed with an application apply to all workloads where the application context of the running user tasks matches the deployed application. For example, the application name, application version, and platform name all match. Therefore, application policies will scope to all operations for a specific version of an application deployed to a specific platform.

Application policies can be scoped to a specific application operation. This scoping is achieved by including a *Policy Scope* definition in the CICS bundle. Application operation policies apply to a specific operation for a specific application deployed to a specific platform.

An example of an application operation policy can be seen in section 5.2.2, “Application operation policy example” on page 109.

5.2 Example CICS policies

You have been introduced to CICS policies. This section provides information about the creation, deployment, and threshold triggering of two example policies in the context of the GENAPP application.

The following list describes the results of completing the steps in this section:

- ▶ An active policy in the CICS TS platform that provides automated control and protection against the use of 24 bit storage by CICS TS applications.
- ▶ An active policy to monitor resource usage for a particular operation.
- ▶ The ability to search active policies for a CICS TS platform, CICS TS application, and an application operation.

5.2.1 Platform policy example

This section describes the steps required to create and deploy a platform-wide policy. A policy is created that prevents any application within the GeneralInsuranceDev platform from using 24-bit task storage. If an application in the platform attempts to use 24-bit storage, the policy forces an abend of the task.

The following list describes the steps in this section:

- ▶ Create and define the policy.
- ▶ Export the policy project to the platform.
- ▶ Deploy the policy into an active platform.
- ▶ View the outcome of exceeding the policy threshold.

Prerequisites

The prerequisites include the following software setup:

- ▶ CICS TS regions running V5.1 or later. The walkthrough is done using CICS TS V5.2.
- ▶ CICS Explorer V5.2 or later, or CICS Explorer V5.2 SDK or later.

The platform policy is deployed into the active platform GeneralInsuranceDev created in Chapter 3, “Creating a platform” on page 25.

The creation and deployment of the platform policy example uses the CICS Explorer CICS Cloud perspective.

Create and define the policy

Perform the following steps to create the GeneralInsurancePlatformPolicy CICS Bundle Project, and define the policy, using the CICS Cloud perspective in CICS Explorer:

1. Right-click in the Project Explorer view and select **New** → **Project**. The New Project wizard displays.
2. In the New Projects wizard expand CICS Resources, click **CICS Bundle Project**, then click **Next**. The CICS Bundle Project wizard displays.

Alternatively, to create a project to contain the files to be referenced by a CICS bundle definition, click the bundle plus (📁) icon in the toolbar to start the new CICS Bundle Project wizard.

3. Enter `general.insurance.platform.policy` as the Project name.
4. Enter `GeneralInsurancePlatformPolicy` as the ID.
5. Enter `1.0.0` as the Version.

Figure 5-2 shows the completed CICS Bundle Project wizard.

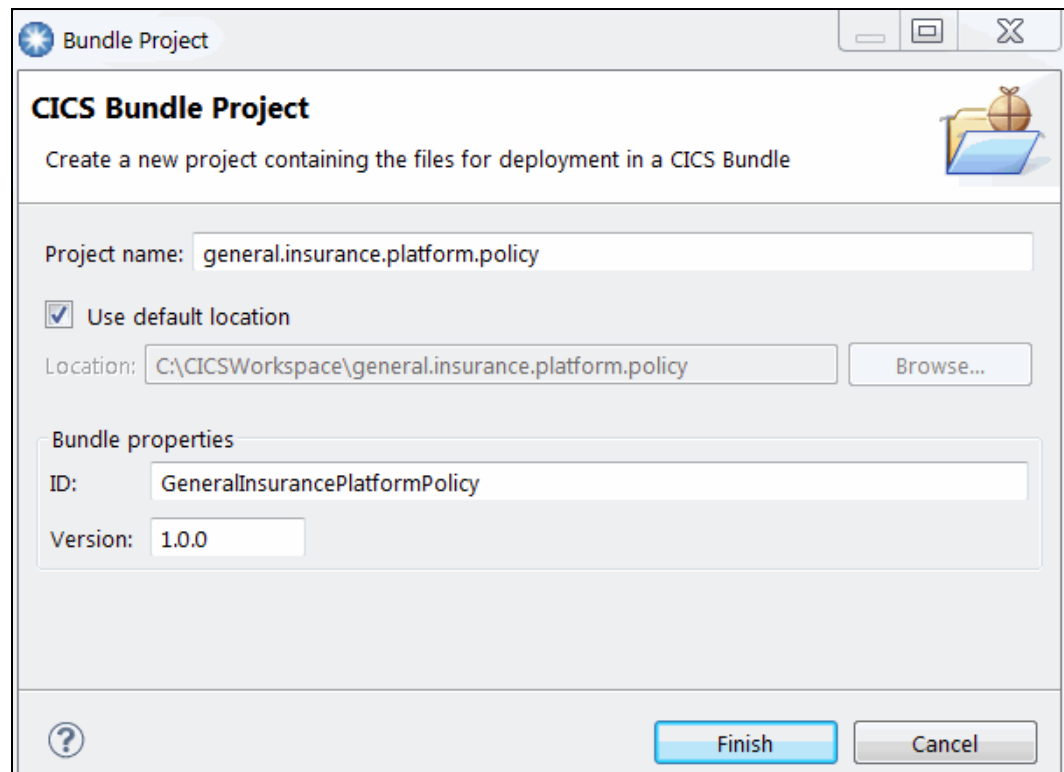


Figure 5-2 Create GeneralInsurancePlatformPolicy Bundle Project wizard

6. Click **Finish**. The bundle project is created and the CICS Bundle Manifest Editor displays.
The GeneralInsurancePlatformPolicy CICS Bundle Project has been created. Now create the policy definition in the CICS bundle using the CICS Bundle Manifest Editor.
7. The CICS Bundle Manifest Editor will automatically start after the creation of the CICS Bundle Project in the previous step. If the editor is not open, locate the bundle manifest file to start the editor. In the Project Explorer view, expand the `general.insurance.platform.policy` project, expand the META-INF subdirectory, and then double-click the `cics.xml` manifest file to start the CICS Bundle Manifest Editor.
8. On the Overview tab in the CICS Bundle Manifest Editor, for the GeneralInsurancePlatformPolicy bundle, click **New** in the Defined Resources section.
9. Select **Policy Definition** to open the Create Policy Definition wizard.

Figure 5-3 illustrates how to start the Create Policy Definition wizard from the bundle manifest editor.

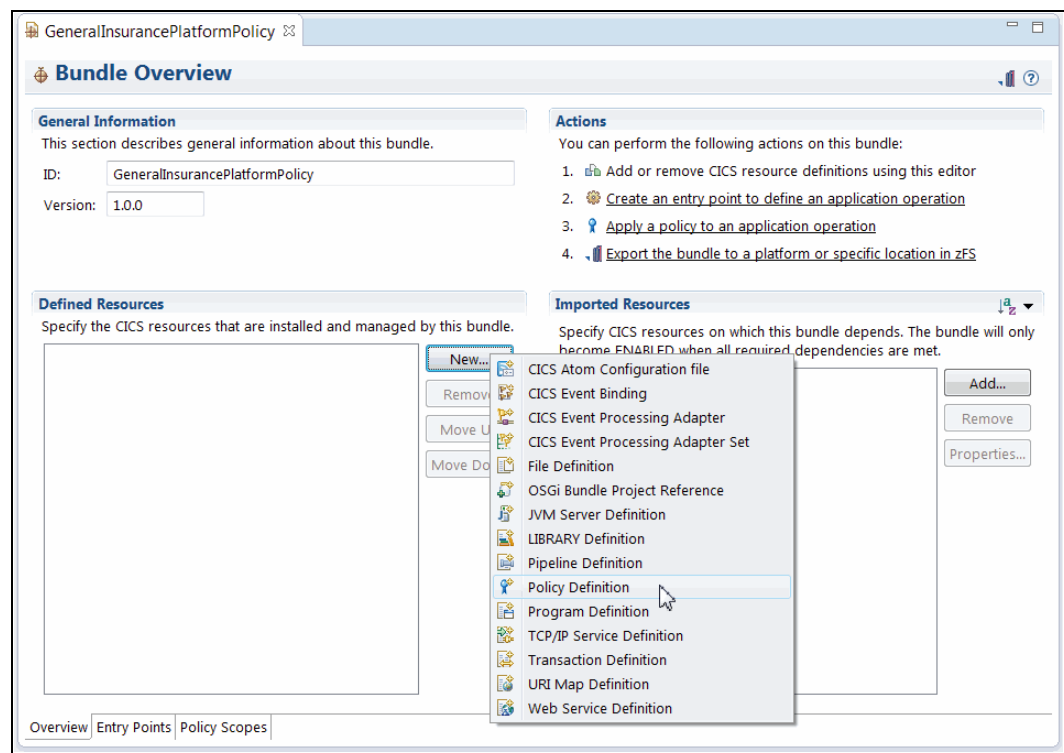


Figure 5-3 Starting the Create Policy Definition wizard from the CICS Bundle Manifest Editor

10. Enter GeneralInsurancePlatformPolicies as the Policy name.
11. Enter Policies for the General Insurance platform as the Policy description.
12. Enter Prevent24bitStorage as the Rule name.
13. Enter Prevent 24 bit Storage use as the Rule description.
14. Select the threshold Type **Storage**.
15. Select the threshold Item **24-bit Task storage**.
16. Select threshold Operator **Greater Than**, enter Value 0, and select Unit **Bytes**.
17. Select the policy to **Abend task with abend code AMPB**.

18. Clear **Open Editor**.

Figure 5-4 shows the completed Create Policy Definition wizard.

The screenshot shows the 'Create Policy Definition' wizard window. The title bar says 'Create Policy Definition'. The main heading is 'Add a rule' with a subtext: 'Add a policy rule to define what action is taken when a task exceeds a specified condition.' There is a ribbon icon in the top right corner.

Policy Information

Name:* GeneralInsurancePlatformPolicies (is also the bundle part name)
Description: Policies for the General Insurance platform

Rule Information

Name:* Prevent24bitStorage
Description: Prevent 24 bit Storage use

What is the condition that triggers the rule?

Type: Database request, File request, Program request, Start request, **Storage**
Item: **24-bit Task storage**, 31-bit Task storage, 64-bit Task storage, 24-bit Shared storage, 31-bit Shared storage

Operator: Greater Than Value:* 0 Unit: Bytes

What action should be taken when the rule's condition is exceeded?

☐ Issue message DFHMP3001
☐ Emit event to
☒ EP Adapter ☐ EP Adapter Set
☒ Abend task with abend code AMPB

☐ Open Editor

Buttons: ? < Back Next > Finish Cancel

Figure 5-4 Create GeneralInsurancePlatformPolicy wizard

19. Click **Finish**. The policy is defined in the CICS bundle.

You have now created the GeneralInsurancePlatformPolicy CICS bundle, and defined the GeneralInsurancePlatformPolicies policy.

Export the policy project to the platform

Perform the following steps to export the CICS bundle containing the example policy to the GeneralInsuranceDev platform home directory using the CICS Bundle Manifest Editor:

1. If the CICS Bundle Manifest Editor for the `general.insurance.platform.policy` project is not open, locate the bundle manifest file to start the editor. In the Project Explorer view, expand the `general.insurance.platform.policy` project, expand the `META-INF` subdirectory, and then double-click the `cics.xml` manifest file to start the CICS Bundle Manifest Editor.
2. On the Overview tab in the CICS Bundle Manifest Editor, for the `GeneralInsurancePlatformPolicy` bundle, click **Export the bundle to a platform or specific location in zFS** in the Actions section. The Export to z/OS UNIX File System wizard opens.
3. In the Export to z/OS UNIX File System wizard, select **Export to the home directory of a Platform**, as shown in Figure 5-5.

Remember: The user is not asked for a specific location on the z/OS UNIX file system to deploy the CICS bundle. This location has already been established by the CICS TS platform and the platform home directory.

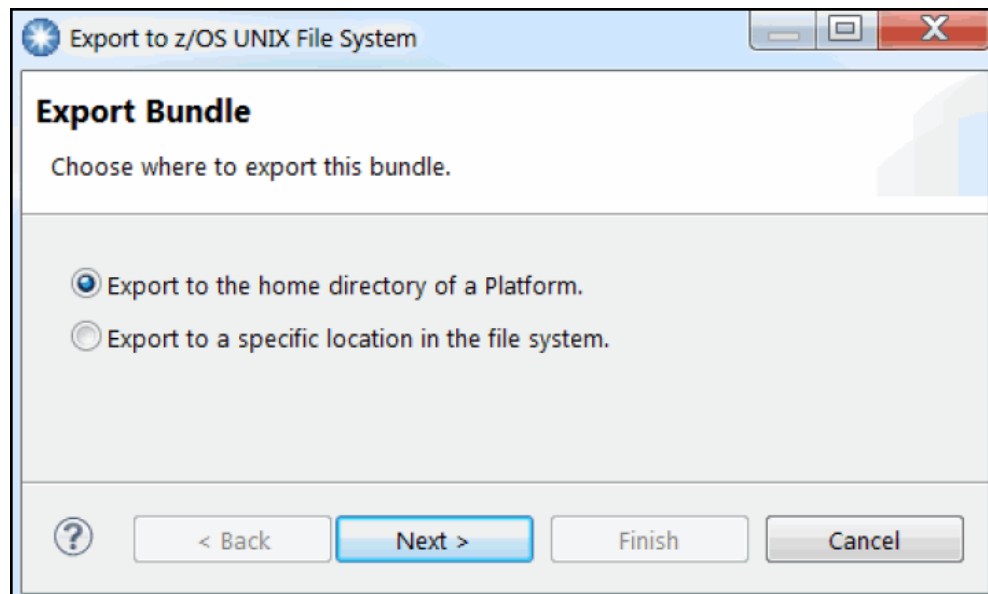


Figure 5-5 Export the CICS bundle to the home directory of a platform on the z/OS UNIX file system

4. Click **Next**.
5. If required, connect to your CICS TS environment's CICS Management Client Interface (CMCI) connection. Select the GeneralInsuranceDev platform.

Tip: An active connection is required to discover the platform home directories for installed platforms. Only installed platforms are listed. If the GeneralInsuranceDev platform is not listed, then ensure that the correct connection is connected, and that the platform is installed.

Figure 5-6 shows the completed page of the Export to z/OS UNIX File System wizard.

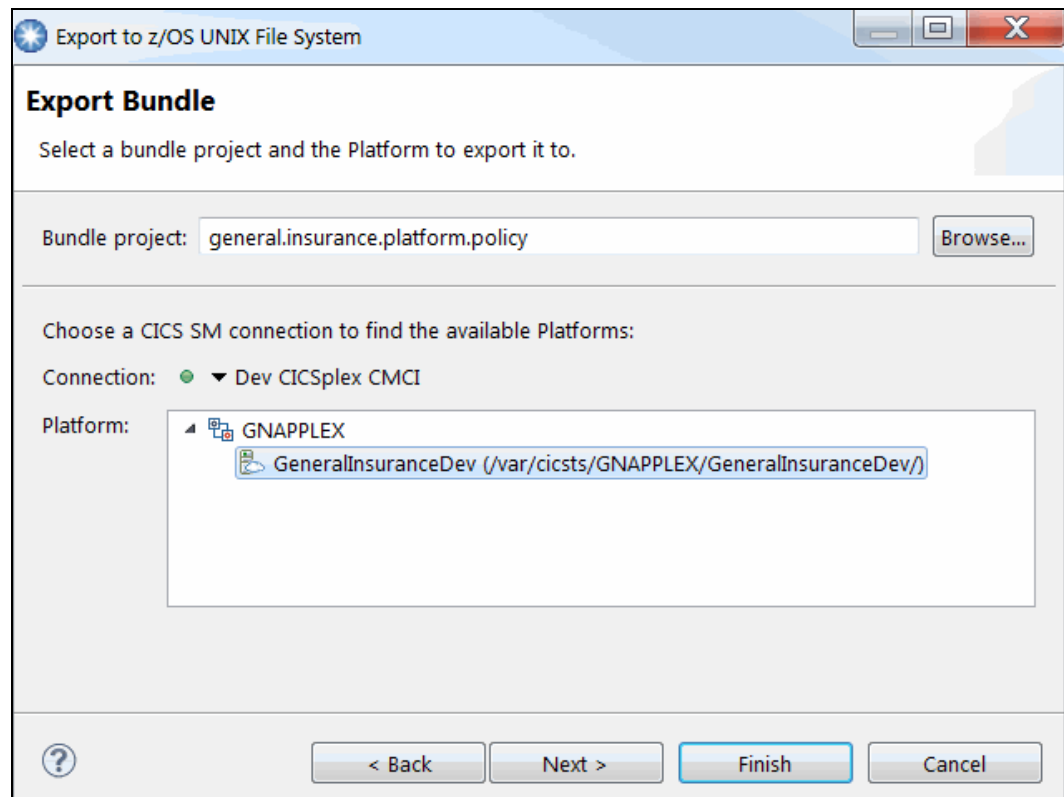


Figure 5-6 Export Bundle project *general.insurance.platform.policy*

6. Click **Next**.
7. If required, connect to your CICS TS Environment's z/OS File Transfer Protocol (FTP) connection. Click **Finish**.

The CICS bundle containing the policy is exported to the file system.

Deploy the policy into an active platform

Perform the following steps to install the exported `GeneralInsurancePlatformPolicy` CICS bundle into the installed platform.

1. In the Cloud Explorer view, expand the `GeneralInsuranceDev` platform and expand the subdirectory `Region Types`. You should see the region types that compose the `GeneralInsuranceDev` platform listed.
2. Right-click the **ApplicationServices** region type.

3. Select **Add Bundle**, as shown in Figure 5-7. The Perform ADDBUNDLE Operation dialog displays.

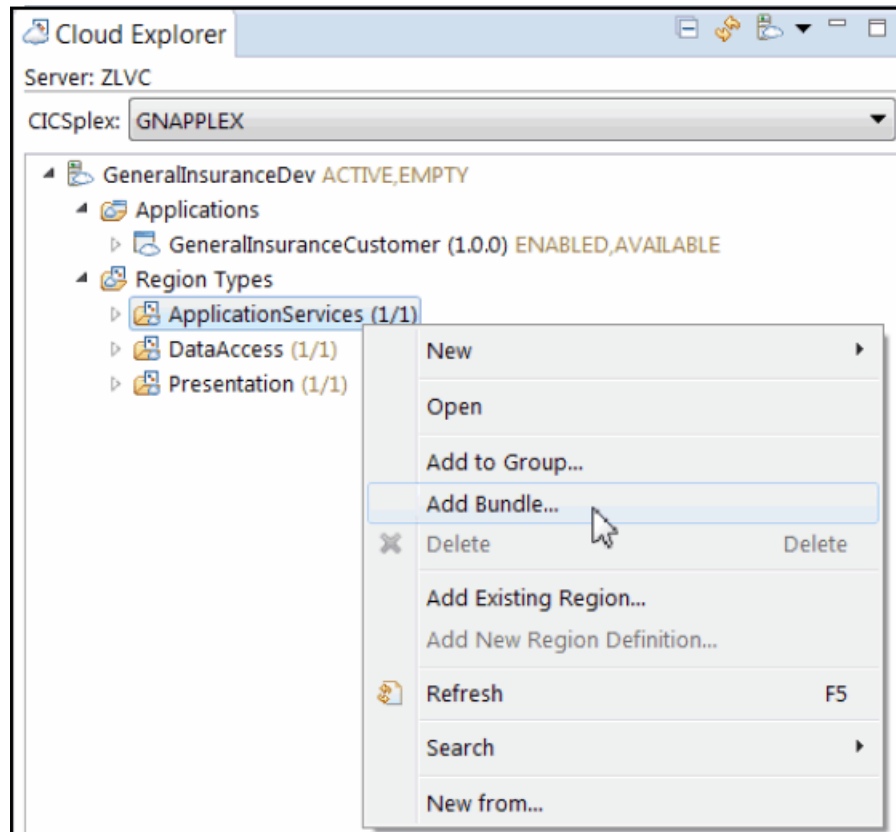


Figure 5-7 Using the Cloud Explorer view to add a CICS bundle to an installed platform

4. In the Perform ADDBUNDLE Operation dialog, ensure that the ApplicationServices RegionType is selected in the drop-down list.

5. Select the GeneralInsurancePlatformPolicy_1.0.0 Platform bundle.

Figure 5-8 shows the completed Perform AADBUNDLE operation dialog to add the GeneralInsurancePlatformPolicy CICS bundle to the GeneralInsuranceDev platform.

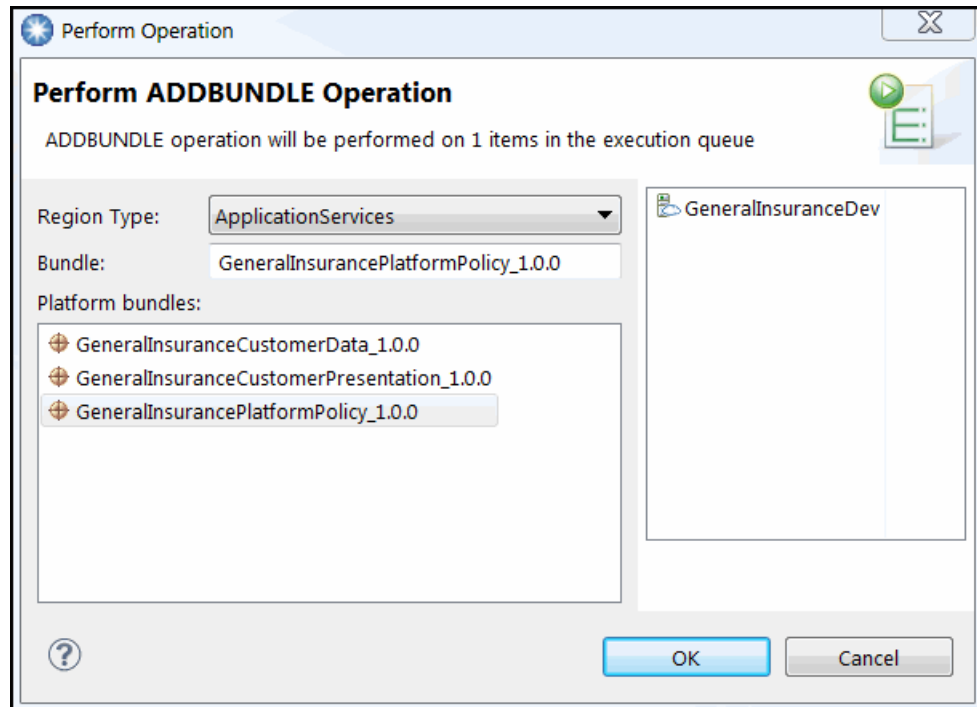


Figure 5-8 Perform AADBUNDLE Operation action to add a CICS bundle into an installed platform

6. Click **OK**.

The GeneralInsurancePlatformPolicy CICS bundle is deployed to all CICS TS regions in the ApplicationServices region type for the GeneralInsuranceDev platform.

Tip: The AADBUNDLE operation attempts to set the status of the CICS bundle to the intended status for the platform. In this case, the bundle is installed into a disabled state. The status can also take a short amount of time to settle while data is collected across the CICSplex. If the status is not as you expect, wait for 15 seconds and refresh the Cloud Explorer view.

7. The CICS bundle can be enabled if the platform has previously been instructed to become enabled. If the platform status is not enabled, in the Cloud Explorer view, right-click the GeneralInsuranceDev platform and select **Enable**. The Perform Operation dialog displays.
8. In the Perform ENABLE Operation dialog, click **OK**. The platform enables all of the CICS bundles that compose the CICS TS platform, including the GeneralInsurancePlatformPolicy bundle.

Figure 5-9 shows the updated Cloud Explorer view displaying the enabled platform and CICS TS policy bundle.

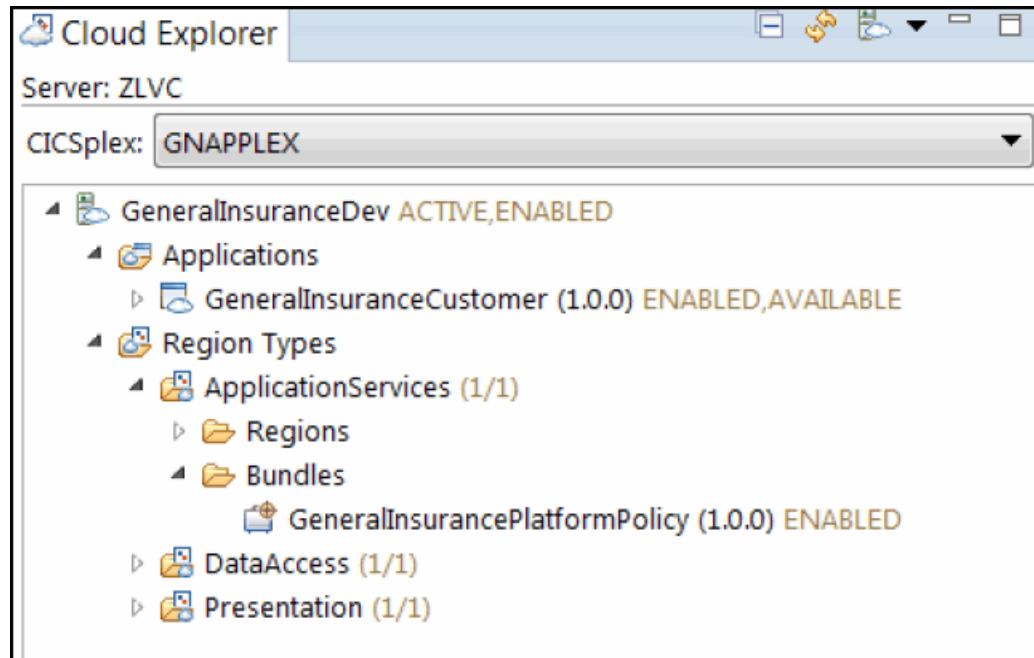


Figure 5-9 Cloud Explorer view of an active and enabled CICS TS platform

The GeneralInsurancePlatformPolicy CICS bundle includes a policy definition, and has been deployed to the GeneralInsuranceDev platform.

View the outcome of exceeding the policy threshold

The GeneralInsurancePlatformPolicy CICS bundle contains the GeneralInsurancePlatformPolicies policy and the associated Prevent24bitStorage policy rule. The policy has been deployed to the GeneralInsuranceDev platform for user tasks for applications deployed to the platform that attempt to obtain 24-bit storage.

Figure 5-10 shows the CICS TS MSGUSR data set resulting from an application attempting to obtain 24-bit storage on the GeneralInsuranceDev platform. The application task undergoes an abend with the code AMPB due to running an EXEC CICS GETMAIN BELOW API command.

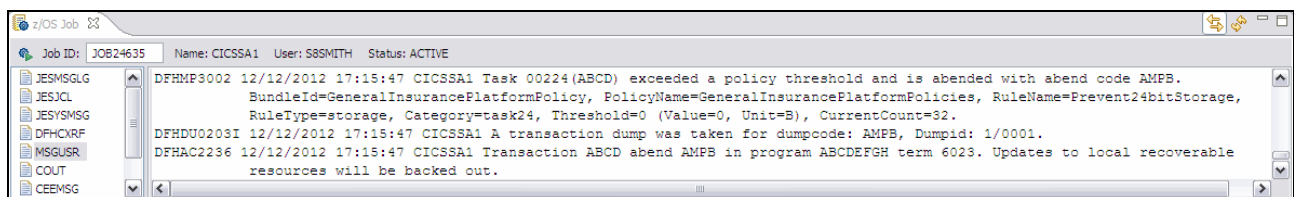


Figure 5-10 CICS TS MSGUSR data set after exceeding the threshold for policy GeneralInsurancePlatformPolicies

5.2.2 Application operation policy example

This section describes the creation and deployment of an application operation policy. A policy is deployed for the `inquireCustomer` operation in the `GeneralInsuranceCustomer` application. The policy outputs a message if the operation takes longer than expected. An example use of this message is an indicator to the application provider that the response times are not as expected.

Important: In our example, we have decided that the policy needs to change while the application moves from development into production. We might want to make the enforcement tougher in pre-production test, and force tasks that take too long to an `abend`. Therefore, the decision is made to add the policy into the application binding rather than the application, which enables us to leave the application unchanged through the move from development and into production.

The following list describes the steps in this section:

- ▶ Create and define the policy.
- ▶ Include the policy in the application binding.
- ▶ Discard the active application.
- ▶ Deploy the updated application.
- ▶ View the outcome of exceeding the policy threshold.

Prerequisites

The prerequisites include the following software setup:

- ▶ CICS TS regions running V5.1 or later. The walkthrough is done using CICS TS V5.2.
- ▶ CICS Explorer V5.2 or later, or CICS Explorer V5.2 SDK or later.

Tip: It is possible to follow the steps in this chapter if you are using CICS TS V5.1 and CICS Explorer V5.1. If this is the case, you can skip the instructions on making an application `Available` or `Unavailable`, because this capability was added in CICS TS V5.2.


The policy is added to the `GeneralInsuranceCustomer` application created in Chapter 4, “Creating an application” on page 51. The updated application is deployed to the `GeneralInsuranceDev` platform created in Chapter 3, “Creating a platform” on page 25.

The creation and deployment of the application operation policy example uses the CICS Explorer CICS Cloud perspective.

Create and define the policy

Perform the following steps to create the `GeneralInsuranceCustomerPolicy` CICS Bundle Project to define the policy, using the CICS Cloud perspective in CICS Explorer:

1. Right-click in the Project Explorer view and select **New** → **Project**. The New Project wizard displays.
2. In the New Projects wizard, expand **CICS Resources**, click **CICS Bundle Project**, and then click **Next**. The CICS Bundle Project wizard displays.

Alternatively, to create a project to contain the files to be referenced by a CICS bundle definition, click the bundle plus () icon in the toolbar to start the new CICS Bundle Project wizard.

3. Enter `general.insurance.customer.policy` as the Project name.

4. Enter `GeneralInsuranceCustomerPolicy` as the ID.
5. Enter `1.0.0` as the Version. Figure 5-11 shows the completed CICS Bundle Project wizard.

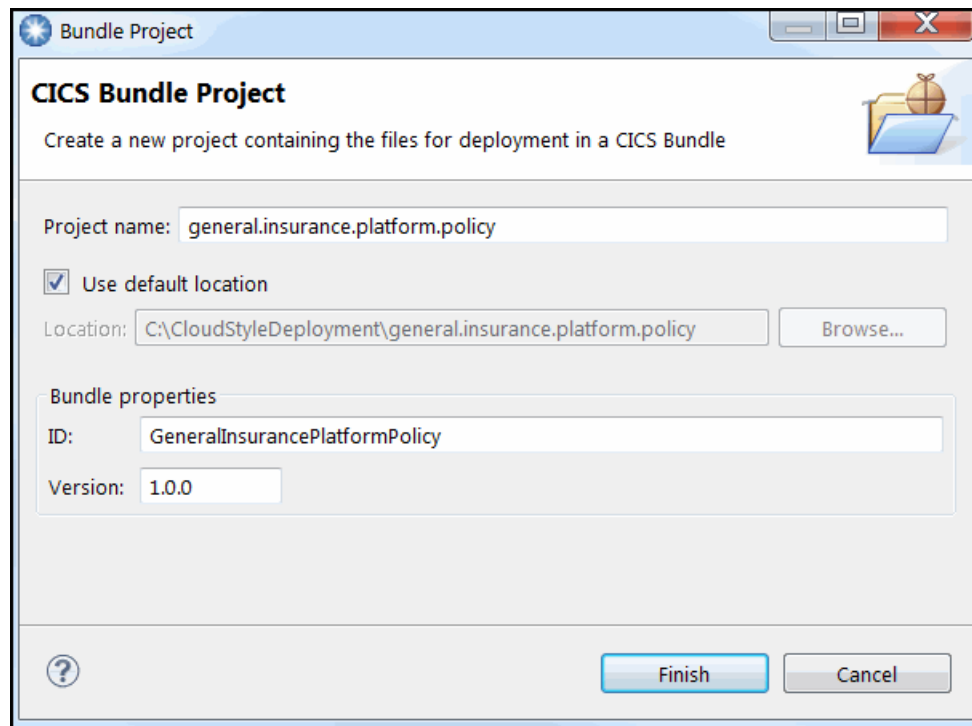


Figure 5-11 Create `GeneralInsuranceCustomerPolicy` bundle wizard

6. Click **Finish**. The bundle project is created and the CICS Bundle Manifest Editor appears.

Note: For the purpose of this example, to ensure that the policy threshold is met, the policy rule will be set for a very short interval.

The `GeneralInsuranceCustomerPolicy` CICS Bundle Project has been created. Next, create the policy definition in the CICS bundle using the CICS Bundle Manifest Editor.

7. The CICS Bundle Manifest Editor will automatically start after the creation of the CICS Bundle Project in the previous step. If the editor is not open, locate the bundle manifest file to start the editor. In the Project Explorer view, expand the `general.insurance.customer.policy` project, expand the `META-INF` subdirectory, and then double-click the `cics.xml` manifest file to start the CICS Bundle Manifest Editor.
8. Access the Overview tab in the CICS Bundle Manifest Editor for the `GeneralInsuranceCustomerPolicy` bundle. Click **New** in the Defined Resources section and select **Policy Definition** to open the Create Policy Definition wizard.
9. Enter `GeneralInsuranceInquireCustomerPolicies` as the Policy name.
10. Enter `Policies` for the general insurance inquire customer application operation as the Policy description.
11. Enter `ExcessiveCPUTime` as the Rule name.
12. Enter `Rule` to highlight inquire customer operation using excessive CPU time as the Rule Description.
13. Select the threshold Type **Time**.

14. Select the threshold Item **CPU Time**.
15. Select threshold Operator **Greater Than**, enter Value 1, and select Unit **Microseconds**.
16. Select the policy to **Issue message DFHMP3001**.
17. Clear **Open Editor**.

Figure 5-12 shows the completed Create Policy Definition wizard for the GeneralInsuranceInquireCustomerPolicies policy.

Figure 5-12 Completed Create Policy Definition wizard for the GeneralInsuranceInquireCustomerPolicies policy

18. Click **Finish**. The policy will be defined in the CICS bundle.

Important: The threshold and actions for the policy definition have been added to the CICS bundle. If this CICS bundle was deployed with the application, the policy would apply to all tasks in the application. Define a policy scope in the CICS bundle to identify a particular application operation for this policy.

19. In the CICS Bundle Manifest Editor for GeneralInsuranceCustomerPolicy, click the **Policy Scopes** tab.

20. In the Policy Scope section, click **Add**. The Create Policy scope wizard will appear.

21. In the Create Policy scope wizard, enter the Operation `inquireCustomer`.

22. Enter the Policy name `GeneralInsuranceInquireCustomerPolicies`.

Figure 5-13 shows the completed Create Policy scope wizard.

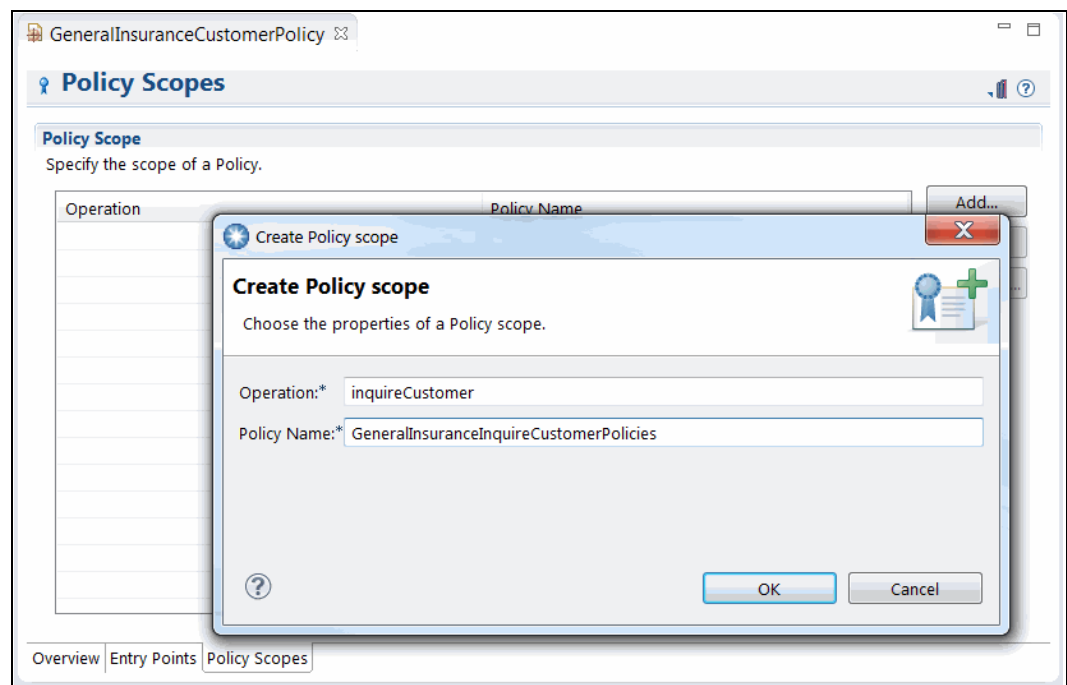


Figure 5-13 Create Policy scope wizard to associate the `inquireCustomer` operation with `GeneralInsuranceInquireCustomerPolicies` policy

23. Click **OK**. The policy scope association will be added to the CICS bundle.

As shown in Figure 5-14, a new entry will appear in the Policy Scope table to bind the scope of the GeneralInsuranceInquireCustomerPolicies policy to the inquireCustomer operation.

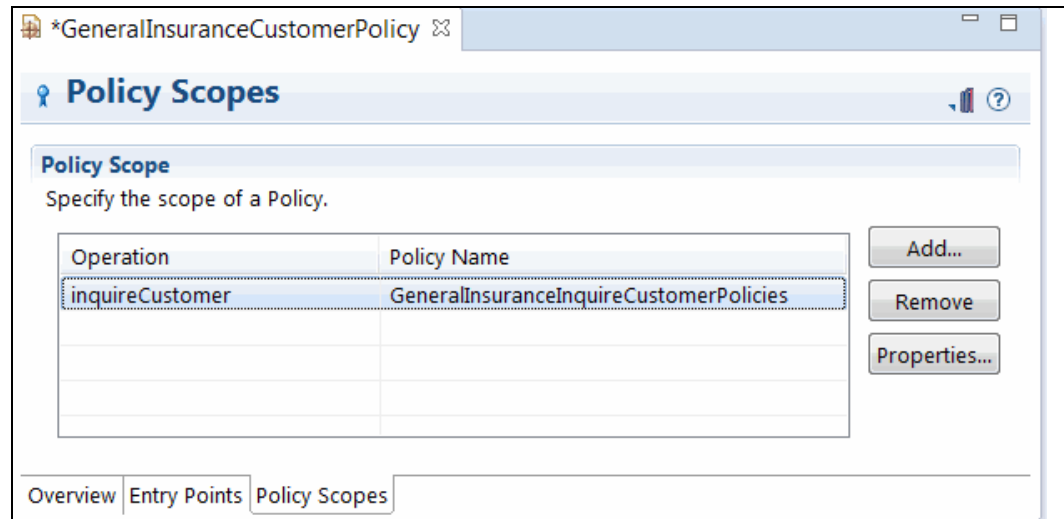


Figure 5-14 GeneralInsuranceCustomerPolicy CICS Bundle Manifest Editor Policy Scopes view

24. Press **Ctrl+S** to save the updated CICS bundle.

The GeneralInsuranceCustomerPolicy CICS Bundle Project has now been created with the GeneralInsurancePlatformPolicies policy associated with the inquireCustomer application operation.

Re-version the application binding and the application

By adding a new policy to the application, there is the potential for new CICS TS messages to be emitted if the policy threshold is exceeded. To help identify the cause of the new message if it is seen in test or production, we have chosen to increase the version number of the application binding and of the application itself.

To re-version, follow these steps:

1. If required, open the Application editor for the GeneralInsuranceCustomer application.
In the Project Explorer view, expand the general.insurance.customer.application application project, expand the META-INF subdirectory, and double click application.xml to open the Application editor.
2. On the Overview tab in the Application editor for GeneralInsuranceCustomer, in the General Information section, update the version from 1.0.0 to 1.0.1.

3. Press **Ctrl+S** to save updates in the Application editor.

Figure 5-15 shows the updated Application editor for the GeneralInsuranceCustomer application.

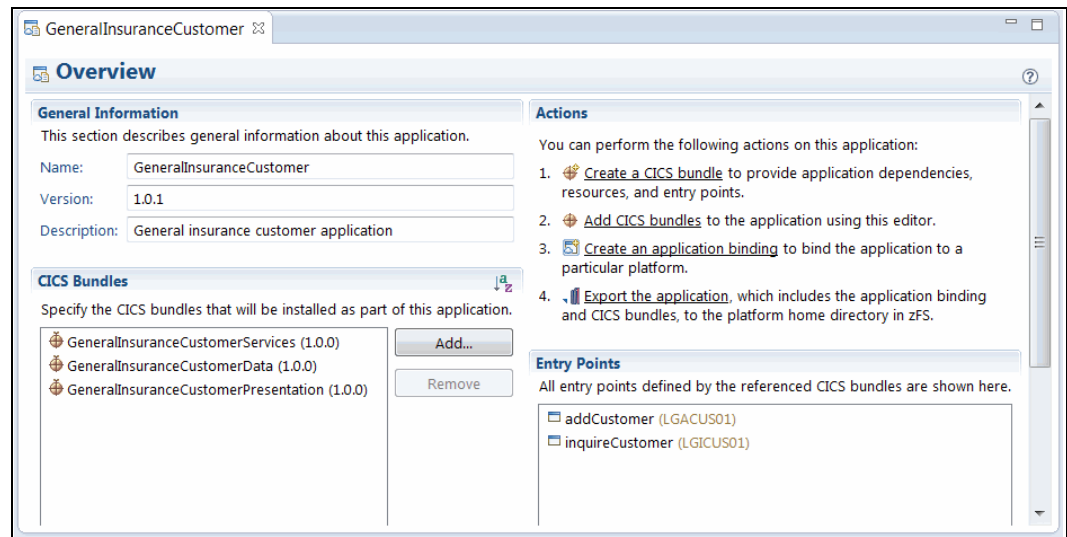


Figure 5-15 Completed Application editor changes for GeneralInsuranceCustomer

Remember: Errors will be reported in the application binding editor because it still refers to version 1.0.0 of the application. This will be corrected in the next steps.

4. If required, open the application binding editor for the GeneralInsuranceCustomerToDev application binding.

In the Project Explorer view, expand the `general.insurance.customer.to.dev.platform.binding` application binding project, expand the META-INF subdirectory, and double-click the `appbinding.xml` file to open the application binding editor.

5. On the Overview tab in the application binding editor for GeneralInsuranceCustomerToDev, in the General Information section, update the version from 1.0.0 to 1.0.1.
6. In the General Information section, click **Browse** next to Application: GeneralInsuranceCustomer (1.0.0). The Application Selection dialog displays.
7. In the Application Selection dialog, select **GeneralInsuranceCustomer (1.0.1)** and click **OK**.

8. Press **Ctrl+S** to save updates in the application binding editor.

Figure 5-16 shows the updated application binding editor for the GeneralInsuranceCustomerToDev application binding.

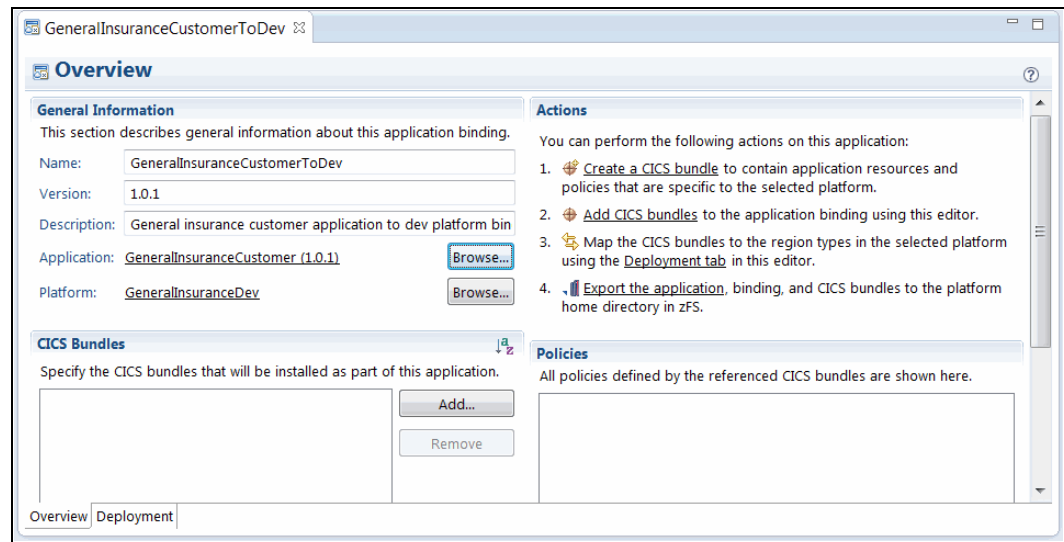


Figure 5-16 Completed application binding editor changes for GeneralInsuranceCustomerToDev

You have updated the micro version of the application and the application binding, due to the addition of a new policy into the CICS TS application, which can result in new messages being emitted.

Include the policy in the application binding

Perform the following steps to include the example policy in the application binding project for deployment to the dev platform:

1. In the Project Explorer view, expand the `general.insurance.customer.to.dev.platform.binding` application binding project, expand the `META-INF` subdirectory, and then double click the `bundles.xml` file to start the application binding editor.
2. On the Overview tab in the GeneralInsuranceCustomerToDev application binding editor, in the CICS bundles section, click **Add**. The CICS Bundle Selection dialog will appear.

3. In the CICS Bundle Selection dialog, select **GeneralInsuranceCustomerPolicy (1.0.0)**.

Figure 5-17 shows the GeneralInsuranceCustomerPolicy bundle selected in the CICS Bundle Selection dialog for addition to the CICS Application Binding Project.

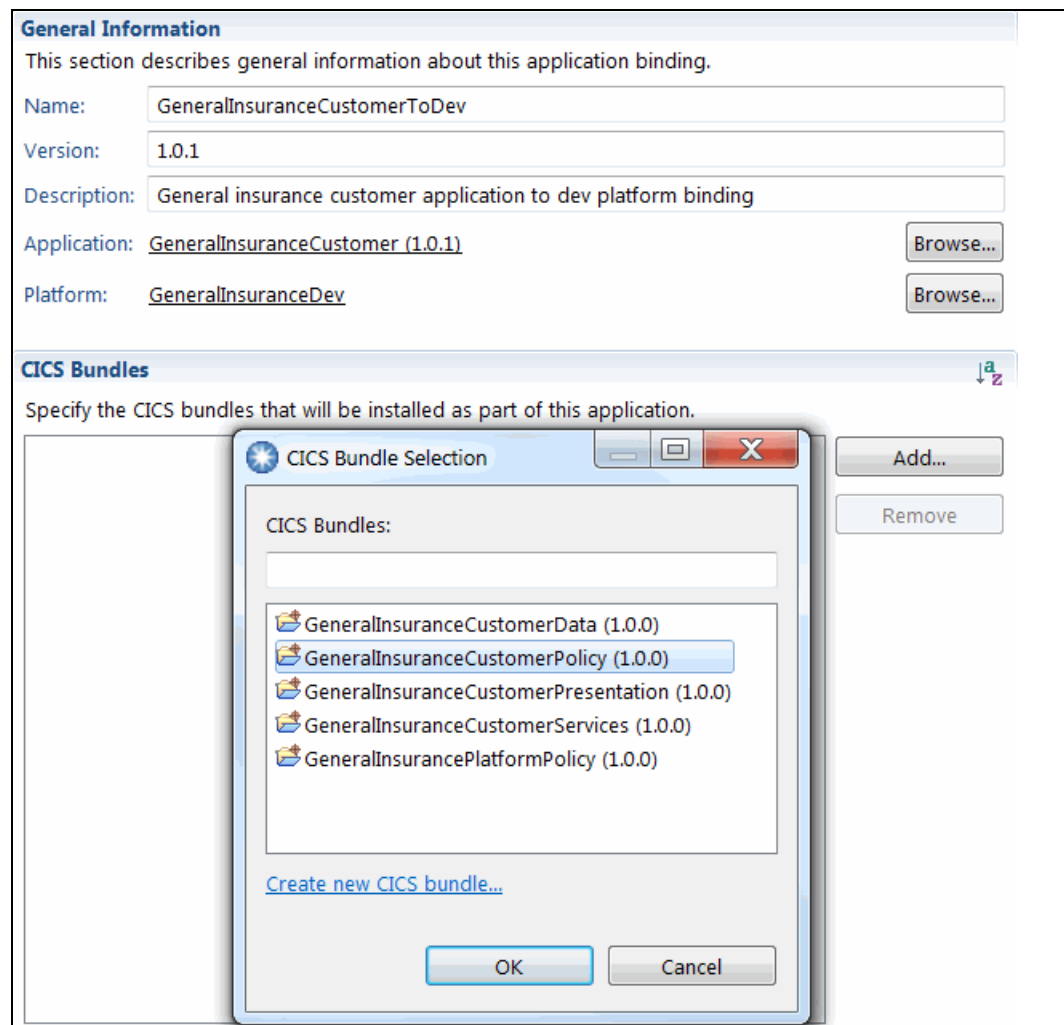


Figure 5-17 CICS Bundle Selection dialog to add a CICS bundle to an application binding project

4. Click **OK** in the CICS Bundle Selection to add the bundle to the application binding.

Remember: Errors will be reported in the application binding editor, because the deployment rules have not been defined for the policy bundle within the binding project. This will be corrected in the next steps.

5. Click the **Deployment** tab in the GeneralInsuranceCustomerToDev application binding editor.
6. Access the application binding editor Deployment tab. In the Region Types section, select **ApplicationServices**.
7. Again, access the application binding editor Deployment tab and, in the CICS bundles section, select the check box for the GeneralInsuranceCustomerPolicy (1.0.0) CICS bundle, in addition to any CICS bundles that are already selected.

You have now declared the deployment rule for the GeneralInsuranceCustomerPolicy bundle to the ApplicationServices region type in the GeneralInsuranceDev platform.

Figure 5-18 shows the updated deployment rules for the application binding. This includes the deployment rules for the GeneralInsuranceCustomerPolicy policy bundle to be deployed to a platform region type.

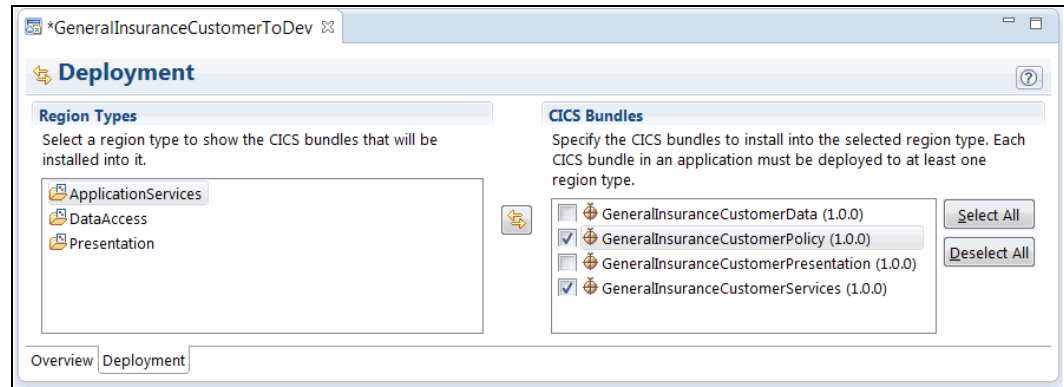


Figure 5-18 Deployment rules for CICS bundles in the application binding editor

8. Press **Ctrl+S** to save the updates made in the application binding editor.

The GeneralInsuranceCustomerPolicy CICS bundle, which contains the application operation policy, has been added to the application binding project, and the deployment rules have been defined.

Deploy the CICS TS application and create the application definition

Use the following steps to deploy the CICS TS application, CICS application binding, and CICS Bundle Projects using the application editor. Because we have increased the version number of the application and the application binding projects, we will create a new application definition to represent the new version:

1. If the Application editor for the general.insurance.customer.application project is not open, locate the application files to start the application editor.

In the Project Explorer view, expand the general.insurance.customer.application project, and expand the META-INF subdirectory. Then, double click the application.xml file to start the Application editor.

2. In the Actions section of the Application editor, click **Export the application**.

The Export Application to the home directory of a platform wizard displays. This wizard enables you to specify the CICSplex and CICS TS platform to which you want to deploy your CICS TS application. This action identifies the CICS application binding that will be used for deployment of the application.

3. If required, connect to your CICS TS environment's CMCI connection.

4. In GNAPPLEX, under GeneralInsuranceDev, select **GeneralInsuranceCustomerToDev (1.0.1)**, as shown in Figure 5-19.

Important: The export can only be made to an installed platform using this wizard. This enables the CICS Explorer to ensure that the artifacts are exported to the correct locations for use in the selected platform.

5. Select **Create Application Definition after export finishes**.

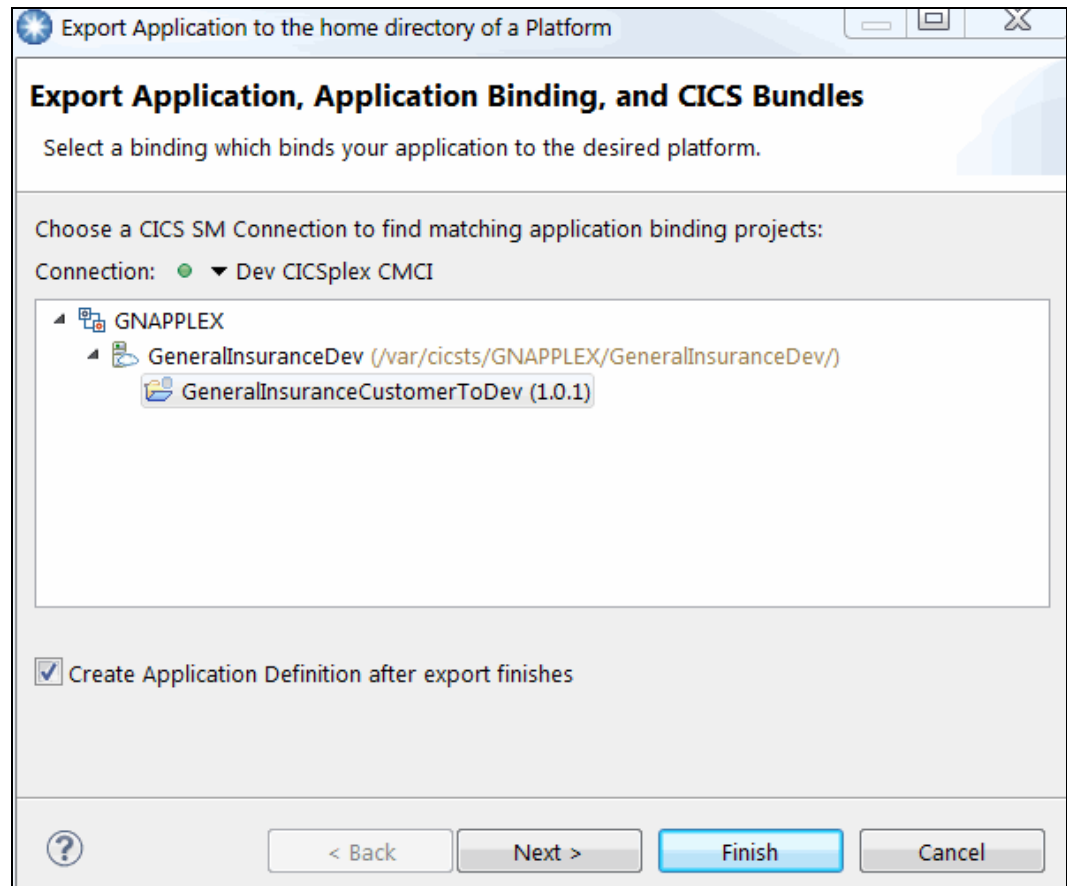


Figure 5-19 Export Application to the home directory of a platform

6. Click **Next**.
7. If required, connect to your CICS TS environment's z/OS FTP connection. Click **Finish**.

The CICS Explorer begins to export the CICS TS application, CICS application binding, and CICS Bundle Projects to the z/OS UNIX File System. The projects are deployed to the platform home directory of the CICS TS platform in the applications, bindings, and bundles subfolders, where appropriate.
8. When the export has completed, the New Application Definition wizard displays. This wizard can be used to define the CICS TS application to your CICS TS environment. All fields except the Name field are filled in using information found during the export process.

Use the following steps to define the CICS TS application definition:

1. Enter GENACUST in the Name field.
2. Enter General insurance customer application in the Description field. The New Application Definition wizard should look like Figure 5-20.

New Application Definition

Create Application Definition
Create a application definition.

CICSplex:* GNAPPLEX

Name:* GENACUST

Description: General insurance customer application

Platform Definition:* GNAPDEV

Version:* 1.0.1

Connect to z/OS to browse directories

z/OS connection: ☒ Dev CICSplex FTP

Application Directory:* /var/cicsts/GNAPPLEX/GeneralInsuranceDev/applications/GeneralInsuranceCustomer_1.0.1/ Browse...

Binding Directory:* /var/cicsts/GNAPPLEX/GeneralInsuranceDev/bindings/GeneralInsuranceCustomerToDev_1.0.1/ Browse...

☐ Open editor

Finish Cancel

Figure 5-20 New Application Definition wizard for V1.0.1 of the application

3. Click **Finish**. The CICS TS application definition has now been created. The Cloud Explorer view should now look similar to Figure 5-21.

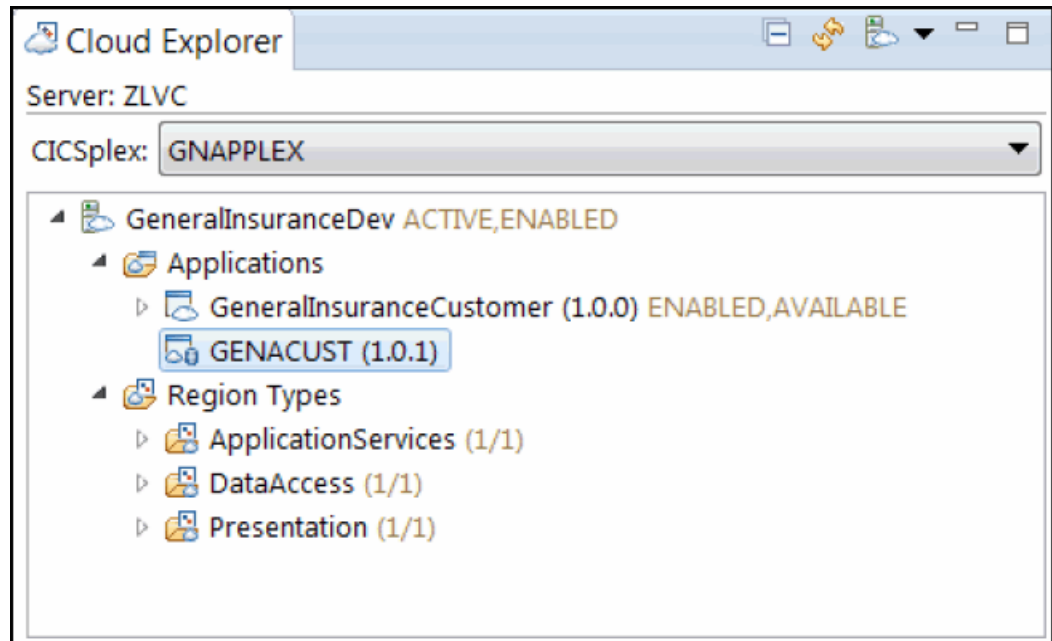


Figure 5-21 Cloud Explorer view after the CICS TS application definition has been created

The CICS TS application, CICS application binding, and CICS bundle that compose the GeneralInsuranceCustomer application have been created and deployed to the GeneralInsuranceDev platform. The application definition has been created and is ready to be installed into the GeneralInsuranceDev platform.

Make Unavailable, Disable, and Discard V1.0.0 of the application

Because both version 1.0.0 and version 1.0.1 of the GeneralInsuranceCustomer application need to modify the same CICS system definition (CSD)-defined CICS TS programs as their application entry points, it is not possible for them both to be enabled at the same time.

Tip: In Chapter 6, “Packaging an application for multiversion deployment” on page 127, we demonstrate how to package your CICS TS application resources, so that it is possible to make more than one version of the CICS TS application available at the same time.

Perform the following steps to Make Unavailable and Disable the installed GeneralInsuranceCustomer V1.0.0 Application:

1. In the Cloud Explorer view, expand the GeneralInsuranceDev platform and expand subdirectory Applications to view the applications in the platform. Double-click the GeneralInsuranceCustomer (1.0.0) application to start the Online Application editor for the GeneralInsuranceCustomer application.
2. On the Overview tab in the Online Application editor for the GeneralInsuranceCustomer application, click **Make Unavailable** in the General Information section. The Perform UNAVAILABLE Operation dialog displays.
3. In the Perform UNAVAILABLE Operation dialog, click **OK**.

The application will be made unavailable, and external callers will not be able to start new tasks associated with the application.

- On the Overview tab of the Online Application editor for the GeneralInsuranceCustomer application, in the General Information section, expand the drop-down menu next to the Disable button and select **Disable and Discard**, as shown in Figure 5-22. The Perform DISABLE Operation dialog displays.

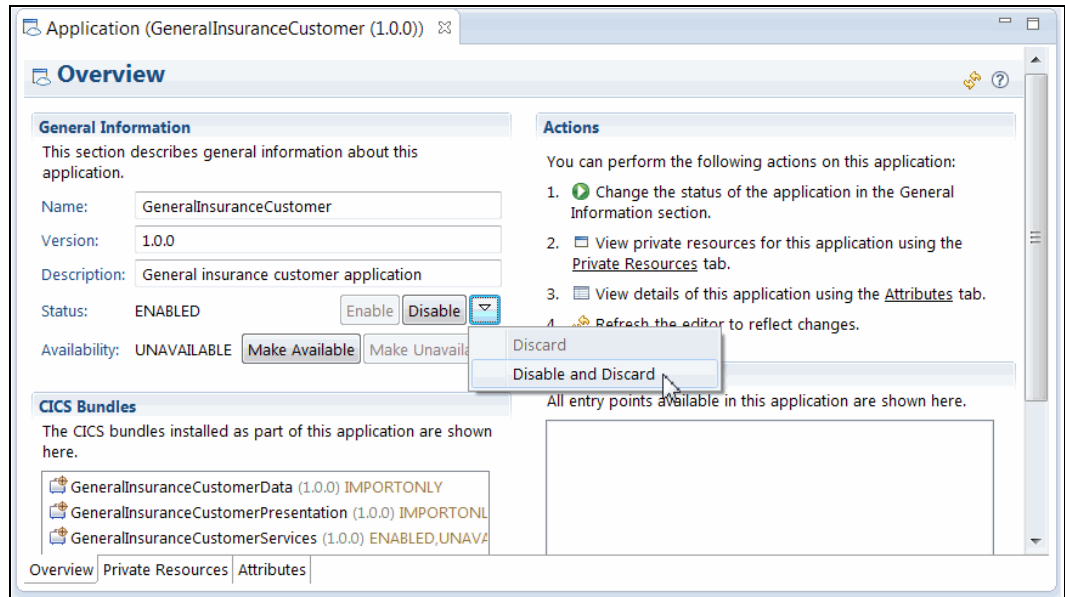


Figure 5-22 Drop-down menu to disable and discard the application in the Online Application Editor

- Click **OK**. The application is disabled. The Perform DISCARD Operation dialog displays.
- Click **OK**. The application is discarded. The Online Application Editor closes. Figure 5-23 shows the updated Cloud Explorer view after the discard of the installed application.

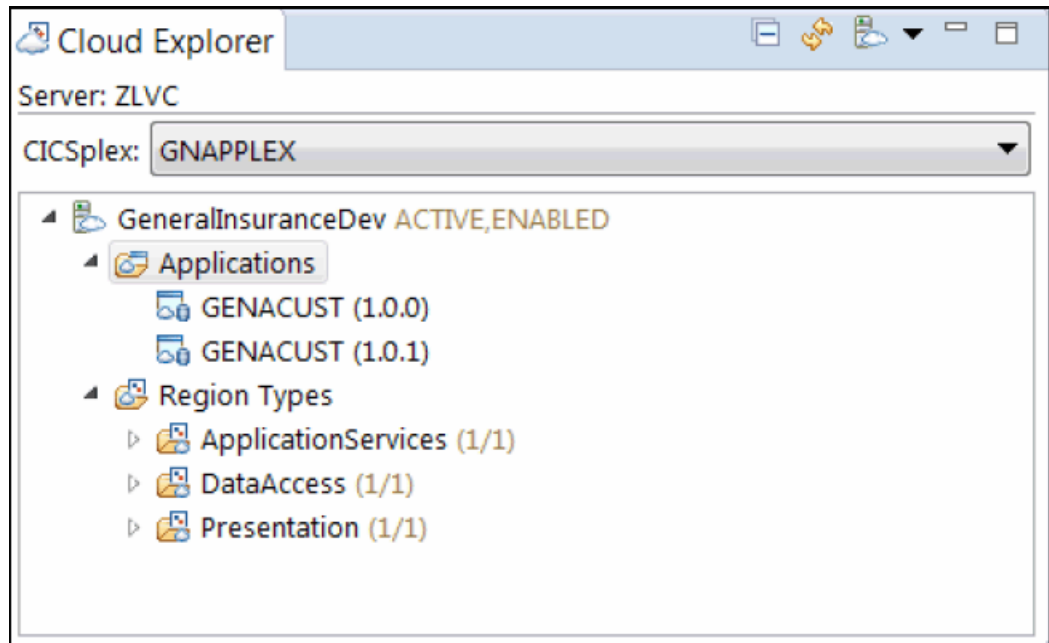


Figure 5-23 After the discard of the installed application GeneralInsuranceCustomer version 1.0.0, the application definition GENACUST is displayed in the Cloud Explorer view

The application has been discarded from the platform.

5.2.3 Installing the updated application

Use the following steps to install the GeneralInsuranceCustomer V1.0.1 application using the CICS Cloud perspective in CICS Explorer:

1. In the Cloud Explorer view, expand the GeneralInsuranceDev platform and expand the Applications subdirectory. Right-click the **GENACUST (1.0.1)** CICS TS application definition and click **Install**. The Perform INSTALL Operation dialog displays.
2. In the Perform INSTALL Operation dialog for the installation action, click **OK**.

The CICS TS application and all associated CICS bundles now install into the CICS TS regions in the CICS TS platform. The GeneralInsuranceCustomer application is visible in the Cloud Explorer view, as shown in Figure 5-24.

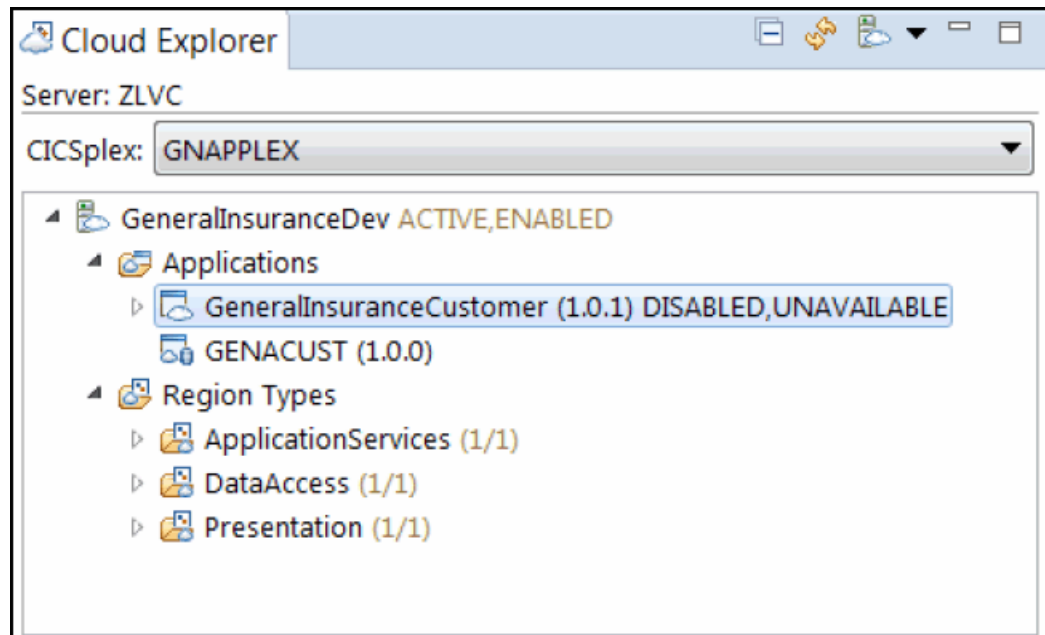


Figure 5-24 GeneralInsuranceCustomer V1.0.1 visible in the Cloud Explorer after being installed

Tip: The installation action on the CICS TS application installs the CICS bundles that compose the application into the CICS TS regions in the platform region types, as defined by the deployment bindings. If the CICS TS application state does not immediately display as DISABLED, then wait for 15 seconds and refresh the Cloud Explorer view.

The CICS TS application installs all associated bundles into a disabled and unavailable state.

Use the following steps to enable the GeneralInsuranceCustomer V1.0.1 CICS TS application:

1. Double-click the **GeneralInsuranceCustomer (1.0.1)** application in the Cloud Explorer view. The Online Application editor for the GeneralInsuranceCustomer application displays, as shown in Figure 5-25.

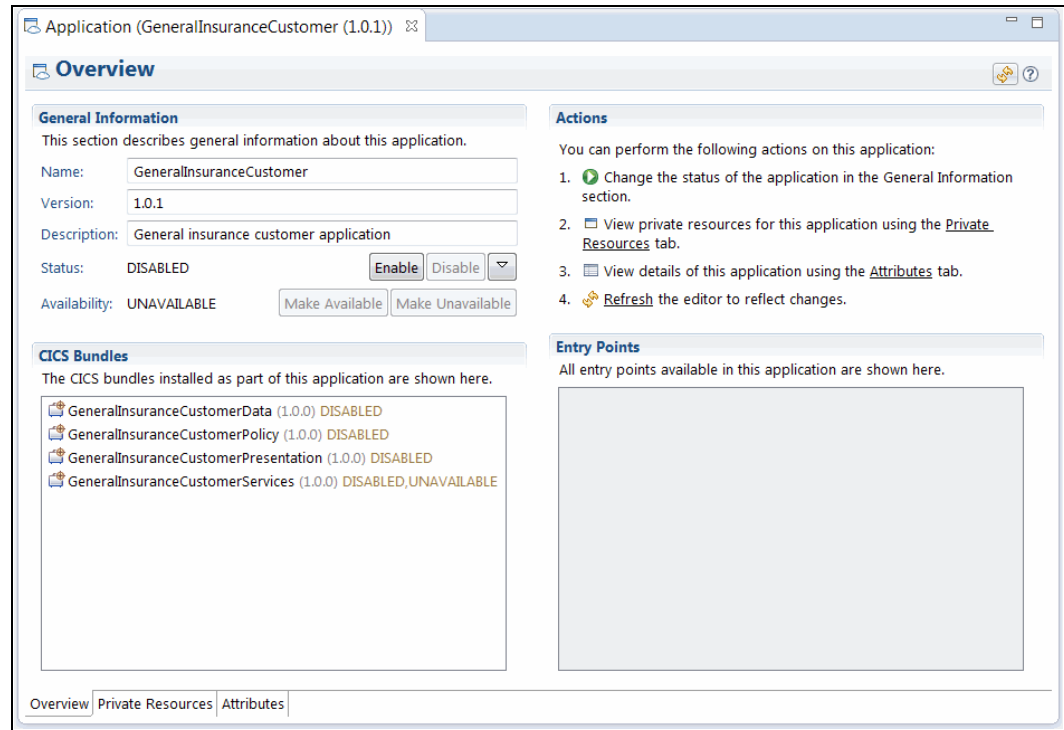


Figure 5-25 Online Application editor for GeneralInsuranceCustomer application version 1.0.1

2. Access the Overview tab of the Online Application editor for the GeneralInsuranceCustomer V1.0.1 application. In the General Information section, click **Enable**. The Perform ENABLE Operation dialog displays.
3. Click **OK**.

The CICS TS application now attempts to become enabled. After a short amount of time, the status of the CICS TS application will reflect the status of the dependencies that were declared in the CICS bundles. Therefore, the status of the CICS TS application reflects the status of the GENAPP application in the CICS TS environment.

The application is now enabled in an unavailable state. The unavailable state means that the application's entry points are not yet applied, so the new threshold policy is not active.

Use the following steps to make available the GeneralInsuranceCustomer V1.0.1 CICS TS application using the online application editor:

1. Access the Overview tab of the online application editor for the GeneralInsuranceCustomer V1.0.1 application. In the General Information section, click **Make Available**. The Perform AVAILABLE Operation dialog displays.
2. Click **OK**. The application is made available.

The CICS Cloud Explorer view should look similar to Figure 5-26.

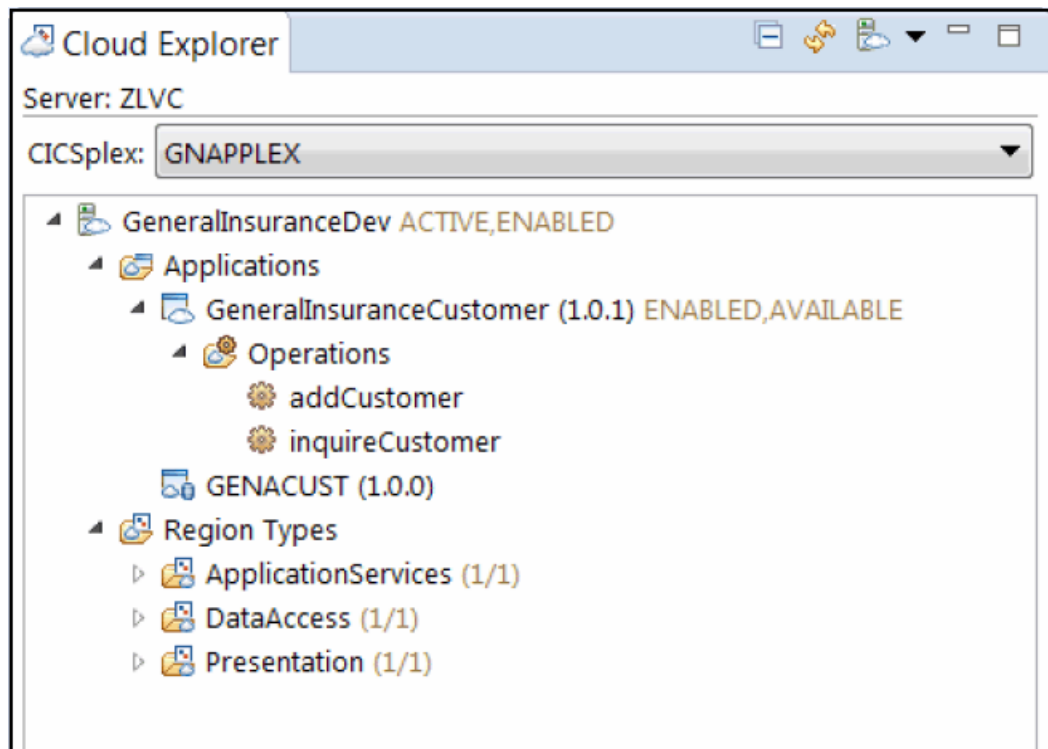


Figure 5-26 The Cloud Explorer view showing the deployed version 1.0.1 of CICS TS application GeneralInsuranceCustomer

The GeneralInsuranceCustomer V1.0.1 application is now available for users to start through the identified application entry points. The application entry points set the application context on tasks, which enables the new threshold policy to be applied.

View the outcome of exceeding the policy threshold

The GeneralInsuranceInquireCustomerPolicies policy, and the associated ExcessiveCPUTime policy rule, have been deployed to the GeneralInsuranceCustomer V1.0.1 application. This policy has been scoped to apply only to tasks associated with the inquireCustomer operation. The policy outputs a message if the tasks associated with the operation take longer then the set threshold.

For this example, the policy threshold has been set to a low trigger point. To exceed the policy threshold, the GENAPP application must run a customer inquiry.

Use the following steps to trigger the policy in CICS TS:

1. Run the SSC1 transaction to start the General Insurance Customer menu.
2. Enter a Customer Number, for example 0000000001.
3. Select option 1 for Customer Inquiry and click Enter.

By driving the customer inquiry action, the policy exceeds the defined threshold and results in a message being outputted.

Figure 5-27 shows the emitted message resulting from the GENAPP inquireCustomer operation exceeding the policy threshold. This can be used by an application provider as an indicator that the application response times are not as expected.

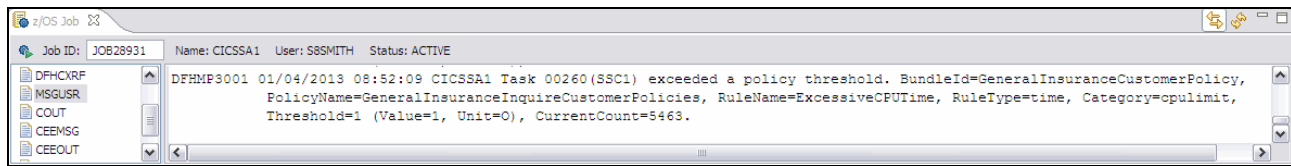


Figure 5-27 CICS MSGUSR data set after exceeding the threshold for policy GeneralInsuranceInquireCustomerPolicies

5.2.4 View active policy

This section shows how to view policies that have been deployed into a platform or application.

The example policies shown in this section are based on the GENAPP example and policies deployed in the following sections:

- ▶ 5.2.1, “Platform policy example” on page 100
- ▶ 5.2.2, “Application operation policy example” on page 109

View deployed policy

Use the Search menu action in the Cloud Explorer view to identify which policies are deployed. Searching for deployed policies can be conducted at four resource levels:

- ▶ On a platform

In the Cloud Explorer, right-click the GeneralInsuranceDev platform and select **Search → Policy rules for platform**.
- ▶ On a region type

In the Cloud Explorer, expand the GeneralInsuranceDev platform, expand Region Types, and right-click the ApplicationServices region type. Select **Search → Policy rules for region type**.
- ▶ On an application

In the Cloud Explorer, expand the GeneralInsuranceDev platform, expand Applications, and right-click the GeneralInsuranceCustomer V1.0.1 application. Select **Search → Policy rules for application**.
- ▶ On an application operation

In the Cloud Explorer, expand the GeneralInsuranceCustomer V1.0.1 application, expand Operations, and right-click the inquireCustomer operation. Select **Search → Policy rules for operation**.

Notice that the platform policy example is returned for searches at the platform and region type level. Platform policies are also returned for searches at the application level, because the policy is in effect for tasks associated with the application on the GeneralInsuranceDev platform.

Figure 5-28 shows the results of a search for policies conducted on the GeneralInsuranceDev platform.

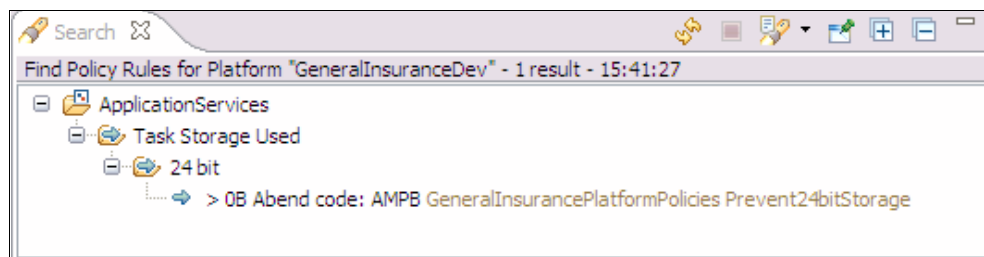


Figure 5-28 Search results for policies deployed to platform GeneralInsuranceDev

Operation-level searches return results for the application operation policy example. The search results also include the platform policy example. This indicates that both example policies are in effect for tasks associated with the inquireCustomer operation and GeneralInsuranceDev platform.

Figure 5-29 shows the results of a search for policies conducted on the inquireCustomer application operation.

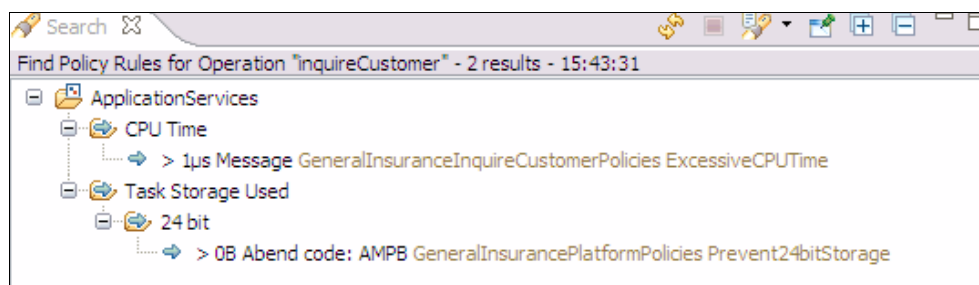


Figure 5-29 Search results for policies deployed to operation inquireCustomer

5.3 Results of adding a policy

This chapter introduced CICS TS policies, and stepped through two examples of policies working with the GENAPP example application.

The first example was a platform-level policy that abnormally ends application tasks that attempt to use 24-bit task storage. This is an example of how a policy can deliver automated control.

The second example alerted the application provider to tasks in the inquire customer operation that are deemed to take an excessive amount of execution time. This is an example of how a policy can monitor critical thresholds, and provide alerts when usage exceeds defined limits.

For further information and design considerations for CICS TS policies, see Chapter 8, “Managing by policy” on page 181.



Packaging an application for multiversion deployment

Chapter 4, “Creating an application” on page 51 showed how to create a *virtual application representation* of the General Insurance Application (GENAPP). It does not make any changes to the underlying application. However, it provides business value with the description of application entry points.

The application entry points identify when tasks running in IBM Customer Information Control System (CICS) Transaction Server (CICS TS) are entering into your application, and associates application context data with the workload. The chapter also demonstrated how you can view the health and availability of required resources within a single view, with no changes to the underlying application.

This chapter walks you through the steps of the typical process of developing the CICS TS application for application multiversion deployment. By using the application multiversioning support, introduced in CICS TS V5.2, you can have numerous versions of your application running at the same time.

The benefits of application multiversioning include the following abilities:

- ▶ Have more than one version of the application running concurrently.
- ▶ Maintain support for particular versions of applications for migrating users.
- ▶ Showcase new application features to a subset of your users.
- ▶ Prepare and verify the installation of a new application without deprovisioning the in-service application.
- ▶ Bring a new version of the application into service with no service outage.
- ▶ Roll back versions of the application.

This chapter centers around two worked examples for the GeneralInsuranceCustomer CICS TS application. Focused on the application multiversion support, these examples require a minimum level of CICS TS V5.2:

- ▶ The first example prepares the application for adoption of application multiversioning.
You repackage key resources into the application for lifecycle management. These resources include the programs identified as the application entry points, and version-specific load libraries.
- ▶ The second example uses the application multiversion support.
You provide a source code update and deploy a new version of the application. You deploy the next version of the application and migrate all users to the new version with no service interruptions.

This chapter contains the following topics:

- ▶ 6.1, “Prerequisites” on page 128
- ▶ 6.2, “Example 1: Repackaging CICS TS resources into the GeneralInsuranceCustomer CICS TS application” on page 128
- ▶ 6.3, “Example 2: Provisioning a logic change to the GeneralInsuranceCustomer CICS TS application” on page 154
- ▶ 6.4, “Possible extensions for the application examples” on page 171
- ▶ 6.5, “Summary” on page 172

6.1 Prerequisites

The following sections build on previous instructions described in Chapter 2, “GENAPP introduction” on page 15, and Chapter 3, “Creating a platform” on page 25. To proceed, you need the following components:

- ▶ CICS TS regions running V5.2 or later
- ▶ CICS Explorer V5.2 or later, or CICS Explorer V5.2 SDK or later
- ▶ The GeneralInsuranceDev platform defined and installed from Chapter 3, “Creating a platform” on page 25
- ▶ The GENAPP application available and installed into a single managed CICS TS region, as described in Chapter 2, “GENAPP introduction” on page 15 and Appendix A, “Setup and environment” on page 207
- ▶ (optional) Created policy definitions from Chapter 5, “Applying a policy” on page 97

6.2 Example 1: Repackaging CICS TS resources into the GeneralInsuranceCustomer CICS TS application

In this example, you prepare the application for adoption of multiversioning support. You repackage key resources into the application for lifecycle management. These key resources include the programs identified as application entry points and application version-specific load libraries.

By bringing CICS TS resource definitions into a CICS TS application, you provide better management for the application throughout its lifecycle. During application development, it is beneficial to have the resources described in a single development environment, and editable by the application developers.

When the application is in source code management, it can encompass and track all related artifacts through application deployment. The application can be provisioned and deprovisioned through a single control point, while the states of all of the related CICS TS resources are managed by the system.

In this example, you provision an update to the CICS TS application. You step through the process of repackaging the CICS TS resources related to the ApplicationServices CICS TS regions (the application-owning region (AOR) CICS TS regions) in preparation for application multiversioning. You move definitions from the CICS system definition (CSD) into the GeneralInsuranceCustomer CICS TS application for application entry point resources.

The change in the CICS TS application does not change any external characteristics of the application, and you want all users of the application to use the updated version. Therefore, this will be a micro version increment of the application, from version 1.0.1 to version 1.0.2.

The following list describes the steps that you undertake:

- ▶ 6.2.1, “Discarding the existing application and resources” on page 129
- ▶ 6.2.2, “Allocating a new data set, preparing load modules, and creating a dynamic load library definition” on page 133
- ▶ 6.2.3, “Updating CICS TS resources” on page 137
- ▶ 6.2.4, “Updating the application version” on page 144
- ▶ 6.2.5, “Exporting, installing, and enabling the application” on page 148
- ▶ 6.2.6, “Making the application available and running it” on page 153

6.2.1 Discarding the existing application and resources

In the previous chapter, you deployed the GeneralInsuranceCustomer version 1.0.1 CICS TS application. In this chapter, you update the application to version 1.0.2, while repackaging CICS TS resources for multiversion support.

Version 1.0.1 relied on key CICS TS resources to be provided by the CSD. You cannot have application entry points for different versions of an application share the same resource name while the resource is provided by CSD/CICSplex SM Business Application Services (BAS). The application multi-version support enables multiple versions of an application to declare CICS TS resources with the same name.

This can result in multiple programs in CICS TS with the same name, each being unique, and private to the application version instance. This is only possible for program resources defined using a CICS bundle associated with an application (or auto-installed using an application private LIBRARY). As a result, you discard GeneralInsuranceCustomer version 1.0.1.

Perform the following steps, using the CICS Cloud perspective in CICS Explorer to discard the existing application:

1. In the Cloud Explorer view, expand the GeneralInsuranceDev platform and the Applications subfolder to view the currently deployed CICS TS applications, as shown in Figure 6-1.

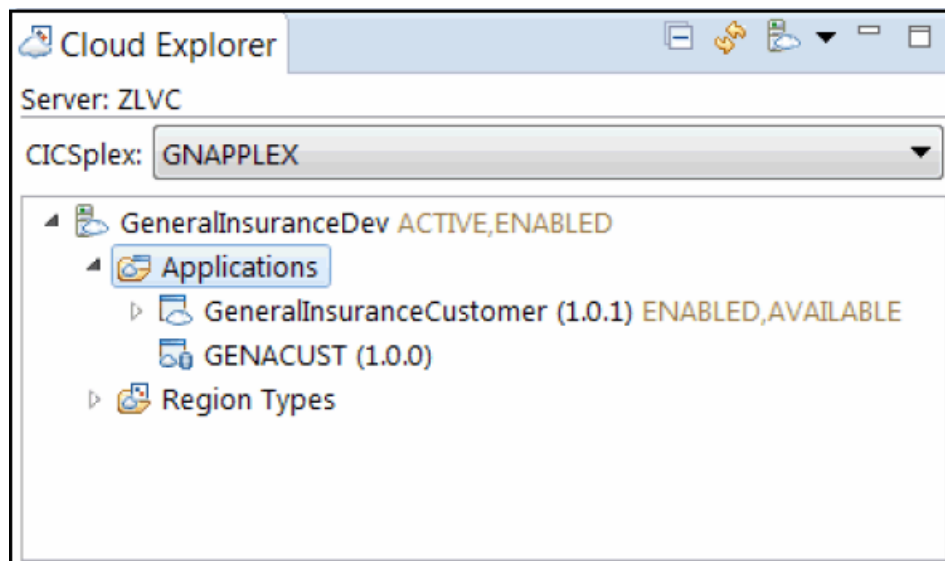


Figure 6-1 Cloud Explorer view showing existing deployed CICS TS applications

2. Right-click (CICS TS application) **GeneralInsuranceCustomer (1.0.1)** and select **Make Unavailable**.
The Perform Operation dialog appears. Click **OK** to make the application unavailable.
3. Right click (CICS TS application) **GeneralInsuranceCustomer (1.0.1)** and select **Disable**.
The Perform Operation dialog appears. Click **OK** to disable the application.
4. Right click (CICS TS application) **GeneralInsuranceCustomer (1.0.1)** and select **Discard**.
The Perform Operation dialog appears. Click **OK** to discard the application.

The GeneralInsuranceCustomer version 1.0.1 CICS TS application has been discarded. The Application entry in the Cloud Explorer view is replaced with the application definition, as shown in Figure 6-2.

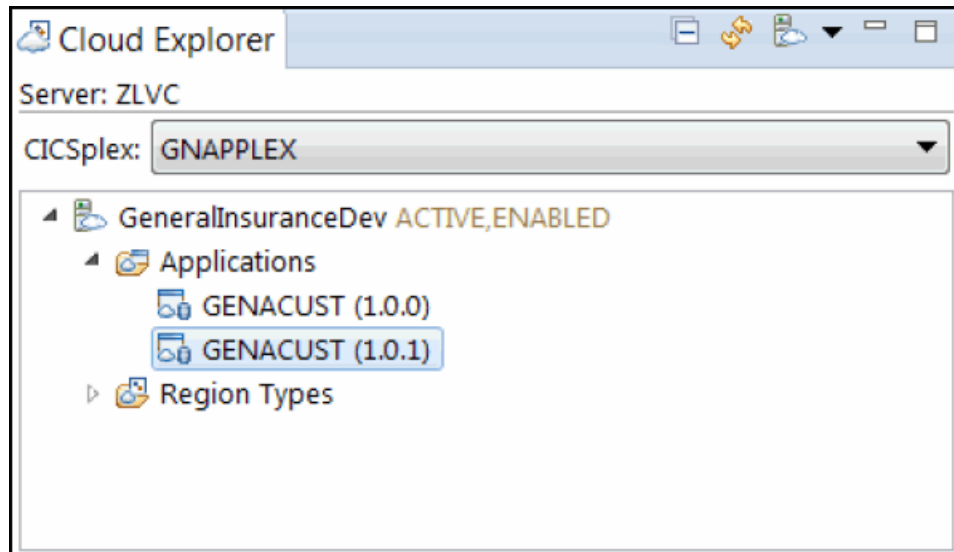


Figure 6-2 Cloud Explorer view showing application definition after discard of the GeneralInsuranceCustomer version 1.0.1 application

You repackage the ApplicationServices layer CICS TS programs (LGACUS01 and LGICUS01) in to the CICS TS application. Perform the following steps to remove the resources that were created during the initial installation of GENAPP described in Appendix A, “Setup and environment” on page 207:

1. Use the CICS Explorer CICS SM perspective to manage CICS TS resources and CSD definitions.
2. From the menu, click **Window** → **Open Perspective** → **Other**. The Open Perspective dialog appears.

- Click **CICS SM**, as shown in Figure 6-3, and click **OK** to open the CICS SM perspective.

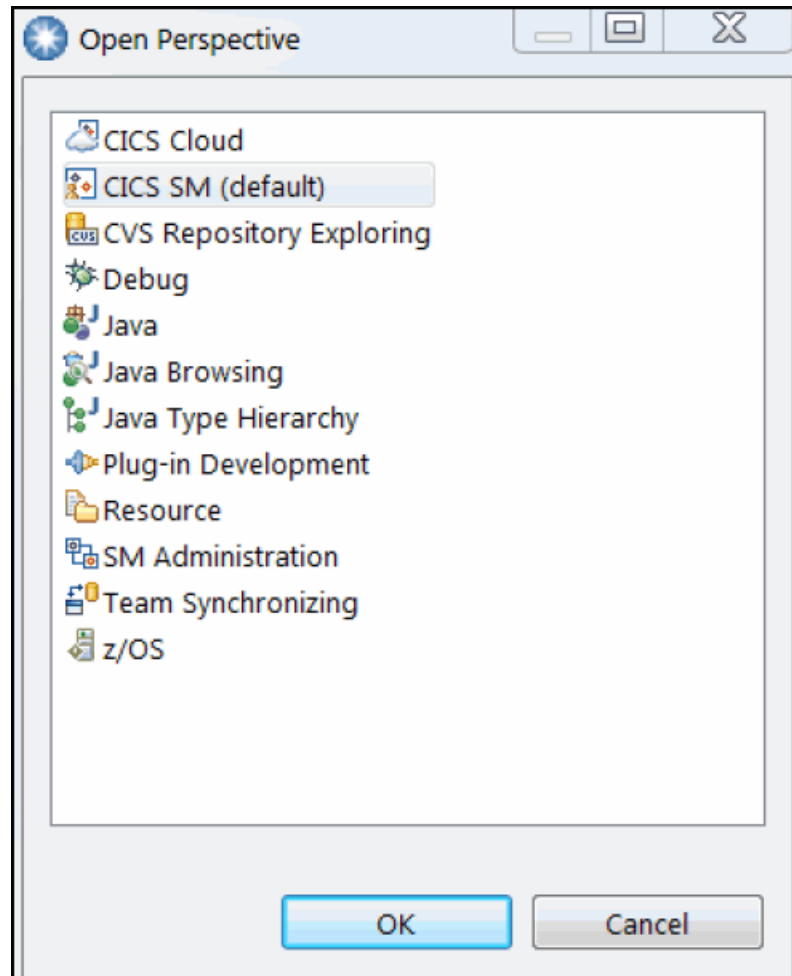


Figure 6-3 Dialog to open the CICS SM perspective in CICS Explorer

- From the menu, click **Operations** → **Programs**. The Programs view displays.
- In the Programs view, add the filter **LGACUS01** and press **Enter**. The Programs view refreshes and only displays the results of the LGACUS01 program, as shown in Figure 6-4.

Region	Name	Status	Use Count	Concurr...	Language	Share St...	CEDF Sta...	NEW...
IYK2ZLVG	LGACUS01	✓ ENABLED	0	0	COBOL	PRIVATE	CEDF	NOT...

Figure 6-4 Program Operations view with applied filter name LGACUS01

- Right-click **LGACUS01** and select **Disable**. The Perform Operation dialog displays. Click **OK** to disable the program.
- Right-click **LGACUS01** and select **Discard**. The Perform Operation dialog displays. Click **OK** to discard the program.

You have discarded the CICS TS LGACUS01 program resource. Repeat the process for the LGICUS01 program.

8. In the Programs view, replace the filter with LGICUS01 and press **Enter**. The Programs view refreshes and displays only the results of the LGICUS01 program.
9. Right-click **LGICUS01** and select **Disable**. The Perform Operation dialog displays. Click **OK** to disable the program.
10. Right-click **LGICUS01** and select **Discard**. The Perform Operation dialog displays. Click **OK** to discard the program.
11. Clear the filter text and close the Programs view. You have discarded the two CICS TS program resources that are packaged into the CICS TS application.

Optionally, you can delete the CSD definitions for the LGACUS01 and LGICUS01 program definitions in the GENASAP group to prevent the programs from being re-installed on future group installs. Using these steps, you have removed the CICS TS program resources originating from the CSD for the GENAPP ApplicationServices regions, in order for the definitions to be provided by the application.

6.2.2 Allocating a new data set, preparing load modules, and creating a dynamic load library definition

In “Building the application environment” on page 212, all of the GENAPP program load modules are provided in one data set load library, `<USERHLQ>.CB12.LOAD`. This data set is referenced by a dynamic load library in CICS TS called *GENALIB*.

You will now allocate a new data set to provide the program load modules that will be private to a particular version of the application. The new data set load library contains the application programs specifically for the ApplicationServices region type that we plan to repackage in to the CICS TS application. The new data set load library will be `<USERHLQ>.CB12.APPLSERV.#1.#0.#2`. You will then create a new dynamic load library definition called *GENAAPSV* to add this newly allocated data set to the application.

This section contains the following tasks:

- ▶ “Allocating a new data set and copying load modules” on page 133
- ▶ “Creating a CICS bundle and defining a dynamic load library definition” on page 134

Allocating a new data set and copying load modules

Perform the following steps to create a new load library data set to serve application programs specifically for the ApplicationServices CICS TS application region type:

1. Allocate the following load library, replacing `<USERHLQ>` with your high-level qualifier that you used for the installation of GENAPP. You can base the data set characteristics rules on the existing `<USERHLQ>.CB12.LOAD` data set, `<USERHLQ>.CB12.APPLSERV.#1.#0.#2`. You have now created a load library data set to contain the program load modules for the ApplicationServices region type in the GeneralInsuranceApplication V1.0.2 CICS TS application.
2. Copy the following load modules from the `<USERHLQ>.CB12.LOAD` data set to the `<USERHLQ>.CB12.APPLSERV.#1.#0.#2` data set:
 - LGACUS01
 - LGICUS01

You have now provided an instance of the program load module for a particular version of the CICS TS application to reference.

Creating a CICS bundle and defining a dynamic load library definition

You have created a load library data set <USERHLQ>.CB12.APPLSERV.#1.#0.#2, which contains the program load modules for application resources in the ApplicationServices region type. You next prepare the resources required to associate the load library with the application.

Perform the following steps to create a new CICS bundle and LIBRARY definition (steps later in this example associate the CICS bundle with the application binding project):

1. Right-click in the Project Explorer view and select **New** → **Project**. The New Project wizard displays.
2. In the New Project wizard, expand CICS Resources. Click **CICS Bundle Project**, then click **Next**. The CICS Bundle Project wizard displays.

Alternatively, to create a project to contain the files to be referenced by a CICS bundle definition, click the bundle plus (📁) icon in the toolbar to start the new CICS Bundle Project wizard.

3. Enter `general.insurance.customer.applicationservices.to.dev.bundle` as the Project name.
4. Enter `GeneralInsuranceCustomerApplicationServicesDev` as the ID.
5. Enter `1.0.0` as the Version. Figure 6-5 shows the completed new CICS Bundle Project wizard.

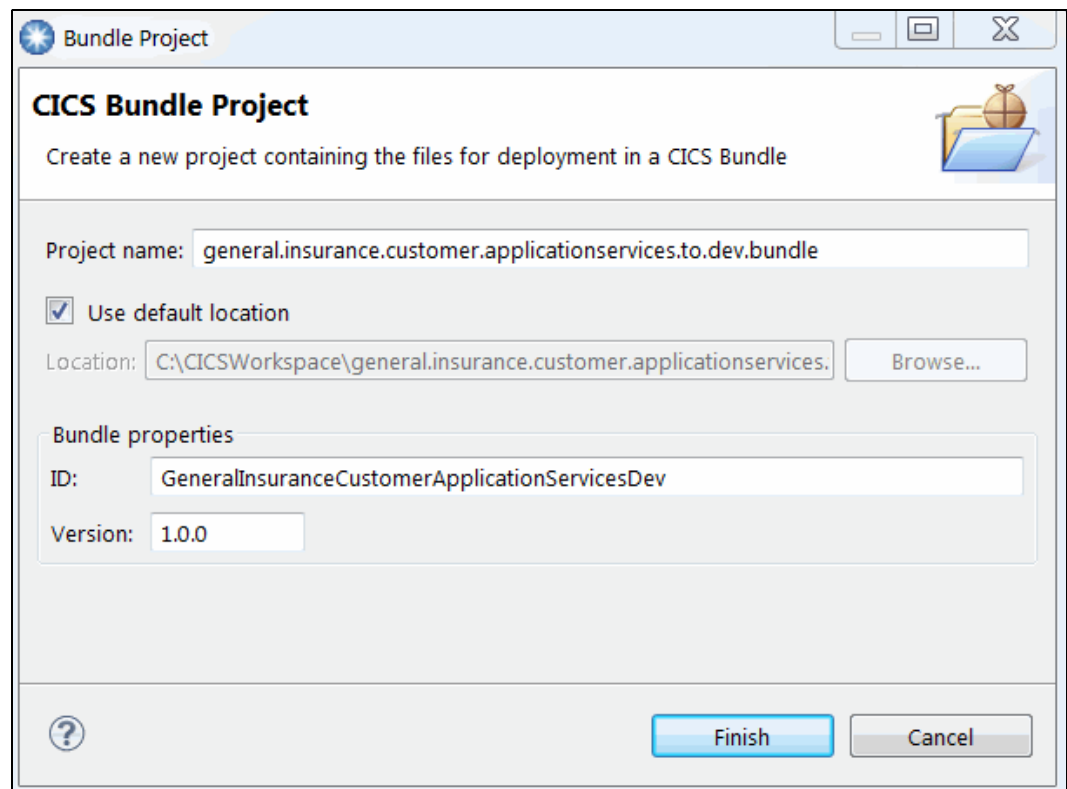


Figure 6-5 Completed new CICS Bundle Project wizard for GeneralInsuranceCustomerApplicationServicesDev

6. Click **Finish**. The bundle project is created and the CICS Bundle Manifest Editor appears. The GeneralInsuranceCustomerApplicationServicesDev CICS Bundle Project has been created. Now create the LIBRARY definition in the CICS bundle using the CICS Bundle Manifest Editor.

7. On the Overview tab in the Defined Resources section, click **New**. A list of CICS TS resources appears. Select **LIBRARY Definition** in the list of CICS TS resources, as shown in Figure 6-6. The New LIBRARY Definition wizard displays.

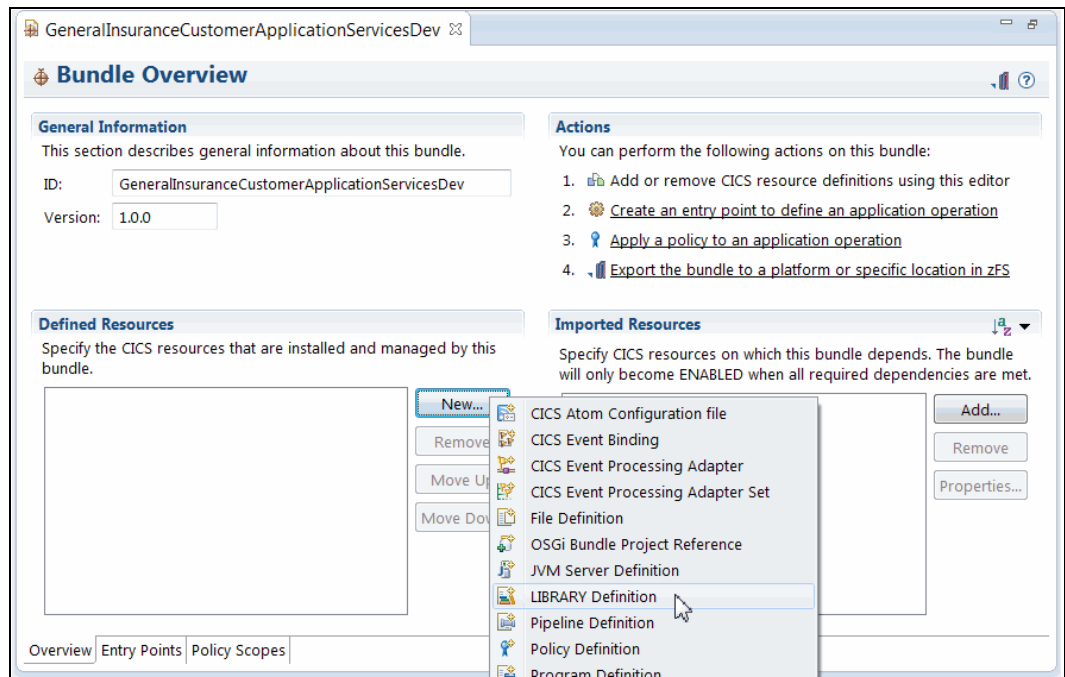


Figure 6-6 Starting the new LIBRARY Definition wizard from the CICS Bundle Manifest Editor

8. Enter the following details in the New LIBRARY Definition wizard:
 - a. Name: GENAAPSV.
 - b. Description: Development load libraries for ApplicationServices.
 - c. Keep the default for Ranking: 50.
 - d. Data set Name 01: <USERHLQ>.CB12.APPLSERV.#1.#0.#2.
 Replace the high-level qualifier <USERHLQ> with the user ID that you used during the installation of GENAPP.
 - e. Clear the **Open editor** check box.

Figure 6-7 shows the completed New LIBRARY Definition wizard.

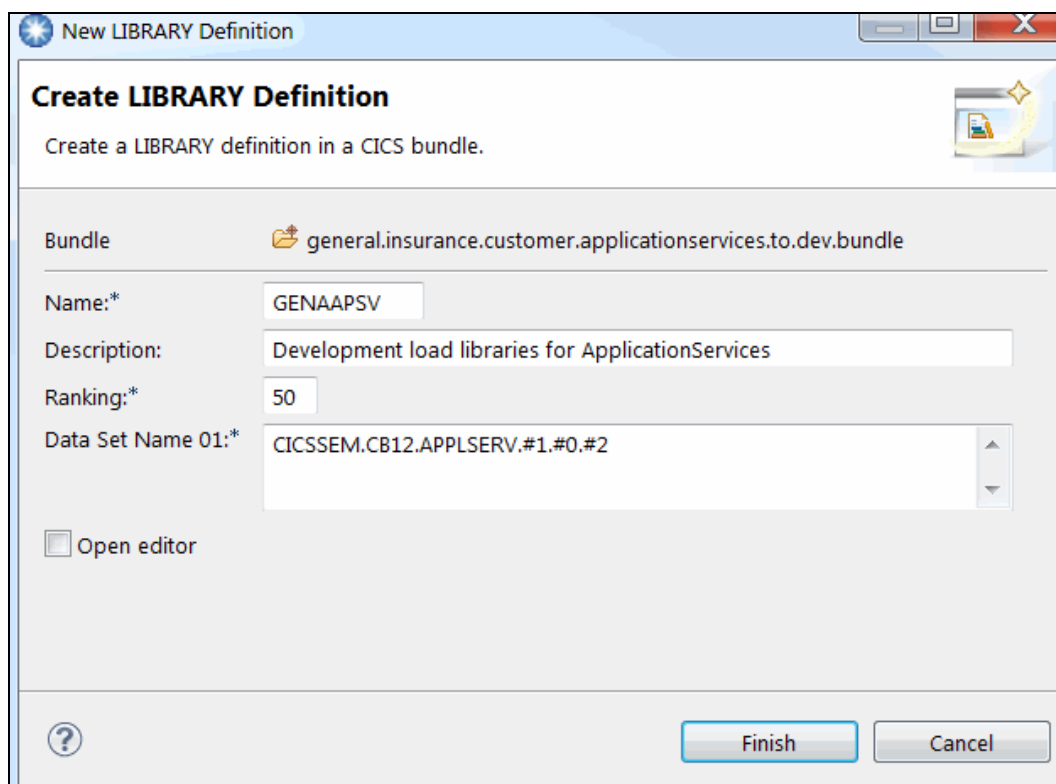


Figure 6-7 Completed New LIBRARY Definition wizard for GENAAPSV

9. Click **Finish**. The Create LIBRARY Definition wizard closes, and the definition is added to the CICS bundle.
10. Press **Ctrl+S** to save changes.

Figure 6-8 shows the CICS Cloud perspective after the creation of the CICS bundle.

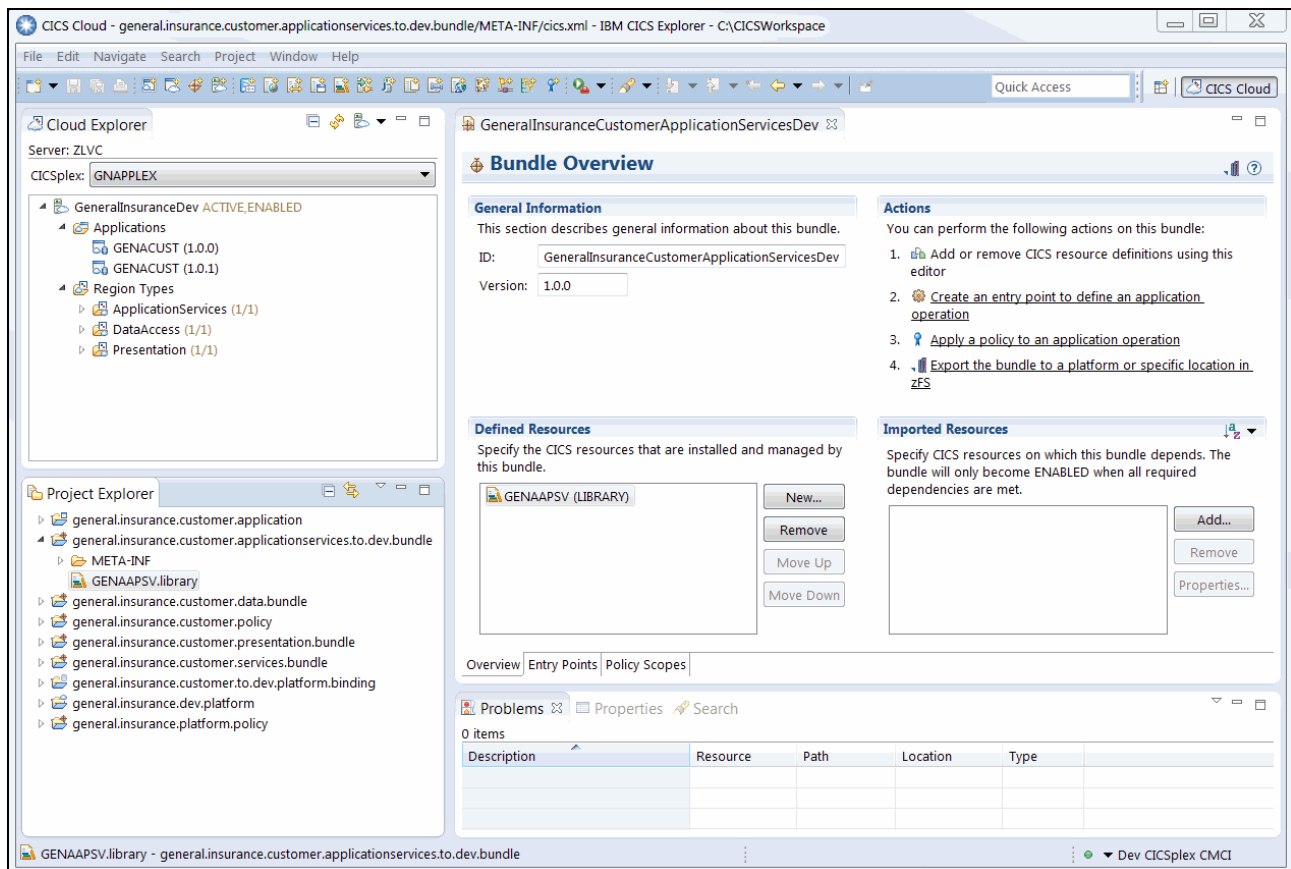


Figure 6-8 The CICS Cloud perspective in CICS Explorer

You have created a new CICS bundle that contains the definition for a dynamic load library. This load library contains the program load modules for the application entry point resources in the ApplicationServices region type. You will later associate this new CICS bundle with the application binding resource.

Tip: During application execution, private dynamic load libraries, such as GENAAPSV, take preference over the system-provided libraries, such as GENALIB. Therefore, it is OK to leave a copy in the system library data set (as opposed to moving the load module), because the application private data sets are prioritized.

6.2.3 Updating CICS TS resources

You can define CICS TS resources within CICS bundles (rather than expressing dependencies on them in the system as you did in Chapter 4, “Creating an application” on page 51). This enables you to manage them together with your application as a single entity.

Perform the following steps to define the program resources for the Application Services layer of the GeneralInsuranceCustomer application:

1. Open the GeneralInsuranceCustomerServices CICS Bundle Manifest Editor.
2. In the Project Explorer view, expand the `general.insurance.customer.services.bundle` project and expand the META-INF subdirectory.

- Then, double-click the `cics.xml` manifest file to start the CICS TS Bundle Manifest editor. The CICS Bundle Manifest Editor for GeneralInsuranceCustomerServices is shown in Figure 6-9.

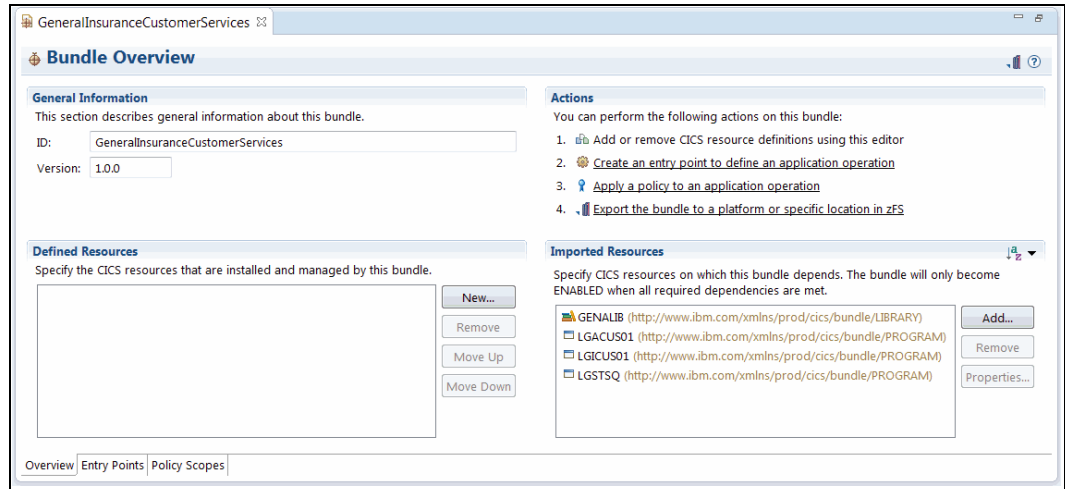


Figure 6-9 The GeneralInsuranceCustomerServices bundle manifest editor expressing numerous dependencies on the system

Currently, the bundle has a dependency on the CICS TS system to provide the program resources. You remove this dependency and define them using the CICS bundle for selected resources.

- At the Entry Points tab in the Entry points section, click the **addCustomer** entry point for the LGACUS01 program resource, as shown in Figure 6-10.

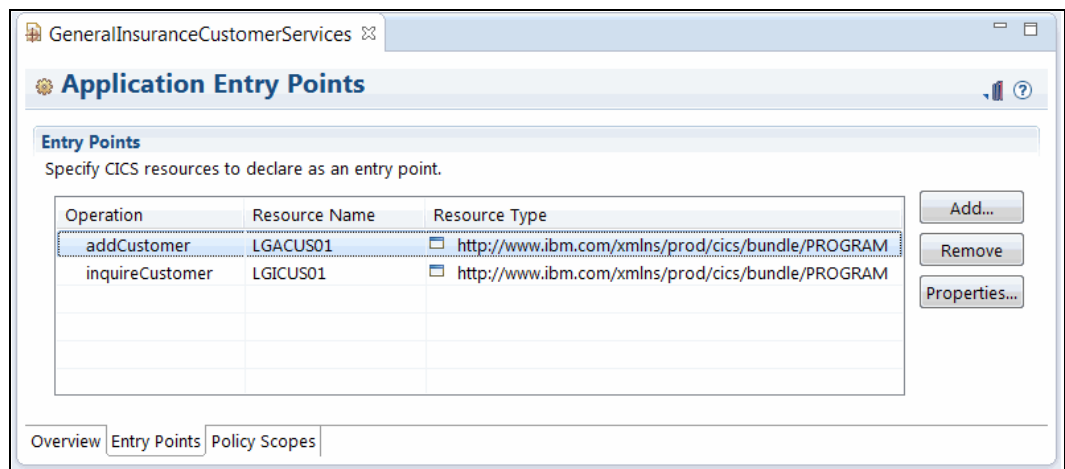


Figure 6-10 The current view of application entry points in the GeneralInsuranceCustomerServices bundle

- Click **Remove**. A confirmation dialog box displays.
- Click **Yes** to remove the selected entry point. The entry point declaration is removed from the CICS bundle.
- Press **Ctrl+S** to save changes.
- On the Overview tab in the Imported Resources section, click the **LGACUS01** program resource.

9. Click **Remove**.

The dependency declaration of the LGACUS01 program is removed from the CICS bundle. You now add the program definition to the CICS bundle.

10. On the Overview tab in the Defined Resources section, click **New**. A list of CICS TS resources that can be defined in a CICS bundle displays.

11. Select **Program Definition** in the list of CICS TS resources, as shown in Figure 6-11. The New Program Definition wizard displays.

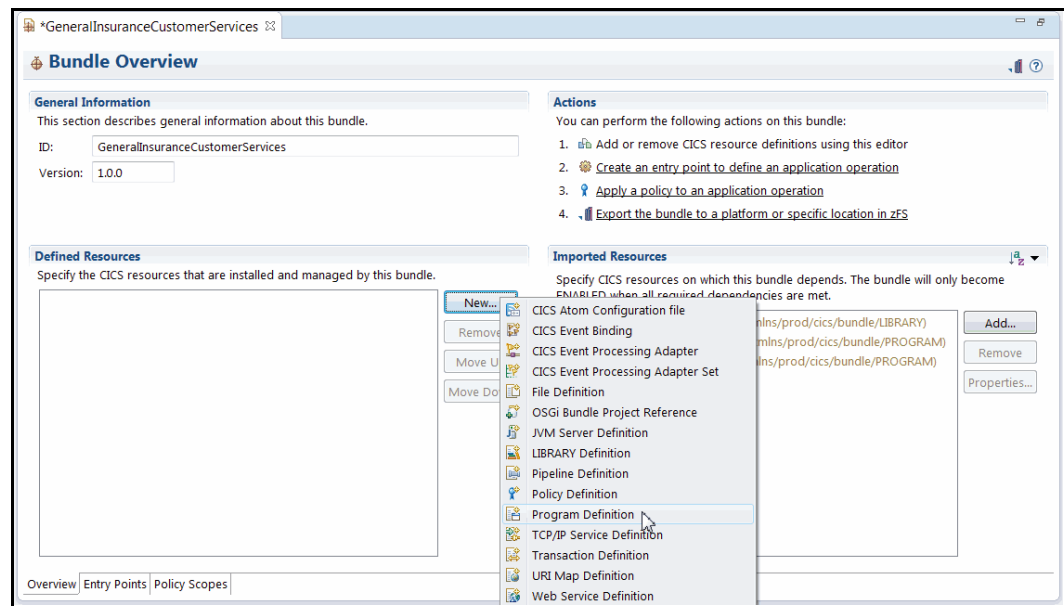


Figure 6-11 Starting the New Program Definition wizard from the bundle manifest editor

12. Enter the following details into the New Program Definition wizard:

- Name: LGACUS01.
- Description: Business logic for adding a new customer.
- Program Type: Assembler, C/C++, COBOL, or PL/I.
- Leave the Remote System and Remote Program Name entries empty.
- Select the **Create an application entry point** check box.
- Operation name: addCustomer.
- Clear the **Open editor** check box.

The completed New Program Definition wizard is shown in Figure 6-12.

New Program Definition

Create a program definition in a CICS bundle.

Bundle: general.insurance.customer.services.bundle

Name*: LGACUS01

Description: Business logic for adding a new customer

Program Type

☒ Assembler, C/C++, COBOL, or PL/I

☐ Java

Service Name:

JVM Server:

Remote System:

Remote Program Name:

☒ Create an application entry point

Operation name*: addCustomer

☐ Open editor

Finish Cancel

Figure 6-12 Completed New Program Definition wizard for program LGACUS01

13. Click **Finish**. The Create Program Definition closes and the definition is added to the CICS bundle.

You have removed the dependency on the CICS bundle for an externally defined program resource, LGACUS01. You have also added a definition for the LGACUS01 program in the CICS Bundle Project. When the bundle is deployed as part of the application, the LGACUS01 program resource is dynamically managed with the application, and does not need to be defined or managed using the CSD.

Repeat the previous steps for the LGICUS01 program:

1. If you have closed the window, open the GeneralInsuranceCustomerServices CICS Bundle Manifest Editor.
2. On the Entry Points tab in the Entry points section, click the **inquireCustomer** entry point for the LGICUS01 program resource.
3. Click **Remove**. A confirmation dialog box displays.
4. Click **Yes** to remove the selected entry point. The entry point declaration is removed from the CICS bundle.
5. Press **Ctrl+S** to save changes.
6. On the Overview tab in the Imported Resources section, click the **LGICUS01** program resource.

7. Click **Remove**. The dependency declaration is removed from the CICS bundle.
8. On the Overview tab in the Defined Resources section, click **New**. A list of CICS TS resources appears. Select **Program Definition** in the list of CICS TS resources. The New Program Definition wizard displays.
9. Enter the following details into the New Program Definition wizard:
 - a. Name: LGICUS01.
 - b. Description: Business logic for inquiring on a customer.
 - c. Program Type: Assembler, C/C++, COBOL, or PL/I.
 - d. Leave the Remote System and Remote Program Name entries empty.
 - e. Select the **Create an application entry point** check box.
 - f. Operation name: inquireCustomer.
 - g. Clear the **Open editor** check box.

The completed New Program Definition wizard is shown in Figure 6-13.

Figure 6-13 Completed New Program Definition wizard for program LGICUS01

10. Click **Finish**. The Create Program Definition closes and the definition is added to the CICS bundle.
11. Press **Ctrl+S** to save the changes.

The LGACUS01 and LGICUS01 programs are loaded from a load library data set specified by the GENAAPSV LIBRARY resource. The GENAAPSV LIBRARY resource was defined in “Creating a CICS bundle and defining a dynamic load library definition” on page 134.

To add a dependency on this dynamic LIBRARY resource in the GeneralInsuranceCustomerServices CICS bundle, perform the following steps:

1. If you have closed the window, open the GeneralInsuranceCustomerServices CICS Bundle Manifest Editor.
2. On the Overview tab in the Imported Resources section, click **Add**. The Create Bundle Import dialog displays.
3. In the Create Bundle Import dialog box for the **Resource Type**, type LIBRARY. As you type, a drop-down list of matching resources appears. In the list, double-click **LIBRARY** to supply the fully qualified resource type name.
4. In the Name field, type GENAAPSV. The completed Create Bundle Import dialog is shown in Figure 6-14.

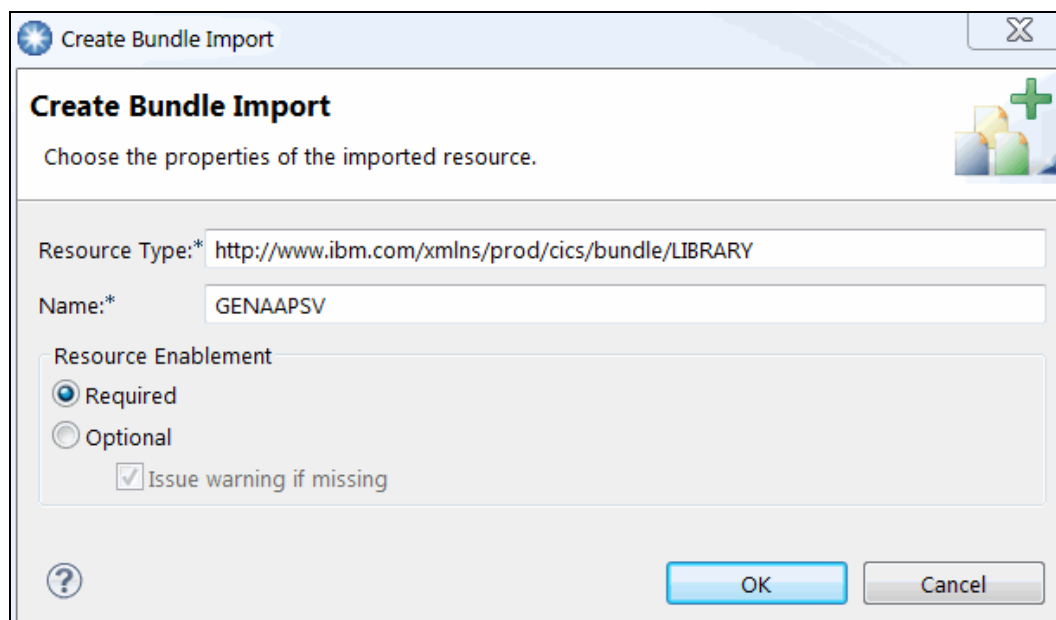


Figure 6-14 Creating a bundle dependency on LIBRARY resource GENAAPSV

5. Click **OK**. The dialog window closes and the resource dependency is added to the CICS bundle.
6. Press **Ctrl+S** to save changes.

Note: You might notice that the dependency for the LGSTSQ program still exists in the CICS bundle, and has not been replaced with a definition for the program resource. This is because, even though it is required in the Application Services region type, it is a shared program resource and, as such, you do not want to tie its lifecycle to that of the other resources in the CICS bundle.

This is an example of a CICS bundle that can define multiple resources and manage those resources as a single entity, while also expressing a dependency on other externally required resources while enabling them to be managed separately.

You have now updated the GeneralInsuranceCustomerServices CICS bundle to define two CICS TS program resources, rather than expressing them as dependencies on the CICS TS system.

As a result of modifying the CICS bundle, you will now update the bundle version by performing the following steps:

1. If you have closed the window, open the GeneralInsuranceCustomerServices CICS Bundle Manifest Editor.
2. On the Overview tab in the General Information section, update the version from 1.0.0 to 1.0.1.
3. Press **Ctrl+S** to save changes.

The CICS bundle version has been updated to track changes made to the bundle. You might notice that errors are reported after the version change to the CICS bundle. This is because the application and application binding projects are expecting a different version of the CICS bundle. The next steps will rectify these errors.

Figure 6-15 shows the updated Overview tab of the GeneralInsuranceCustomerServices CICS bundle.

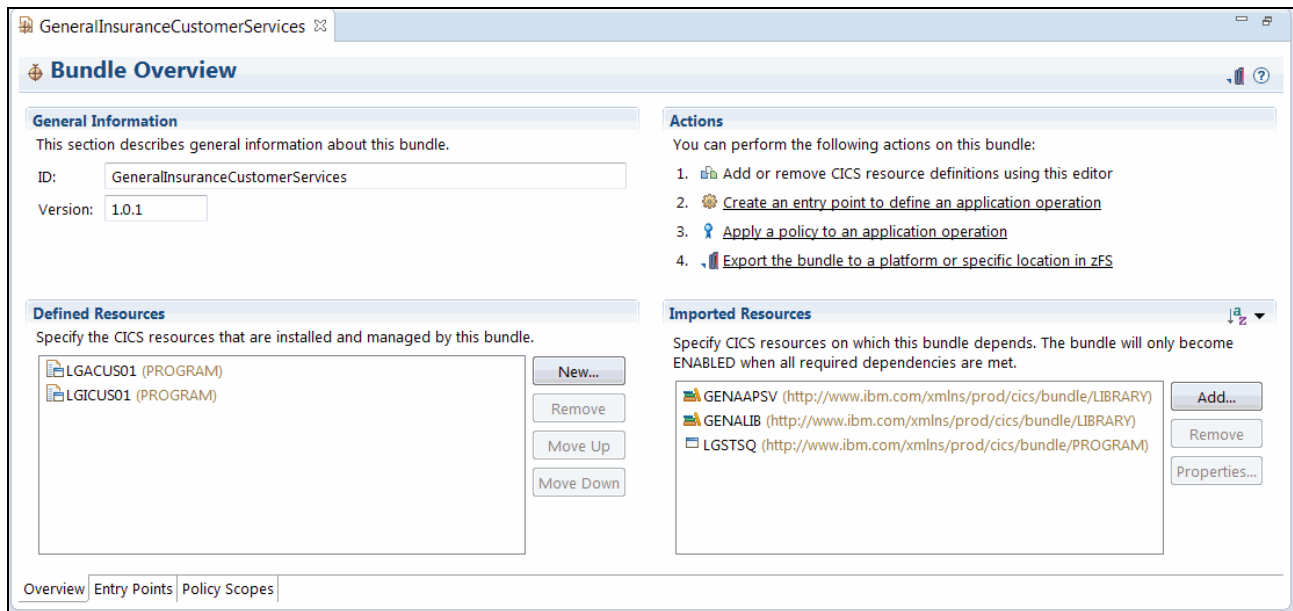


Figure 6-15 GeneralInsuranceCustomerServices bundle with two program definitions

Figure 6-16 shows the updated Entry Points tab of the GeneralInsuranceCustomerServices CICS bundle.

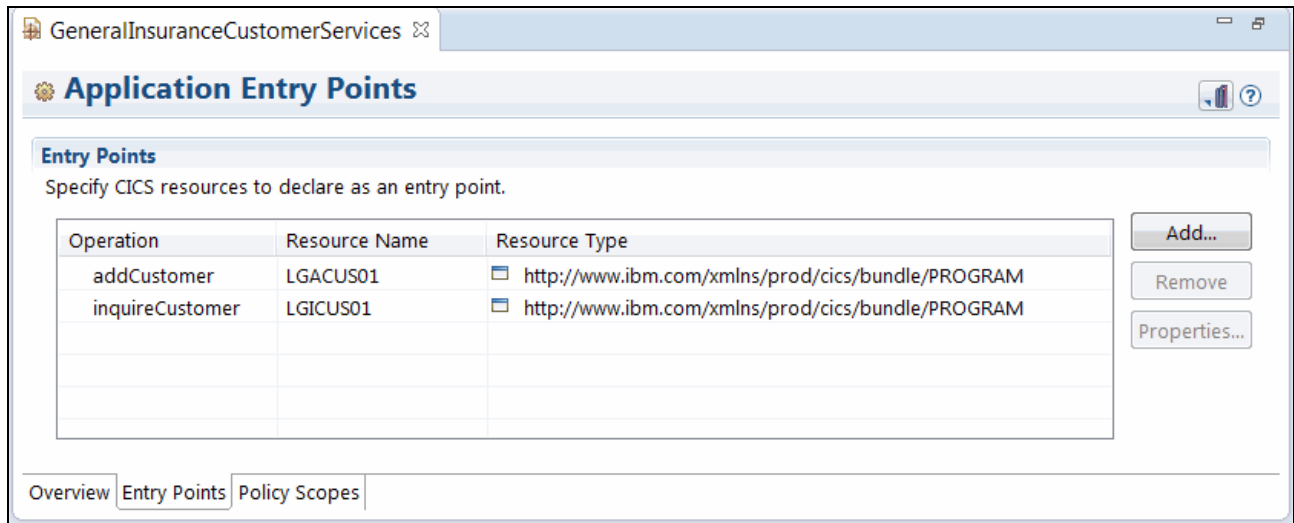


Figure 6-16 GeneralInsuranceCustomerServices bundle with two program application entry point declarations

6.2.4 Updating the application version

In this example, we are providing a micro-level version update to the CICS TS application. In the example steps so far, you have created a new CICS bundle and updated the version of another CICS bundle. In this section, you update the version of the application to reflect the changes that have been made, and include the new CICS bundle into the application binding.

Perform the following steps to the application to update the references of the CICS bundle which was modified earlier:

1. In the Project Explorer view, expand the `general.insurance.customer.application` application project. Then, expand the `META-INF` subdirectory and double-click `bundles.xml` to open the Application editor.
2. In the CICS bundles section, click **GeneralInsuranceCustomerServices (1.0.0)** and click **Remove**, as shown in Figure 6-17. The prior version of the bundle is disassociated from the CICS TS application.

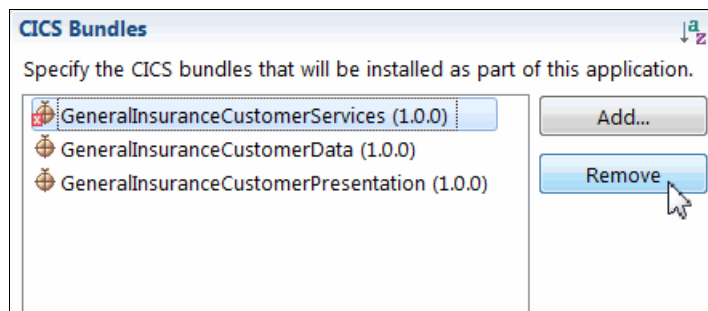


Figure 6-17 Removing a CICS bundle association using the CICS bundles section of the Application Editor

3. In the CICS bundles section, click **Add**. The CICS Bundle Selection dialog displays.

4. In the CICS Bundle Selection dialog, select **GeneralInsuranceCustomerServices (1.0.1)**, as shown in Figure 6-18.

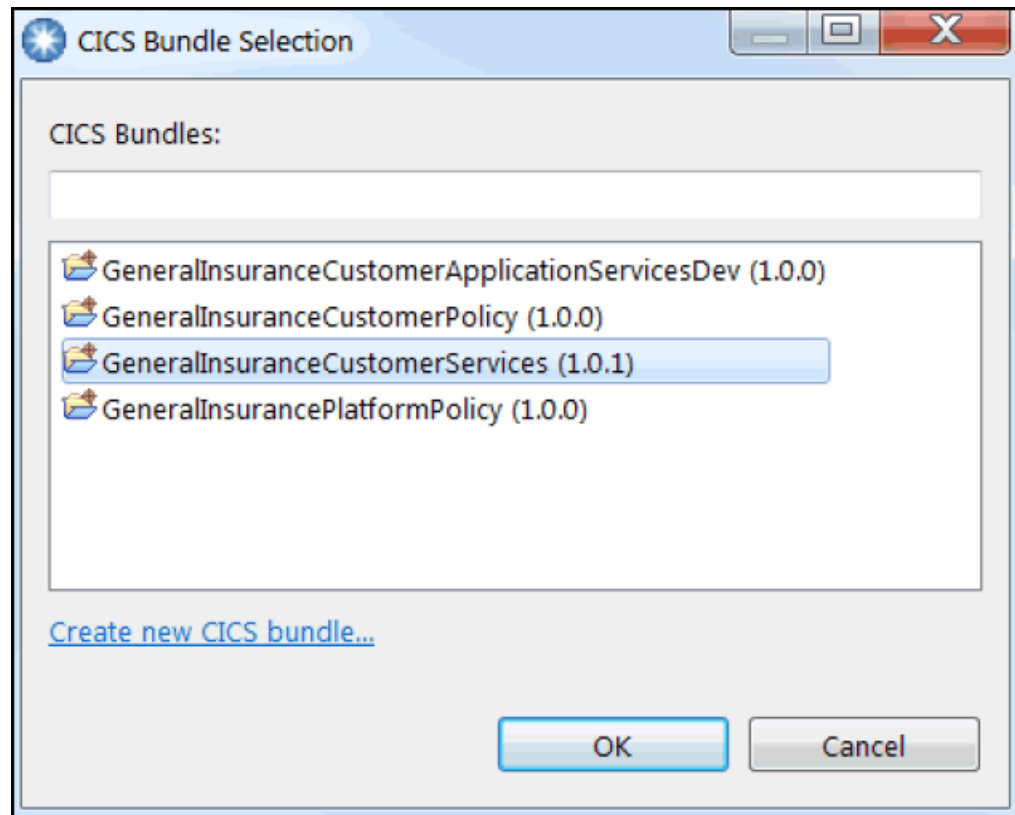


Figure 6-18 Selecting the updated CICS bundle in the CICS Bundle Selection dialog

5. Click **OK**. The CICS Bundle Selection dialog closes and returns to the Application Editor. The updated version of the bundle is associated with the CICS TS application.
The GeneralInsuranceCustomerServices bundle version change has been reflected in the CICS TS application GeneralInsuranceCustomer.
Next, update the version of the CICS TS application to indicate that changes have occurred to the CICS TS application
6. In the GeneralInsuranceCustomer application editor, increment the micro level of the CICS TS application. In the Application Editor, in the General Information section, change the value of the Version from 1.0.1 to 1.0.2.

7. Press **Ctrl+S** to save changes. Figure 6-19 shows the updated application editor view.

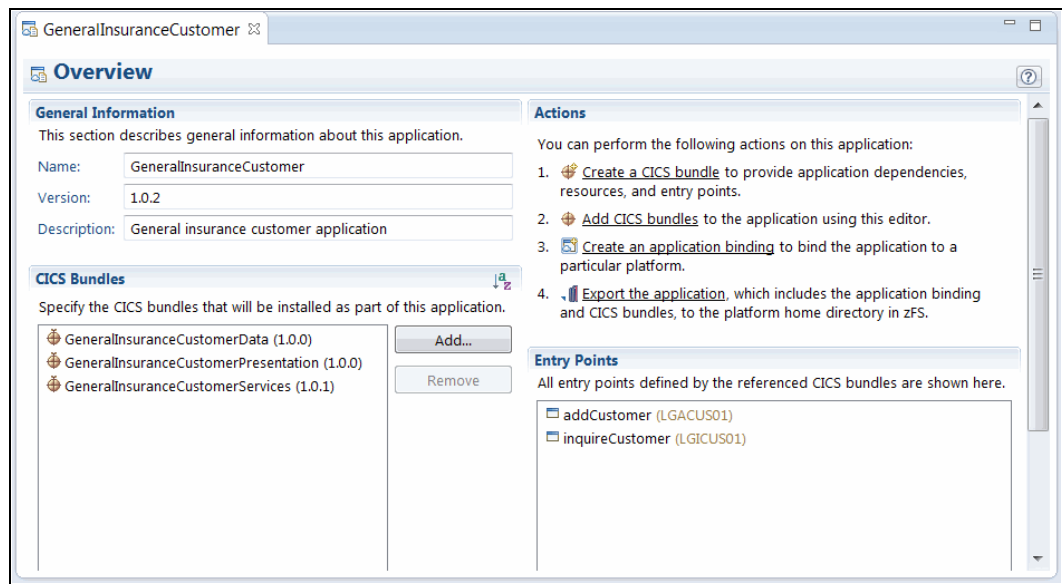


Figure 6-19 Updated GeneralInsuranceCustomer version 1.0.2 application editor view

The GeneralInsuranceCustomerServices bundle version has been updated, and so has the references to the bundle in the GeneralInsuranceCustomer application project. The workspace still displays errors in the GeneralInsuranceCustomerToDev application binding due to references of the CICS bundle and application prior versions.

You will now update the binding project to perform the following actions:

- ▶ Reference the updated version of the application project (version 1.0.2).
- ▶ Add the new CICS bundle to the application binding.
- ▶ Provide the deployment rules for the new and updated CICS bundles.
- ▶ Update the version of the application binding project to reflect the changes being made.

Perform the following steps to reference the updated CICS TS application version:

1. In the Project Explorer view, expand the general.insurance.customer.to.dev.platform.binding application binding project. Then, expand the META-INF subdirectory and double-click appbinding.xml to open the application binding editor.
2. In the General Information section, click **Browse** next to Application: GeneralInsuranceCustomer (1.0.1). The Application Selection dialog displays.
3. In the Application Selection dialog, select **GeneralInsuranceCustomer (1.0.2)** and click **OK**.
4. Press **Ctrl+S** to save changes.

Perform the following steps to add the new CICS bundle created in “Creating a CICS bundle and defining a dynamic load library definition” on page 134:

1. In the GeneralInsuranceCustomerToDev application binding editor on the Overview tab in the CICS bundles section, click **Add**. The CICS Bundle Selection dialog displays.
2. In the CICS Bundle Selection dialog, select **GeneralInsuranceCustomerApplicationServicesDev (1.0.0)** and click **OK**.
3. Press **Ctrl+S** to save changes.

Important: The GeneralInsuranceCustomerApplicationServicesDev CICS bundle contains the definition for a LIBRARY resource. It is likely that the data sets associated with the LIBRARY resource will require updating when the application is promoted through development through to production. Therefore, the bundle is added to the application binding rather than the application.

Perform the following steps to provide the deployment rules for the updated GeneralInsuranceCustomerServices bundle and the new GeneralInsuranceCustomerApplicationServicesDev bundle:

1. In the Application Binding Editor, click the **Deployment** tab.
2. In the Region Types section, select the **ApplicationServices** region type.

Notice that there are two errors shown in the list of CICS bundles associated with the GeneralInsuranceCustomerServices CICS bundle. The error marking on the GeneralInsuranceCustomerServices (1.0.0) bundle is due to that version of the bundle being disassociated with the application. The error marking on the GeneralInsuranceCustomerServices (1.0.1) bundle is because there are no deployment rules associated with that version of the bundle.
3. In the list of CICS bundles, *clear* the selection box next to GeneralInsuranceCustomerServices (1.0.0). That version of the bundle is not referenced in the application or the binding, and will be removed from the list.
4. In the list of CICS bundles, *select* the box next to GeneralInsuranceCustomerServices (1.0.1). The updated version of the bundle is now associated with the ApplicationServices region type.
- You have updated the deployment reference of the GeneralInsuranceCustomerServices CICS bundle. There is still an error reported due to the fact that there is no deployment rule specified for the GeneralInsuranceCustomerApplicationServicesDev CICS bundle.
5. In the list of CICS bundles, select the box next to GeneralInsuranceCustomerApplicationServicesDev (1.0.0). The deployment rule for the new CICS bundle to the ApplicationServices region type has been created.
6. Press **Ctrl+S** to save changes.

Figure 6-20 shows the updated deployment rules for the ApplicationServices region type.

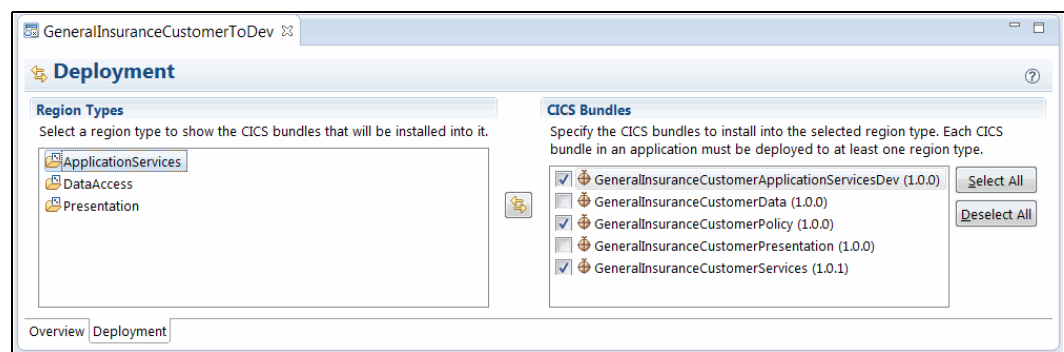


Figure 6-20 GeneralInsuranceCustomerToDev application binding deployment rules for ApplicationServices region type

The references to the correct versions of the bundle and application have been made in the application binding. The bundle containing the LIBRARY definition has been added to the application binding.

To track the changes made in the application binding, you increment the application binding version by performing the following steps:

1. In the GeneralInsuranceCustomerToDev Application Binding Editor, on the Overview tab in the General Information section, change the Version from 1.0.1 to 1.0.2.
2. Press **Ctrl+S** to save changes.

The updated application binding editor is shown in Figure 6-21.

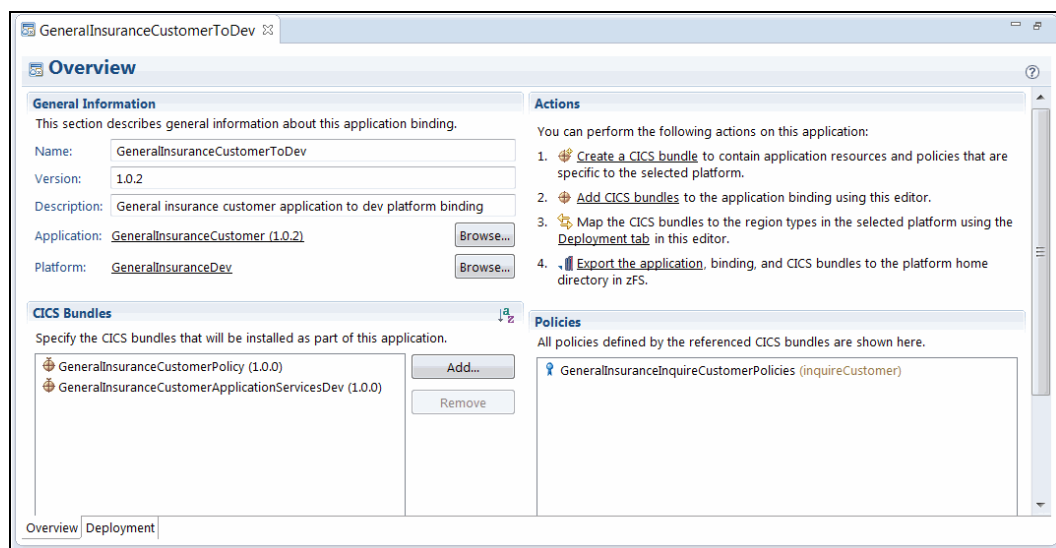


Figure 6-21 Updated version 1.0.2 of application binding GeneralInsuranceCustomerToDev

6.2.5 Exporting, installing, and enabling the application

In this example, you have updated the GeneralInsuranceCustomer CICS TS application. You provided a micro version update to the CICS TS application, and in the process repackaged the CICS TS resources related to the AOR (ApplicationServices) region type. In the previous version of GeneralInsuranceCustomer, the ApplicationServices program resources were provided through the CSD and had to be managed separately from the application.

In the updated version of the application, with the program resources packaged with the application, the programs can be managed together with the application as a single entity. The relationship between the programs and the application is kept, from the Explorer modeling environment, through source code management, and then through to lifecycle deployment and management in a CICS TS system.

Perform the follow steps to export the Explorer projects and install the application into the CICS TS platform:

1. Using the CICS Explorer CICS Cloud perspective, in the Project Explorer view, right-click `general.insurance.customer.application` and select **Export Application Project to z/OS UNIX File System**. The Export Application to the home directory of a platform wizard displays.
2. In the wizard, ensure that the correct CICS Management Client Interface (CMCI) connection is specified.
3. Select the GeneralInsuranceCustomerToDev (1.0.2) application binding.

4. Select the **Create Application Definition after export finishes** check box, as shown in Figure 6-22.

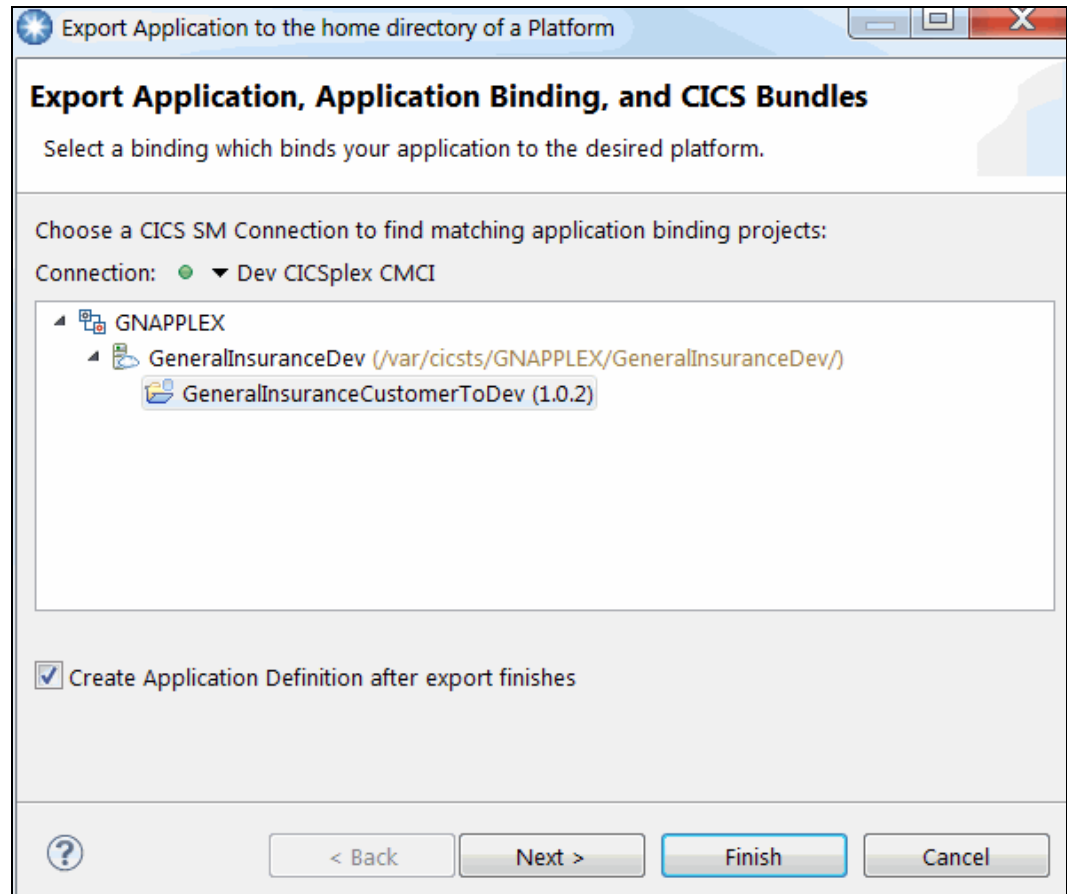


Figure 6-22 Export application GeneralInsuranceCustomer version 1.0.2

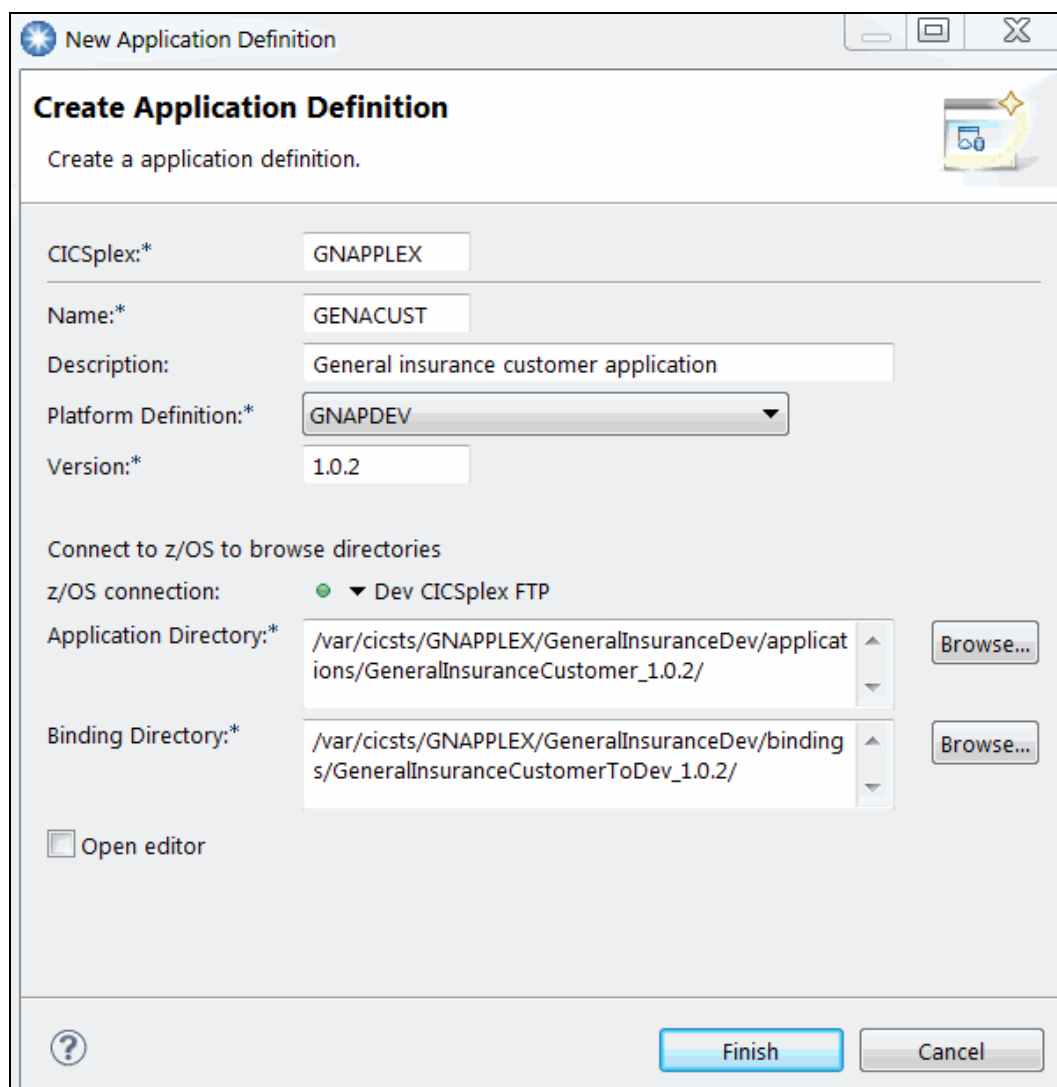
5. Click **Next**.
6. Ensure that the correct File Transfer Protocol (FTP) connection is specified.
7. Click **Finish**.

The application, application binding, and required CICS bundles are transferred to the IBM z/OS UNIX file system. The export process transfers files in the correct format, and puts them in the correct subfolders in the platform home directory.

When the export process is complete, the New Application Definition dialog appears.

8. Specify the following details in the New Application Definition dialog:
 - a. Name: GENACUST.
 - b. Description: General insurance customer application.
 - c. Keep the supplied values for the platform definition, version, application directory, and the binding directory. These values have been completed by the wizard using the values specified by the exported artifacts.
 - d. Clear the **Open editor** check box.

Figure 6-23 shows the completed New Application Definition dialog.



The image shows a 'New Application Definition' dialog box. The title bar says 'New Application Definition'. The main heading is 'Create Application Definition' with a subtext 'Create a application definition.' and a small icon of a folder with a star. The dialog contains several fields: 'CICSplex:*' with value 'GNAPPLEX', 'Name:*' with value 'GENACUST', 'Description:' with value 'General insurance customer application', 'Platform Definition:*' with a dropdown menu showing 'GNAPDEV', and 'Version:*' with value '1.0.2'. Below these is a section 'Connect to z/OS to browse directories' with 'z/OS connection:' set to 'Dev CICSplex FTP'. There are two text boxes for directory paths: 'Application Directory:*' with value '/var/cicsts/GNAPPLEX/GeneralInsuranceDev/applications/GeneralInsuranceCustomer_1.0.2/' and 'Binding Directory:*' with value '/var/cicsts/GNAPPLEX/GeneralInsuranceDev/bindings/GeneralInsuranceCustomerToDev_1.0.2/'. Each has a 'Browse...' button to its right. At the bottom left is an 'Open editor' checkbox. At the bottom right are 'Finish' and 'Cancel' buttons.

Create Application Definition
Create a application definition.

CICSplex:* GNAPPLEX

Name:* GENACUST

Description: General insurance customer application

Platform Definition:* GNAPDEV

Version:* 1.0.2

Connect to z/OS to browse directories

z/OS connection: Dev CICSplex FTP

Application Directory:* /var/cicsts/GNAPPLEX/GeneralInsuranceDev/applications/GeneralInsuranceCustomer_1.0.2/ Browse...

Binding Directory:* /var/cicsts/GNAPPLEX/GeneralInsuranceDev/bindings/GeneralInsuranceCustomerToDev_1.0.2/ Browse...

☐ Open editor

Finish Cancel

Figure 6-23 Completed dialog to create the application definition for version 1.0.2 of GeneralinsuranceCustomer application

9. Click **Finish** in the new Application Definition dialog.

Figure 6-24 shows the updated Cloud Explorer view after the creation of the application definition.

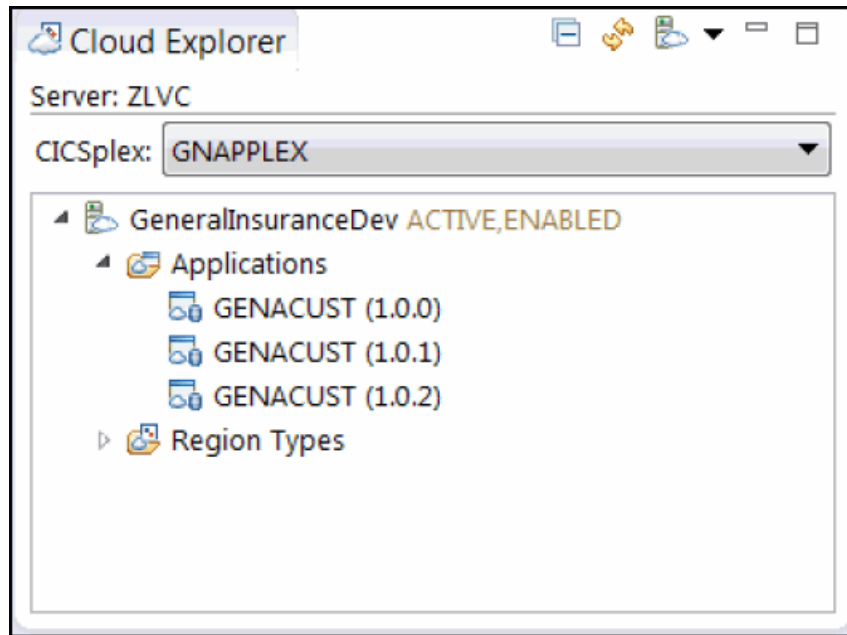


Figure 6-24 Cloud Explorer view displaying the application definition for version 1.0.2

The application-related projects have been exported to the z/OS file system, and an application definition has been created to instruct CICS TS on their location in the file system. Next, install the application definition and enable the application.

10. In the Cloud Explorer view, expand the GeneralInsuranceDev platform and expand the Applications subfolder to view the installed applications and application definitions, as shown in Figure 6-24.
11. Right-click the **GENACUST (1.0.2)** application definition and select **Install**. The Perform Operation dialog displays.
12. Click **OK** to install the application.

The application is installed to the platform CICS TS regions. This includes the installation of the program definitions that are now included in the CICS TS application. The application installs into a disabled state. If the state displays a state other than disabled, wait a short amount of time and refresh the Cloud Explorer view.

The Cloud Explorer view updates to display the installed application rather than the application definition for version 1.0.2, as shown in Figure 6-25.

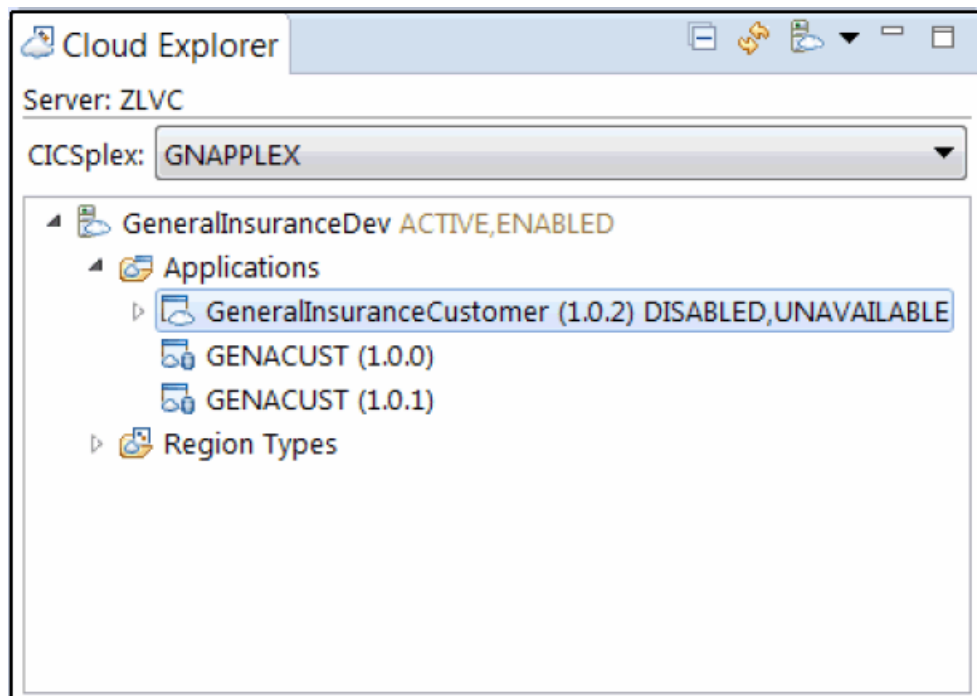


Figure 6-25 Cloud Explorer view showing installed application version 1.0.2

13. Right-click the **GeneralInsuranceCustomer (1.0.2)** application and select **Enable**. The Perform Operation dialog appears. Click **OK** to enable the application.

Figure 6-26 shows the installed and enabled CICS TS application.

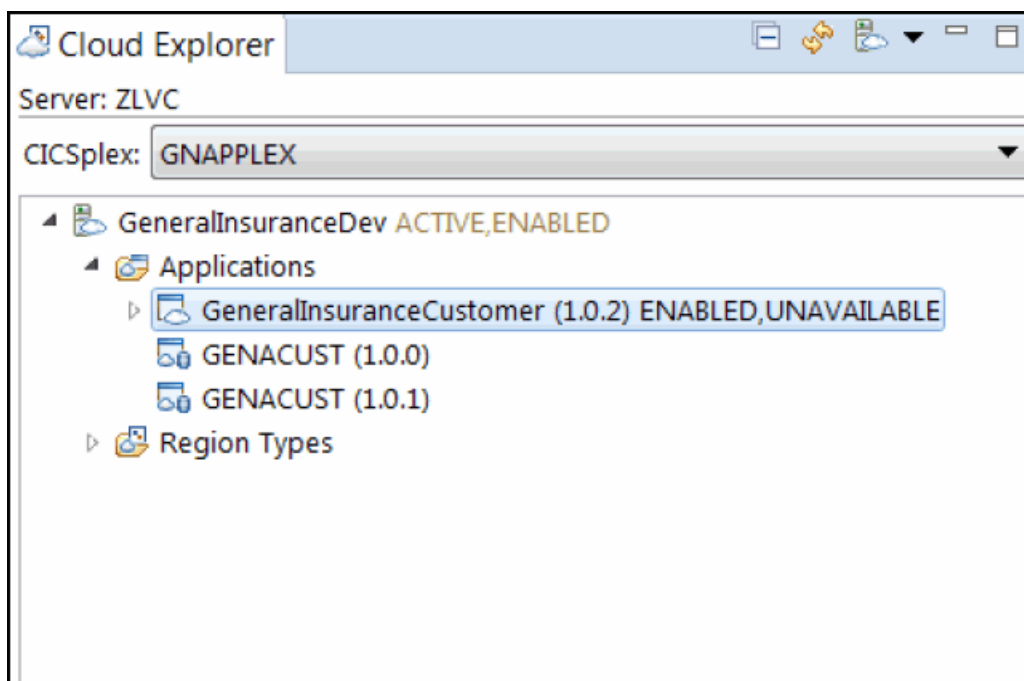


Figure 6-26 Cloud Explorer view after the installation and enablement of application version 1.0.2

You have repackaged the ApplicationServices region type resources from the CSD into the CICS TS application. The updated application has been installed into the platform, and has been enabled. This also installs and enables the application program and LIBRARY resources in the application. As the deployer of the application, you have the ability to check that the enablement has been successful before allowing users to access the application.

6.2.6 Making the application available and running it

The application has been installed and enabled on the CICS TS platform. It is currently in an unavailable state, which ensures that the application cannot be started through its application entry points. The programs and LIBRARY resources packaged with the application are private to the application and cannot be linked to.

Perform the following steps to make the application available to users:

1. In the Cloud Explorer view, right-click the **GeneralInsuranceCustomer (1.0.2)** application and select **Make Available**. The Perform Operation dialog displays.
2. Click **OK** to make the application available through the application entry points.

The Cloud Explorer view updates, and the GeneralInsuranceCustomer application version 1.0.2 displays an enabled and available state, as shown in Figure 6-27.

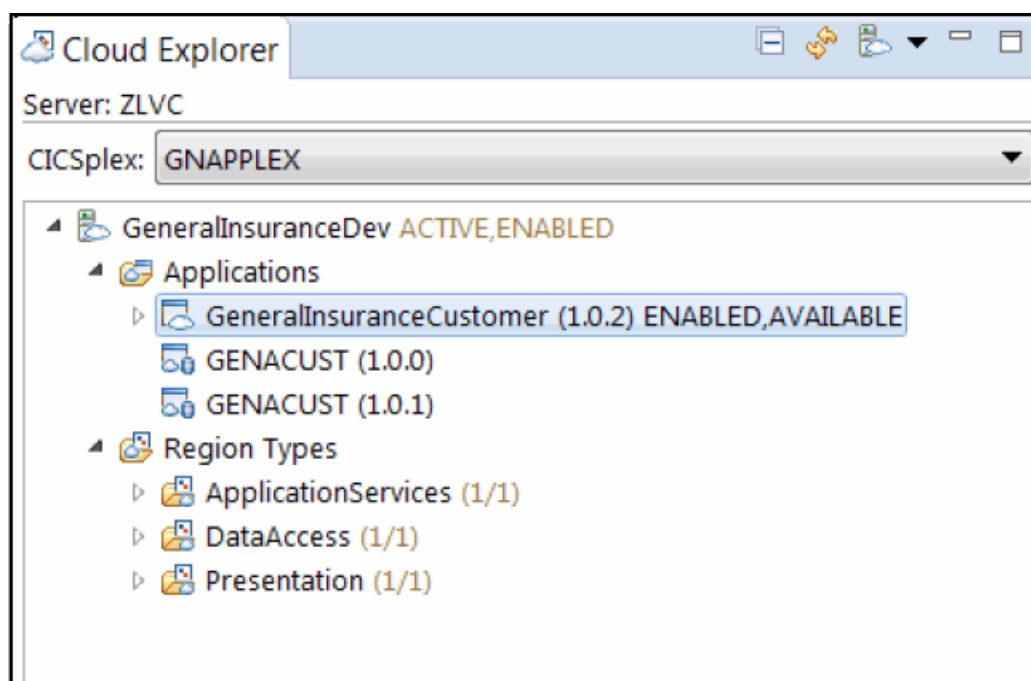


Figure 6-27 Application GeneralInsuranceCustomer is now available to end-users

The updated version of the GeneralInsuranceCustomer CICS TS application has now been deployed. Along with the application actions, the LIBRARY and program resources defined in the application have also been deployed, ready for use.

Perform the following steps to test the deployment:

1. At a CICS TS terminal session, enter the transaction SSC1. The GENAPP application displays.
2. Enter a valid customer number. For example, Cust Number: 0000000001.

3. Select Option 1 for a customer inquiry action and submit.

The panel updates with the customer details for the inquired customer, as shown in Figure 6-28.

```
SSC1      General Insurance Customer Menu

1. Cust Inquiry      Cust Number      0000000001
2. Cust Add          Cust Name :First  Andrew
                    :Last           Pandy
4. Cust Update       DOB              1950-07-11  (yyyy-mm-dd)
                    House Name
                    House Number      34
                    Postcode          PI10100
                    Phone: Home       01962 811234
                    Phone: Mob        07799 123456
                    Email Addr        A.Pandy@beebhouse.com

Select Option  1
```

Figure 6-28 The GeneralInsuranceCustomer application being run at a terminal

The application has been successfully deployed.

6.3 Example 2: Provisioning a logic change to the GeneralInsuranceCustomer CICS TS application

In 6.2, “Example 1: Repackaging CICS TS resources into the GeneralInsuranceCustomer CICS TS application” on page 128, you deployed an update to the application, version 1.0.2. In version 1.0.2, you packaged the program definitions for the application entry points, and provided a private load library for their load modules. This prepares the AOR layer for application multiversioning, and provides a single point of control for the application with the management of all of the packaged resources.

In this second example, you provide a coding update to the application. This new version of the application is compatible with a previous version, yet provides a functional change to the user. Therefore, the update is a minor version upgrade. You deploy this updated version alongside the prior version of the application. You deploy the updated version of the application with no loss-of-service to the user.

6.3.1 Fixing the bug and compiling

You step through the process of a logic change to the GENAPP application.

In this example scenario, your company has mandated a stricter control of all clients’ personal data. The inquire customer operation in the GeneralInsuranceCustomer application has the ability to retrieve customer-sensitive data to the operator. The operator needs to obtain the address details for the customer, but the business considers the date of birth field as personal, sensitive data that should not be displayed.

In this example, you provide an update to the source code for the CICS TS application to mask the customer date of birth. You then compile and provide the program load modules through a minor update to the GeneralInsuranceCustomer CICS TS application as version 1.100.0. You deploy the updated version of the application with no application service downtime, alongside the older version of the application.

Note on semantic versioning: You might have noticed that the version of the application has been proposed to change from 1.0.2 to 1.100.0. There are several reasons for this:

- ▶ The major version part, 1, remains the same, because the application maintains the same interfaces and is compatible with an earlier version.
- ▶ The minor version changes from 0 to 100. The version update would also be valid if we changed the minor version from 0 to 1. However, leaving a gap between the numbers provides you the flexibility to add minor version updates in between if needed. For example, during an application bug fix to an earlier version, the update might force an interface change, resulting in a minor version update.
- ▶ The micro version will reset to 0, because this is the first version at 1.100.<n>.
- ▶ Providing a minor version update enables you to provision both applications concurrently in CICS TS V5.2, and later by using the application multi-versioning support.

For more details about semantic versioning, see 9.1.1, “Semantic versioning” on page 194.

Note for the following steps: The following instructions assume that you have installed GENAPP per the instructions in Appendix A, “Setup and environment” on page 207. Where appropriate, replace the high-level qualifier <USERHLQ> with the user ID that you used during the installation of GENAPP.

The following steps update the source code and job control language (JCL) files for GENAPP. It is advisable to take a backup copy of the following files before performing the following steps. You can then restore the files to their initial state after you have completed this example:

- ▶ <USERHLQ>.CB12.SOURCE(LGICUS01)
- ▶ <USERHLQ>.CB12.CNTL(@COBOL)

Perform the following steps to update and compile the required GENAPP source code into a new data set load library:

1. Allocate the following load library, replacing <USERHLQ> with the high-level qualifier that you used for the installation of GENAPP. You can base the data set characteristic rules on the existing <USERHLQ>.CB12.LOAD data set, <USERHLQ>.CB12.APPLSERV.#1.#100.#0.

You have now created a load library data set to contain the program load modules for the updated programs in the GeneralInsuranceApplication V1.100.0 CICS TS application.

Next, update the source code for the inquireCustomer operation to mask the date of birth (DOB) field.

2. Open the <USERHLQ>.CB12.SOURCE member (LGICUS01) for editing.
3. Directly after the **PERFORM GET-CUSTOMER-INFO** statement, insert the following Common Business Oriented Language (COBOL) statement, at approximately line 135:

```
MOVE '****_**_**' TO CA-DOB.
```

Example 6-1 shows an extract of the updated source code for the LGICUS01 program.

Example 6-1 Update to the LGICUS01 program to mask the date of birth field

```

*-----*
* Process incoming commarea                                *
*-----*
* check commarea length
  MOVE WS-CUSTOMER-LEN      TO WS-REQUIRED-CA-LEN
  ADD WS-CA-HEADERTRAILER-LEN TO WS-REQUIRED-CA-LEN
  IF EIBCALEN IS LESS THAN WS-REQUIRED-CA-LEN
    MOVE '98' TO CA-RETURN-CODE
    EXEC CICS RETURN END-EXEC
  END-IF

  MOVE CA-CUSTOMER-NUM TO EM-CUSNUM

  PERFORM GET-CUSTOMER-INFO.

  MOVE '****_**_**' TO CA-DOB.

```

4. Save the update to the LGICUS01 member.

The source code for the inquireCustomer operation has been updated. You will now update the compilation JCL to compile the updated code into the new load library.

5. Open the <USERHLQ>.CB12.CNTL(@COBOL) member for editing.
6. In the @COBOL file, replace the two instances of <USERHLQ>.CB12.LOAD with <USERHLQ>.CB12.APPLSERV.#1.#100.#0.
7. In the @COBOL file, delete the final block of statements to process the various program files, leaving just the following single entry:

```
//LGICUS01 EXEC DB2PROC,MEM=LGICUS01
```

Example 6-2 shows an extract of the final section of the @COBOL JCL file.

Example 6-2 Update to JCL @COBOL to only compile the updated LGICUS01 COBOL file

```

//*-----*
//* start THE MVS LINKAGE-EDITOR PROGRAM                      *
//*-----*
//*
//LKED      EXEC PGM=HEWL,COND=(7,LT,COBL),
//  PARM='LIST,XREF,RENT,NAME=&MEM'
//SYSLIB    DD DISP=SHR,DSN=CTS520.CICS690.SDFHLOAD
//          DD DSN=SYS2.DB2.V11.SDSNLOAD,DISP=SHR
//          DD DSN=CEE.SCEELKED,DISP=SHR
//          DD DISP=SHR,DSN=CICSSEM.CB12.APPLSERV.#1.#100.#0
//SYSLMOD   DD DISP=SHR,DSN=CICSSEM.CB12.APPLSERV.#1.#100.#0(&MEM)
//SYSUT1    DD UNIT=SYSDA,DCB=BLKSIZE=1024,
//          SPACE=(CYL,(1,1))
//SYSPRINT  DD SYSOUT=*
//SYSLIN    DD DISP=(OLD,DELETE),DSN=&&COPYLINK
//          DD DISP=(OLD,DELETE),DSN=&&LOADSET
//          DD DISP=SHR,DSN=CICSSEM.CB12.SOURCE(LINKPARM)

```

```
//          PEND
//*
//*
//LGICUS01 EXEC DB2PROC, MEM=LGICUS01
```

8. Save the update to the @COBOL member.
9. Submit the @COBOL JCL to recompile the updated LGICUS01 program into the <USERHLQ>.CB12.APPLSERV.#1.#100.#0 data set load library. This job has a return code of 4.

You have now provided an updated program load module, in a new load library location, for reference by the new version of the CICS TS application. Note that you have only provided the programs that have been modified in the new data set, and the CICS TS application can load the unchanged modules from the prior data set locations.

6.3.2 Updating data set name in LIBRARY

In this example, you have coded an update to the GENAPP application source code. The updated program algorithm hides the customer's date of birth during the customer inquiry function. The updated program (LGICUS01) was compiled, and the program load module was supplied in a new data set location (<USERHLQ>.CB12.APPLSERV.#1.#100.#0).

Perform the following steps to add the load library data set to the application, in order for the application to pick up the updated program load modules:

1. Locate the GeneralInsuranceCustomerApplicationServicesDev bundle manifest file to start the editor. Using the CICS Cloud perspective, in the Project Explorer view, expand the general.insurance.customer.applicationservices.to.dev.bundle project. Expand the META-INF subdirectory, then double-click the cics.xml manifest file to start the CICS Bundle Manifest Editor.

The CICS Bundle Manifest Editor is shown in Figure 6-29.

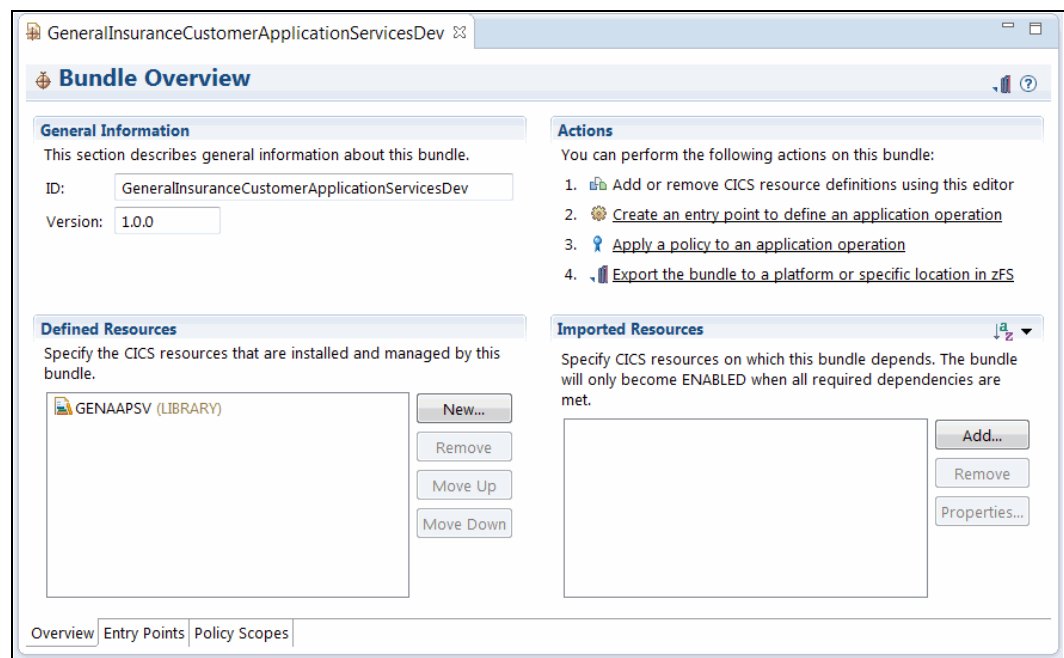


Figure 6-29 CICS Bundle Manifest Editor for the GeneralInsuranceCustomerApplicationServicesDev bundle

2. On the Overview tab in the Defined Resources section, double-click the **GENAAPSV (LIBRARY)** resource to open the editor.

The Attributes editor appears, as shown in Figure 6-30.

Name	CICS Name	Value
Basic		
Critical	CRITICAL	
Data Set Name 01	DSNAME01	CICSSEM.CB12.APPLSERV.#1.#0.#2
Data Set Name 02	DSNAME02	
Data Set Name 03	DSNAME03	
Data Set Name 04	DSNAME04	
Data Set Name 05	DSNAME05	
Data Set Name 06	DSNAME06	
Data Set Name 07	DSNAME07	
Data Set Name 08	DSNAME08	
Data Set Name 09	DSNAME09	
Data Set Name 10	DSNAME10	
Data Set Name 11	DSNAME11	
Data Set Name 12	DSNAME12	
Data Set Name 13	DSNAME13	
Data Set Name 14	DSNAME14	
Data Set Name 15	DSNAME15	
Data Set Name 16	DSNAME16	
Description	DESCRIPTION	Development load libraries for ApplicationServices
Ranking	RANKING	50
Status	STATUS	
Userdata 1	USERDATA1	
Userdata 2	USERDATA2	
Userdata 3	USERDATA3	

Figure 6-30 Attributes editor for LIBRARY resource GENAAPSV

The LIBRARY definition contains an ordered list of 16 possible data set names, called data set Name 01 through data set Name 16. Currently, a value appears for the Name 01 data set only. You add the data set for the updated programs ahead of the others.

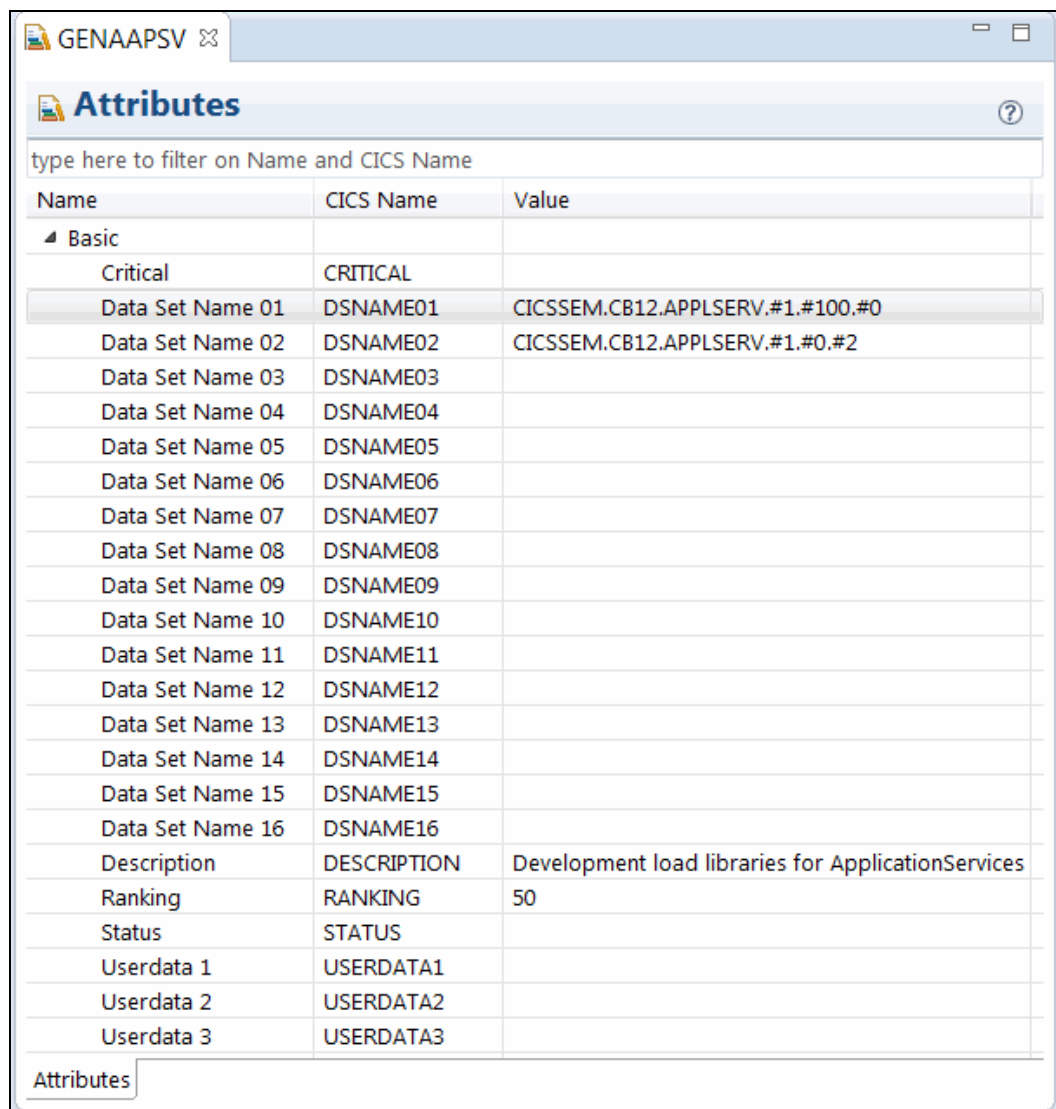
3. Copy the value `<USERHLQ>.CB12.APPLSERV.#1.#0.#2` from data set name 01 to data set name 02.

Tip: In the Attributes editor, double-click a Value field to edit the content.

4. Enter the value `<USERHLQ>.CB12.APPLSERV.#1.#100.#0` into data set name 01.

Hint: In the Attributes editor, after you have edited a field, click elsewhere on the view to signify the end of the field editing.

5. Press **Ctrl+S** to save changes. The completed updates to the LIBRARY attributes are shown in Figure 6-31.



The screenshot shows a window titled "GENAAPSV" with a tab labeled "Attributes". Below the tab is a search bar with the placeholder text "type here to filter on Name and CICS Name". The main area contains a table with three columns: "Name", "CICS Name", and "Value". The table lists various attributes, including "Basic", "Critical", "Data Set Name 01" through "Data Set Name 16", "Description", "Ranking", "Status", and "Userdata 1" through "Userdata 3". The "Data Set Name 01" row is highlighted.

Name	CICS Name	Value
Basic		
Critical	CRITICAL	
Data Set Name 01	DSNAME01	CICSSEM.CB12.APPLSERV.#1.#100.#0
Data Set Name 02	DSNAME02	CICSSEM.CB12.APPLSERV.#1.#0.#2
Data Set Name 03	DSNAME03	
Data Set Name 04	DSNAME04	
Data Set Name 05	DSNAME05	
Data Set Name 06	DSNAME06	
Data Set Name 07	DSNAME07	
Data Set Name 08	DSNAME08	
Data Set Name 09	DSNAME09	
Data Set Name 10	DSNAME10	
Data Set Name 11	DSNAME11	
Data Set Name 12	DSNAME12	
Data Set Name 13	DSNAME13	
Data Set Name 14	DSNAME14	
Data Set Name 15	DSNAME15	
Data Set Name 16	DSNAME16	
Description	DESCRIPTION	Development load libraries for ApplicationServices
Ranking	RANKING	50
Status	STATUS	
Userdata 1	USERDATA1	
Userdata 2	USERDATA2	
Userdata 3	USERDATA3	

Figure 6-31 Updated attributes for LIBRARY resource GENAAPSV

The GeneralInsuranceCustomer application defines the LGICUS01 program. The application also defines a LIBRARY resource called GENAAPSV, which provides the location for the program load modules. During execution, the dynamic LIBRARY resources installed by the application are searched. Even though the LGICUS01 load module appears more than once in the data sets included in the application, you have ensured that the updated version appears ahead of the older versions in the GENAAPSV LIBRARY resource.

You have now updated the GeneralInsuranceCustomerApplicationServicesDev CICS bundle to update the LIBRARY resource. As a result of modifying the CICS bundle, you now update the bundle version to reflect the minor version increase by performing the following steps:

1. If you have closed the window, open the GeneralInsuranceCustomerApplicationServicesDev CICS Bundle Manifest Editor.
2. On the Overview tab in the General Information section, update the version from 1.0.0 to 1.100.0.

3. Press **Ctrl+S** to save changes. Figure 6-32 shows the updated CICS Bundle Manifest for the GeneralInsuranceCustomerApplicationServicesDev bundle.

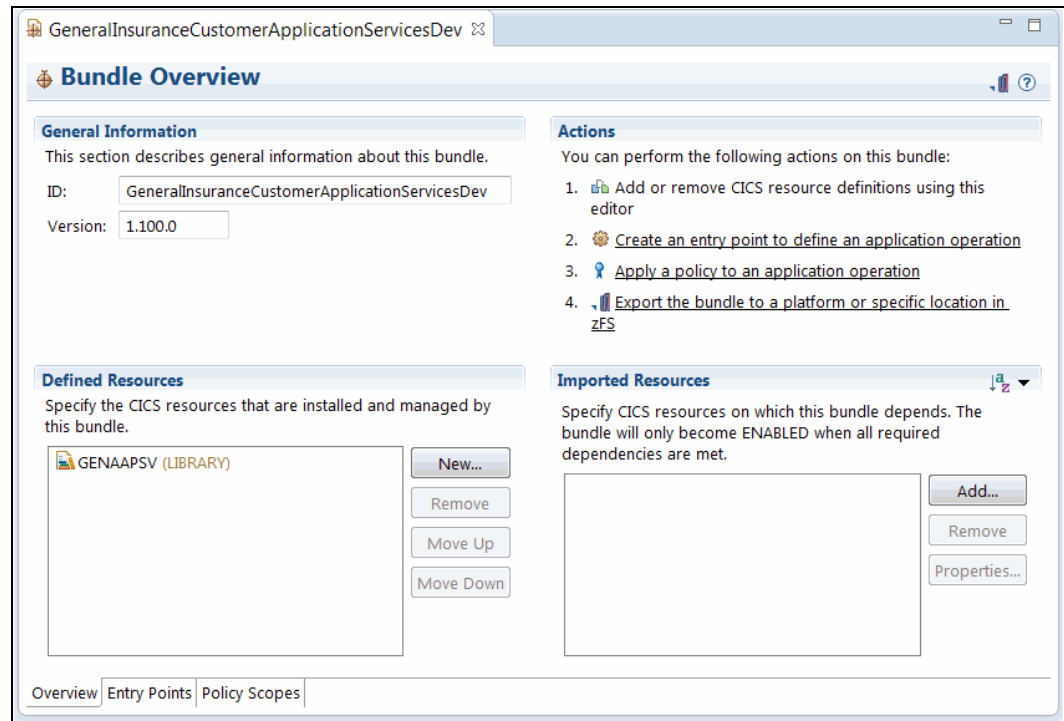


Figure 6-32 CICS Bundle Manifest Editor for version 1.100.0 of bundle GeneralInsuranceCustomerApplicationServicesDev

The CICS bundle version has been updated to track changes made to the bundle. You might notice that errors are reported after the version change to the CICS bundle. This is because the application and application binding projects are expecting a different version of the CICS bundle. The next steps will rectify these errors.

6.3.3 Updating the application version

In this example, you are providing a minor version update to the GeneralInsuranceCustomer application. Perform the following steps to increment the versions of the CICS TS application and application binding project, and to update projects to reference the updated CICS bundle GeneralInsuranceCustomerApplicationServicesDev:

1. In the Project Explorer view, expand the `general.insurance.customer.application` application project, expand the `META-INF` subdirectory, and double-click `bundles.xml` to open the Application Editor.
2. In the Application Editor, in the General Information section, change the value of the Version from 1.0.2 to 1.100.0.

3. Press **Ctrl+S** to save changes. Figure 6-33 shows the updated Application Editor for the GeneralInsuranceCustomer application.

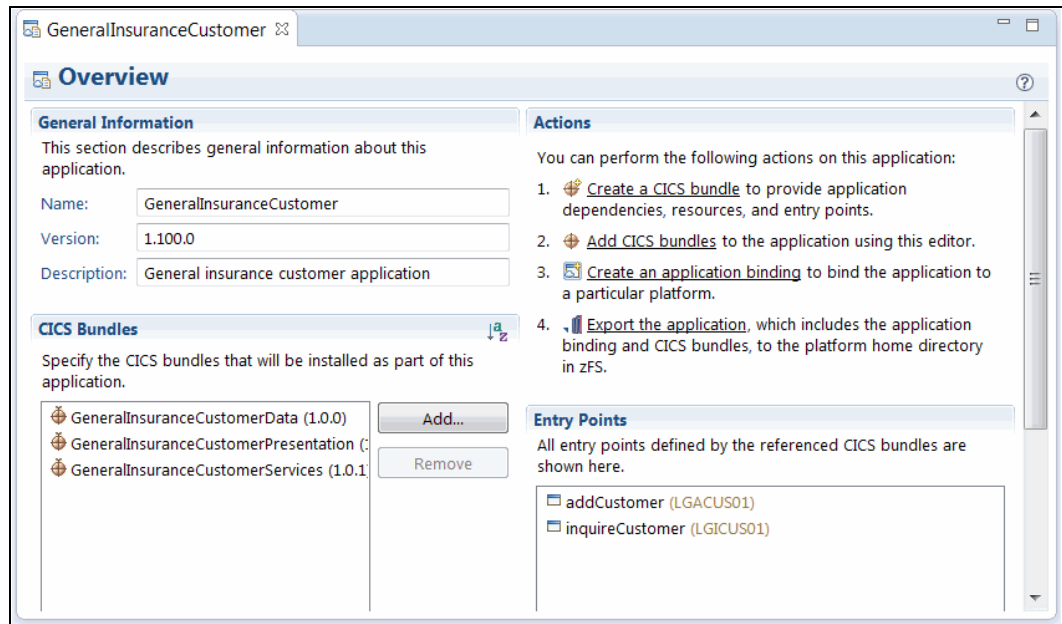


Figure 6-33 Application Editor for version 1.100.0 of the GeneralInsuranceCustomer application

The workspace still displays errors in the GeneralInsuranceCustomerToDev application binding due to references of the CICS bundle and application prior versions. Next, fix the workspace errors.

4. In the Project Explorer view, expand the general.insurance.customer.to.dev.platform.binding application binding project, expand the META-INF subdirectory, and double-click appbinding.xml to open the Application Binding Editor.

You see that the application binding still refers to the old version of the application, as shown in Figure 6-34.

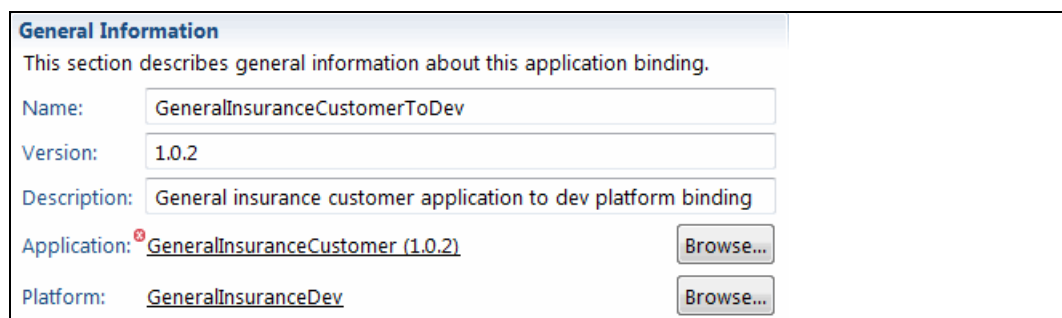


Figure 6-34 General Information section of the Application Binding Editor displaying the old version of the application

5. In the General Information section, click **Browse** next to Application: GeneralInsuranceCustomer (1.0.2). The Application Selection dialog displays.
6. In the Application Selection dialog, select **GeneralInsuranceCustomer (1.100.0)**, and click **OK**.
7. Press **Ctrl+S** to save changes.

Now, fix the reference to the updated version of the CICS bundle.

8. On the Application Binding Editor in the CICS bundles section, click **GeneralInsuranceCustomerApplicationServicesDev (1.0.0)** and click **Remove**. The prior version of the bundle is disassociated from the CICS TS application.
9. On the Application Binding Editor in the CICS bundles section, click **Add**. The CICS Bundle Selection dialog appears.
10. In the CICS Bundle Selection dialog, select **GeneralInsuranceCustomerApplicationServicesDev (1.100.0)**, as shown in Figure 6-35.

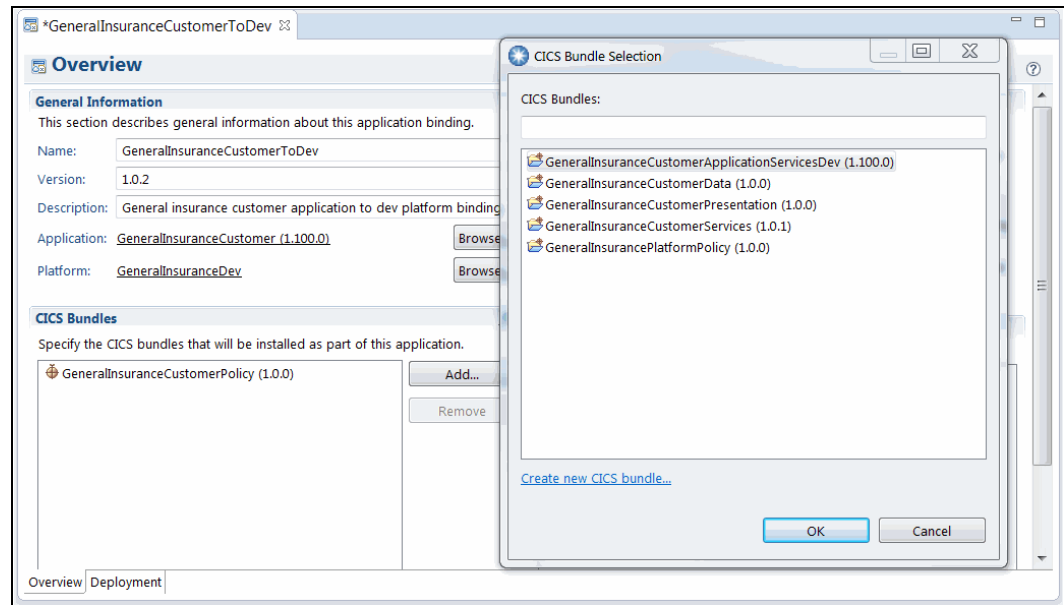


Figure 6-35 Adding the updated version of the bundle to the application binding

11. Click **OK**. The CICS Bundle Selection dialog closes and returns to the Application Binding Editor.

The `GeneralInsuranceCustomerApplicationServicesDev` bundle version change has been reflected in the CICS application binding application binding `GeneralInsuranceCustomerToDev`. Now, update the deployment rules for the CICS bundle.

12. In the Application Binding Editor, click the **Deployment** tab.
13. In the Region Types section, select the **ApplicationServices** region type.
14. In the list of CICS bundles, clear any check box next to `GeneralInsuranceCustomerApplicationServicesDev (1.0.0)`. That version of the bundle is not referenced in the application or the binding, and is removed from the list.
15. In the list of CICS bundles, select the box next to `GeneralInsuranceCustomerApplicationServicesDev (1.100.0)`. The updated version of the bundle is now associated with the `ApplicationServices` region type.

Figure 6-36 shows the updated deployment rules for the `ApplicationServices` region type.

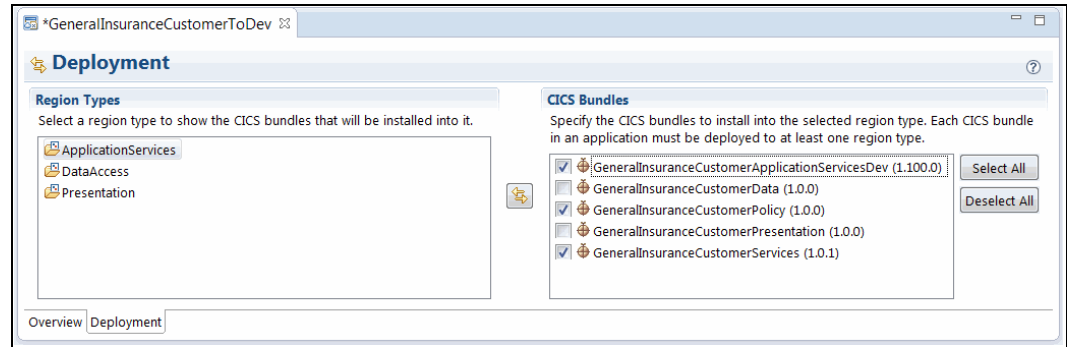


Figure 6-36 Application binding version 1.100.0 deployment rules for the `ApplicationServices` region type

All references to the correct versions of the bundle and application have been made in the application binding. To track the changes made in the application binding, increment the application binding version.

16. In the Application Binding Editor, click **Overview** tab.
17. In the GeneralInsuranceCustomerToDev Application Binding editor, in the General Information section, change the Version from 1.0.2 to 1.100.0.
18. Press **Ctrl+S** to save changes. The updated Application Binding editor is shown in Figure 6-37.

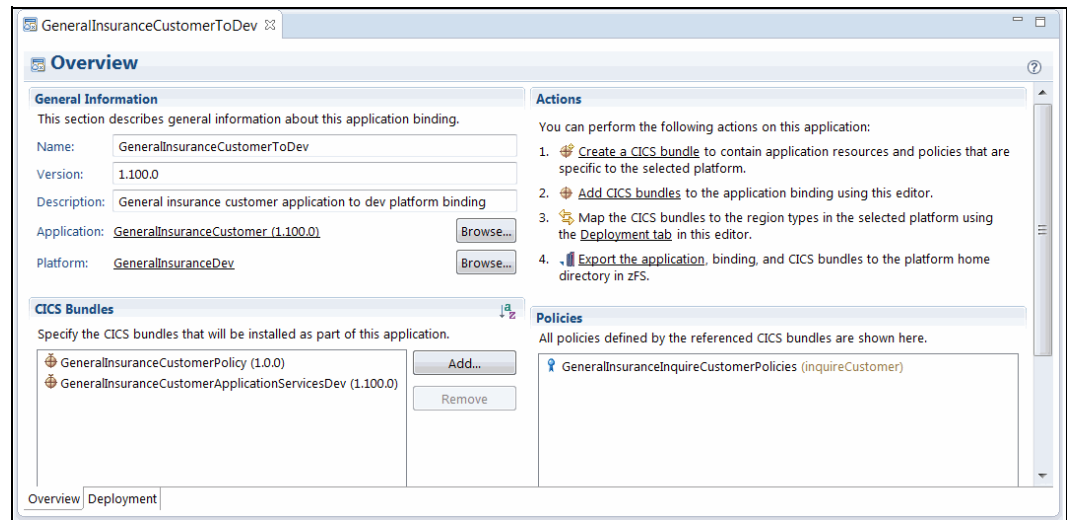


Figure 6-37 Application binding GeneralInsuranceCustomerToDev version 1.100.0

You have updated the versions of the updated CICS bundle, application, and application binding bundles.

6.3.4 Exporting, installing, and enabling

In this example, you have updated your application logic, updating the program source code, compiling it into a new load library, and then updating the CICS TS application to load programs from the new data set location. In the next steps, you will deploy this application.

The following steps describe the process to export, install, and enable the application. The scenario takes advantage of the application multiversion support in CICS TS. In particular, the ability to define program and LIBRARY resources that are private to a particular version of an application. Versions 1.0.2 and 1.100.0 of GeneralInsuranceCustomer both define program resources, LIBRARY resources, application entry points and dependencies with the same names.

In this example, the two versions of the application are deployed simultaneously. The GeneralInsuranceCustomer application version 1.0.2 is already enabled and available in the CICS TS development platform.

Perform the following steps to export the updated version of the CICS TS application, GeneralInsuranceCustomer version 1.100.0:

1. Using the CICS Explorer CICS Cloud perspective, in the Project Explorer view, right-click **general.insurance.customer.application** and select **Export Application Project to z/OS UNIX File System**. The Export Application to the home directory of a Platform wizard displays.
2. In the wizard, ensure that the correct CMCI connection is specified.
3. Select **GeneralInsuranceCustomerToDev (1.100.0)**, as shown in Figure 6-38.
4. Select the **Create Application Definition after export finishes** check box.

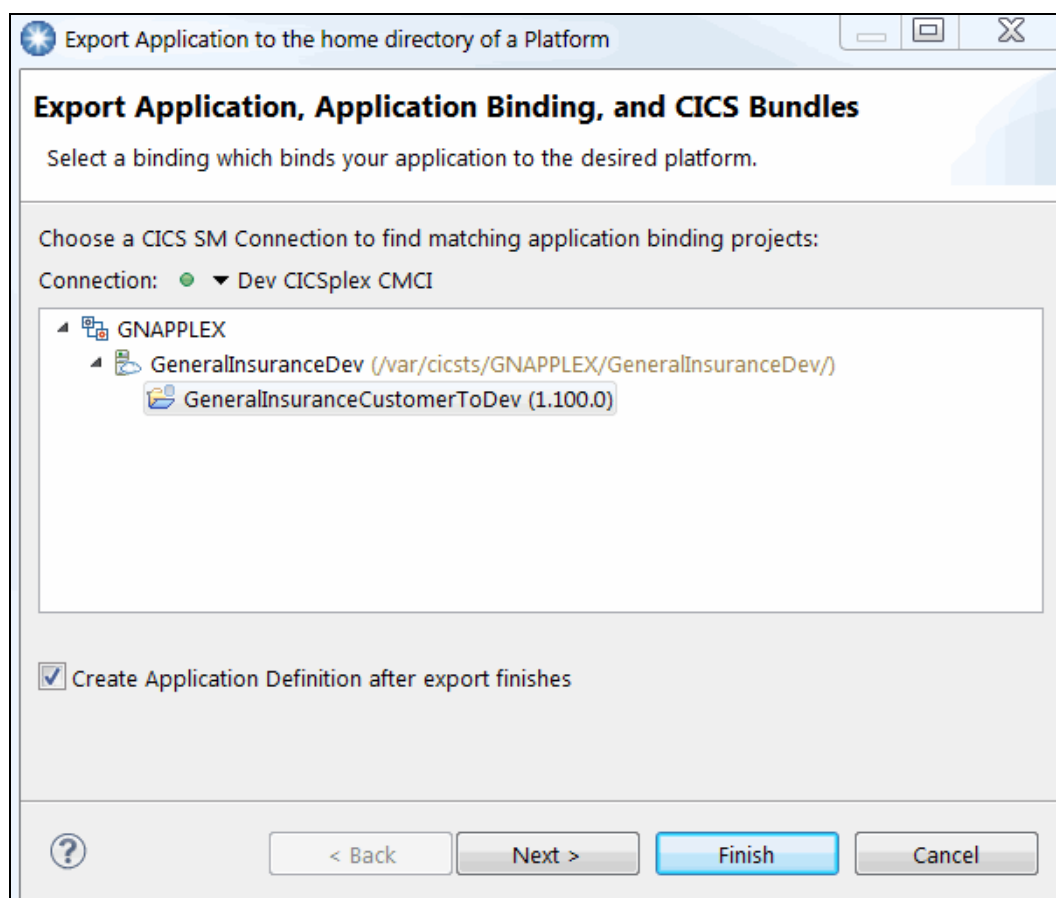


Figure 6-38 Selecting the correct application binding for exporting version 1.100.0 of GeneralInsuranceCustomer application

5. Click **Next**.

6. Ensure that the correct FTP connection is specified.
7. Click **Finish**.

The application, application binding, and required CICS bundles are transferred to the z/OS UNIX file system. The export process transfers files in the correct format, and puts them in the correct subfolders within the platform home directory.

When the export process is complete, the new Application Definition dialog appears.

8. Specify the following information in the new Application Definition dialog:
 - a. Name: GENACUST.
 - b. Description: General insurance customer application.
 - c. Keep the supplied values for the platform definition, version, Application directory, and binding directory. These values have been completed by the wizard using the values specified by the exported artifacts.
 - d. Clear the **Open editor** check box.

Figure 6-39 shows the completed New Application Definition dialog.

Figure 6-39 Creating the application definition for version 1.100.0 of GeneralInsuranceCustomer application

9. Click **Finish** in the New Application Definition dialog. Figure 6-40 shows the updated Cloud Explorer view after the creation of the application definition.

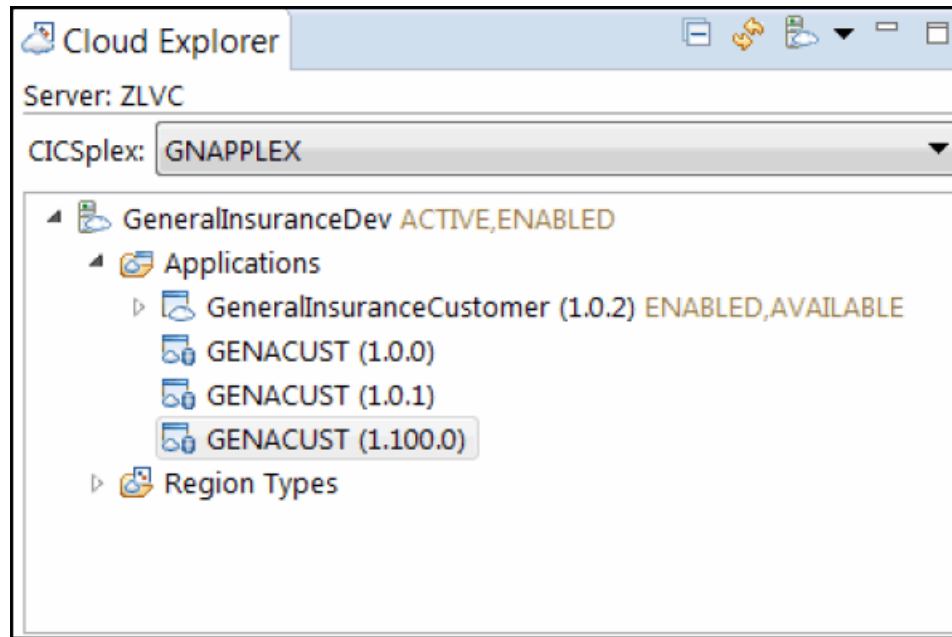


Figure 6-40 Cloud Explorer view after the creation of the application definition for version 1.100.0

The application-related projects have been exported to the z/OS file system, and an application definition has been created to instruct CICS TS on their location in the file system. Next, install the application definition and provision the application.

10. In the Cloud Explorer view, expand the GeneralInsuranceDev platform. Then, expand the Applications subfolder to view the installed applications and application definitions, as shown in Figure 6-40.
11. Right-click the **GENACUST (1.100.0)** application definition and select **Install**. The Perform Operation dialog appears.
12. Click **OK** to install the application.

The application is installed to the platform CICS TS regions. This includes the installation of the program definitions that are now included in the CICS TS application. The application installs into a disabled state. If the state displays a state other than disabled, wait a short amount of time and refresh the Cloud Explorer View.

The Cloud Explorer view refreshes, and the application definition is replaced with the installed application entry for version 1.100.0, as shown in Figure 6-41.

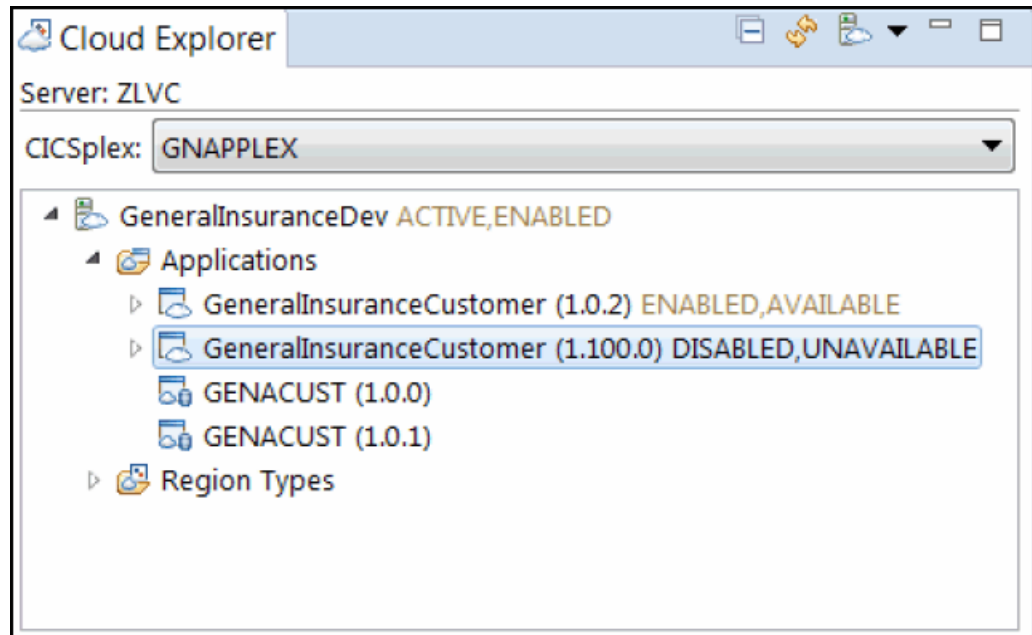


Figure 6-41 Cloud Explorer view showing the installed application, GeneralInsuranceCustomer version 1.100.0

13. Right-click the GeneralInsuranceCustomer (1.100.0) application and select **Enable**. The Perform Operation dialog displays. Click **OK** to enable the application.

You have exported the GeneralInsuranceCustomer V1.100.0 application, in addition to installing and enabling the application, as shown in Figure 6-42 on page 168. You now have two versions of the application (version 1.0.2 and version 1.100.0) concurrently enabled. However, there is an additional step required to make the application version 1.100.0 available to callers. Until this step is undertaken, users continue to use version 1.0.2, and are unaffected by the concurrent version enablement.

Figure 6-42 shows two enabled versions of the application.

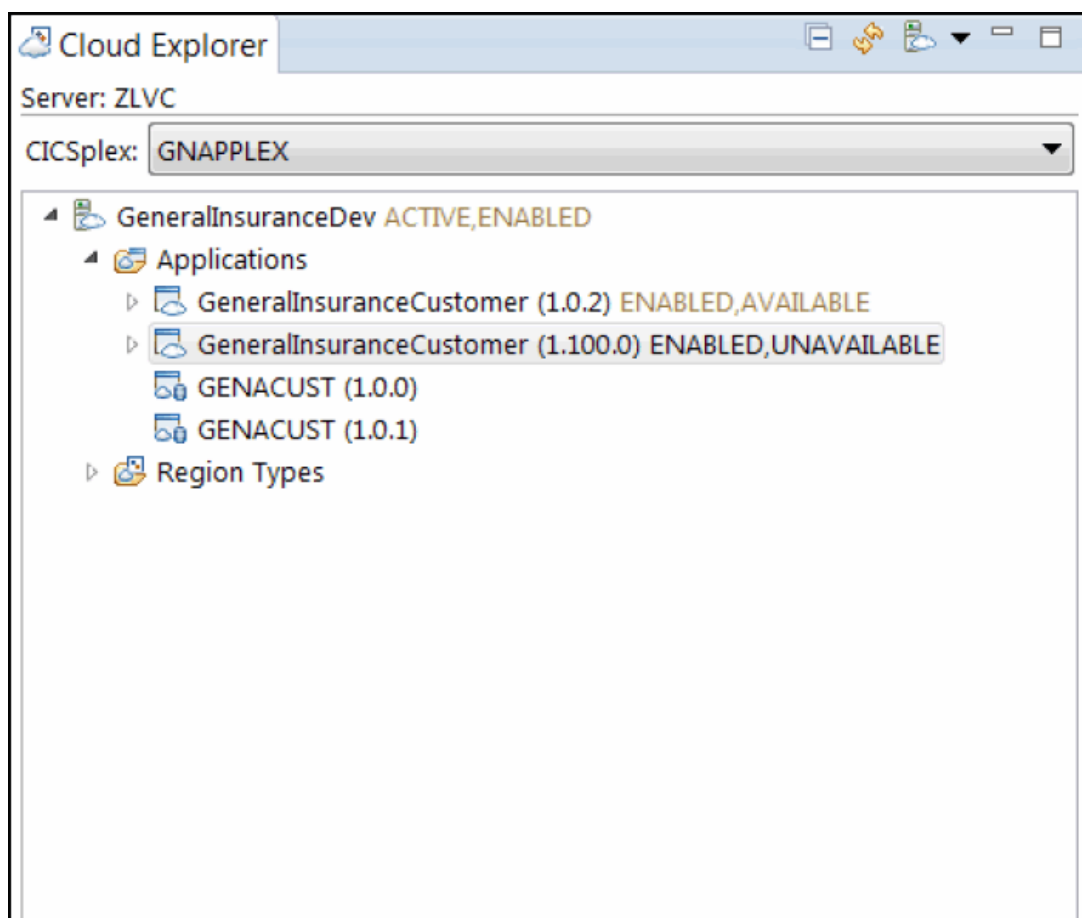


Figure 6-42 Cloud Explorer view showing two versions of the application enabled simultaneously

6.3.5 Validating that application version 1.0.2 is still operational

During the installation and enablement of an application, the CICS TS resources are created and enabled, but the application entry points are not made available. This state enables the application deployer to check the health of the deployment so far, without affecting application users. In previous CICS TS releases, the application was fully provisioned after the enablement.

However, had there been problems during the enablement, the entire application might need to be disabled and discarded while the problem was resolved, resulting in service outages. By having the additional step of making the application available, you have the opportunity to verify that all of the resources have successfully reached the enabled state.

Also, dependencies can be checked before the application is made available to callers. At this stage in the example, V1.0.2 is available but V1.100.0 is not available. Therefore, running GENAPP now should continue to behave as before.

Perform the following steps to run the GeneralInsuranceCustomer application to prove that the enablement of version 1.100.0 of the application has not affected the user:

1. At a CICS TS terminal session, enter the transaction SSC1. GENAPP displays.
2. Enter a valid customer number. For example, Cust Number: 0000000001.

3. Select Option 1 for a customer inquiry action and submit.

The panel updates with the customer details for the inquired customer, as shown in Figure 6-43. Note that the date of birth is still being returned to the user, and version 1.0.2 of the application is still being run.

The enablement of version 1.100.0 has not affected application version 1.0.2, which is currently in service.

```
SSC1      General Insurance Customer Menu

1. Cust Inquiry      Cust Number      0000000001
2. Cust Add          Cust Name :First  Andrew
                    :Last           Pandy
4. Cust Update       DOB             1950-07-11   (yyyy-mm-dd)
                    House Name
                    House Number      34
                    Postcode          PI10100
                    Phone: Home       01962 811234
                    Phone: Mob        07799 123456
                    Email Addr        A.Pandy@beebhouse.com

Select Option  1
```

Figure 6-43 Terminal execution of version 1.0.2 of the GeneralInsuranceCustomer application

6.3.6 Making application version 1.100.0 available and running it

In this second example, you have provided an updated version of the GeneralInsuranceCustomer application. In version 1.100.0, the application has been updated to mask the inquiry of customer date of birth details. The application has been installed and enabled in the CICS TS platform but has not been made available to users.

Perform the following steps to quickly deploy version 1.100.0 of the CICS TS application:

1. In the Cloud Explorer view, expand the GeneralInsuranceDev platform, and expand the Applications subfolder to view the installed applications and application definitions.
2. Right-click the **GeneralInsuranceCustomer (1.100.0)** application and select **Make Available**. The Perform Operation dialog displays. Click **OK** to make the application available through the application entry points.

The updated version of the GeneralInsuranceCustomer CICS TS application has now been deployed and is available for use by users. Figure 6-44 on page 170 shows the updated Cloud Explorer view displaying the multiple versions of the application currently available. With multiple versions of an application concurrently available in a platform, users can target which version of the application to run by using the **EXEC CICS start APPLICATION** application programming interface (API).

In situations where this API is not used, such as with existing usage of **EXEC CICS LINK** or the target program of a transaction resource, the resource from the highest available application version is started. In this example, the **EXEC CICS LINK PROGRAM(LGICUS01)** command targets the LGICUS01 program deployed by the highest available version of the application, which in this case is version 1.100.0.

Figure 6-44 shows two available versions of the application.

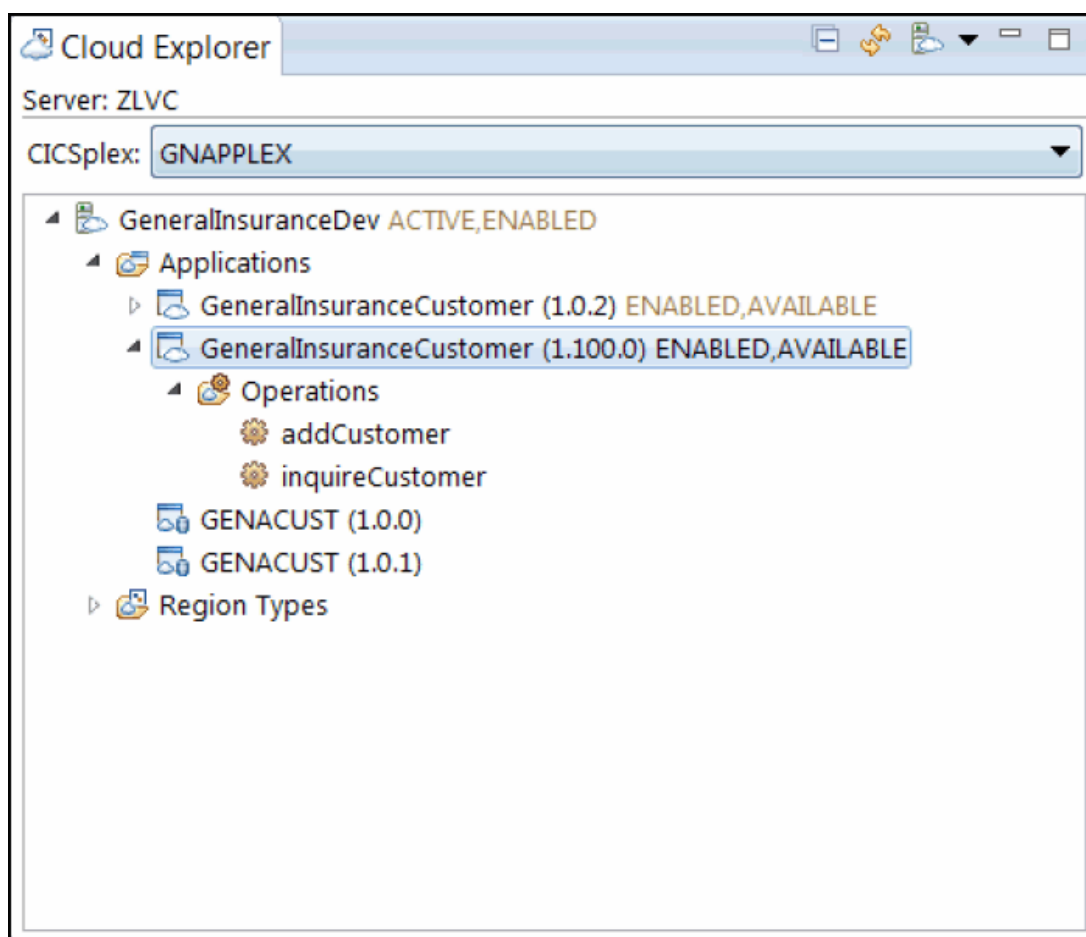


Figure 6-44 Cloud Explorer view displaying two available versions, 1.0.2 and 1.100.0, of the GeneralInsuranceCustomer application

You have deployed version 1.100.0 of the GeneralInsuranceCustomer application. Perform the following steps to test the deployment:

1. At a CICS TS terminal session, enter the transaction SSC1. GENAPP displays.
2. Enter a valid customer number. For example, Cust Number: 0000000001.
3. Select Option 1 for a customer inquiry action and submit.

The window updates with the customer details for the inquired customer, as shown in Figure 6-45. Note that the date of birth field has now been masked after the program updates.

SSC1 General Insurance Customer Menu		
1. Cust Inquiry	Cust Number	0000000001
2. Cust Add	Cust Name :First	Andrew
	:Last	Pandy
4. Cust Update	DOB	****-**-** (yyyy-mm-dd)
	House Name	
	House Number	34
	Postcode	PI10100
	Phone: Home	01962 811234
	Phone: Mob	07799 123456
	Email Addr	A.Pandy@beebhouse.com
Select Option <u>1</u>		

Figure 6-45 Terminal execution of version 1.100.0 of the GeneralInsuranceCustomer application

The application has been successfully deployed. You have provided an update to an application with no service outage to users.

6.4 Possible extensions for the application examples

You have deployed multiple versions of the GeneralInsuranceCustomer CICS TS application. You can try to experiment with the deployed applications using the CICS Cloud perspective in CICS Explorer.

You can observe the ease with which you can roll back to an earlier version of an application. Currently, you have version 1.0.2 and 1.100.0 of the application available. Running the application, you see customer details being returned by version 1.100.0, as shown in Figure 6-45. Make the application version 1.100.0 unavailable. After this single action, the user then sees the customer details being returned by version 1.0.2, as shown in Figure 6-43 on page 169.

With both application versions 1.0.2 and 1.100.0 available, you can use the **EXEC CICS start APPLICATION** API to choose which version of the application to run. Using this API, you can programmatically route work to a particular version of the application. In the GENAPP application, you can replace the **EXEC CICS LINK PROGRAM(LGICUS01)** instructions in the LGTESTC1 program for **EXEC CICS start APPLICATION** statements.

The **start APPLICATION** statement takes attributes of the application name and operation to start, and the intended version of the application. As an example, you can try to write conditional logic in LGTESTC1 (such as testing the User ID), and then choose which subset of users can start version 1.0.2 or 1.100.0.

6.5 Summary

In this chapter, you have been shown steps to transition from the initial adoption of cloud enabling your CICS TS applications. The GeneralInsuranceCustomer application began forming dependencies on the key artifacts of GENAPP, through to defining the resources related to the AOR (ApplicationServices) layer.

With the resources now defined within the CICS TS application and CICS Bundle Projects, you are able to control the lifecycle of the application as an entirety. Using the application multiversioning support added in CICS TS V5.2, you have been able to deploy multiple versions of an application concurrently, provide updates with no service outage, and easily roll back application versions.

You have been shown the different roles that the CICS TS application and CICS application binding can play in cloud enabling your CICS TS application. By correctly using the CICS application binding, you can move your CICS TS application through development, test, and production without the need to modify the core of the application. This enables you to more easily implement DevOps.

You give the application developer the possibility to package code as an application that can be versioned, declare dependencies, and manage the application as an entity through the lifecycle. Alternatively, you enable the operator to give an abstraction of the underlying topology and complexity using a platform. The bindings fill the gap and provide everything specific for that one application on that one platform.



Part 3

What you get by cloud-enabling your CICS TS

In this part of the book, we delve further into the business value benefits of cloud enabling your IBM Customer Information Control System (CICS) Transaction Server (CICS TS) environment.



Measurement by application

The creation of the application context presents an opportunity to review the way performance monitoring and charge back mechanisms are performed. This chapter explains the improvements in the process, and demonstrates how to use them:

- ▶ 7.1, “Overview” on page 176
- ▶ 7.2, “Examples of using the application context in the CICS Monitoring Facility” on page 178

7.1 Overview

When an IBM Customer Information Control System (CICS) Transaction Server (CICS TS) task enters an application through one of the application's entry points, application context is associated with the task, as described in 4.2, "The components of a CICS TS application" on page 53. This task application context information is available in the performance records written to System Management Facilities (SMF) as part of the CICS Monitoring Facility (CMF).

Customers can use the application context to assist with charge back accountancy, performance monitoring, capacity planning, and problem determination. The operation field of the application context can provide a different perspective about which parts of the customer application is running, and which resources are being used.

With the CMF enabled in a CICS TS region, a performance record can be written to record how many resources a single task has consumed. A small subset of resource usage data that is recorded in a performance record is included in the following list:

- ▶ Processor time
- ▶ Virtual storage usage
- ▶ File input/output (I/O) counts
- ▶ Database access

Armed with this wealth of statistical data, an information technology (IT) department can calculate how many processor seconds the payroll department has used in a month, and charge them to a percentage of the IT budget. The performance monitoring staff can use the data to ensure that CICS TS is running efficiently for the needs of the application, and the capacity planners can plan upgrades to the hardware to meet future needs.

Traditionally, one or more of the following fields are used to assign task resource usage to a department or customer:

- ▶ Transaction ID
- ▶ Initial program ID (for example, the first program run in a transaction)
- ▶ Terminal ID
- ▶ User ID
- ▶ Region APPLID

Depending on how the application has been structured, it can be difficult or impossible to determine which business function a transaction is performing, and to which department or customer it should be assigned. In a worst-case scenario, a transaction that cannot be assigned could be put into the IT operating expenses.

The application context within the CMF performance record enables an additional method of identifying the function or owner of a task.

7.1.1 The application context within the CICS Monitoring Facility

New data fields have been made available in the CMF performance record:

- ▶ Application name
- ▶ The major, minor, and micro version numbers of the application
- ▶ Platform name
- ▶ Operation name

The following sections describe possible uses of the new fields.

Application name

Rather than maintaining a list of payroll transactions for charge back, the reports could be summarized on the application name so that future additions to the transaction IDs used by the application are included automatically.

Major, minor, and micro version numbers

These version numbers could be used to compare the resource consumption of an application before and after enhancements have been applied.

Platform name

The platform name can be used to identify where part of an application ran, rather than using the CICS TS applid (TOR1, TOR2, AOR3), and therefore avoid the need to maintain applid lists.

Operation name

This field could be used to identify the function of a transaction where the traditional summary fields do not give the required granularity.

7.1.2 Problems with traditional chargeback and performance monitoring

The application context can be used to resolve problems using the traditional summary fields that were listed earlier. The following sections describe these problems in more detail.

Transaction ID and initial program

Some business applications, third-party or internal, use the same transaction ID and initial program for all of the distinct business functions. This is especially true when a menu is presented to the user to select an inquiry, update, or delete function. After the user has selected the option, the initial program will perform an EXEC CICS LINK to a second program depending on the function.

However, in the performance record for the task, by default only the transaction ID and initial program is recorded. The CMF can be customized so that it can write additional records on each link, but this dramatically increases the number of performance records that are written.

If separate entry points are created for the inquiry, update, and delete programs, the operation name in the performance record can be used to summarize and group all of the tasks based on function.

Multi-region accounting

When a user's transaction runs across multiple regions, it can be difficult to correlate the task performance records from the different regions to obtain a true picture of the transaction's resource consumption. If an entry point is created within the initial region, the resulting application context will flow with the task to the subsequent regions.

The performance record from the task in the subsequent regions contains the application name, version, platform, and operation fields that were set in the initial region. This makes it much easier to assign resource consumption to a business function. This only works if the communications link is multiregion operation (MRO) or Internet Protocol interconnectivity (IPIC).

7.1.3 Pre-requisites to using application context in CICS Monitoring Facility records

To make use of the application context in the CMF performance records, the following elements are required:

- ▶ System initialization table (SIT) parameters **MN=ON** and **MNPERF=ON** (or enable performance monitoring on your running CICS TS region, for example, **CEMT SET MON ON PERF**)
- ▶ One or more applications installed, enabled, and available on your CICS TS region
- ▶ A software product that can process CMF performance records, for example, CICS Performance Analyzer V5.1

7.2 Examples of using the application context in the CICS Monitoring Facility

This section provides you with several examples using the application context with CMF.

7.2.1 Enabling chargeback within the GENAPP application using CICS Performance Analyzer

In Example 7-1, the CICS Performance Analyzer V5.1 example report could be used to calculate the processor resource usage for a series of the business functions.

Example 7-1 CICS Performance Analyzer report

V5R1M0		CICS Performance Analyzer Performance Summary					
SUMM0001 Printed at 13:24:53 1/02/2013		Data from 12:45:26 1/02/2013 to 12:46:22 1/02/2013					
ACAppNm	ACOperNm	Total #Tasks	Total User CPU FC Time	Total TS Count	Total Count		
GeneralInsuranceCustomer	addCustomer	18	.0561	18	0		
GeneralInsuranceCustomer	inquireCustomer	39	.0773	0	0		
GeneralInsuranceCustomer		57	.1334	18	0		
Total		57	.1334	18	0		

Example 7-2 shows the CICS Performance Analyzer V5.1 parameters used to generate the example report.

Example 7-2 CICS PA report

```
CICSPA IN(SMFIN001),  
      LINECNT(60),  
      FORMAT(':', '/', ),  
      PRECISION(4),  
      SUMMARY(OUTPUT(SUMM0001),  
      EXTERNAL(CPAXW001),
```

```

SELECT (PERFORMANCE(
INC (ACAPPLNM (General*)))),
TOTALS (8),
INTERVAL (00:01:00),
FIELDS (ACAPPLNM (ASCEND),
ACOPERNM (ASCEND),
TASKCNT,
CPU (TIME (TOT)),
FCTOTAL (TOT),
TSTOTAL (TOT)))

```

7.2.2 Monitoring performance of the GENAPP application using CICS Performance Analyzer

In Example 7-3, the following CICS Performance Analyzer V5.1 example report can be used to detect an increase in resource usage when an application is upgraded.

Example 7-3 CICS PA report

V5R1M0			CICS Performance Analyzer Performance Summary						
SUMM0001 Printed at 13:51:51 1/02/2013			Data from 12:45:26 1/02/2013 to 12:46:22 1/02/2013						
ACApp1Nm	ACApp1Vr	ACOperNm	#Tasks	User	Avg CPU Time	Avg SC24UHWMT Count	Avg SC31UHWMT Count	Avg DB2 Reqs FC	Total Count
GeneralInsuranceCustomer	1.0.0	addCustomer	27		.0029	1465	372232	1	1
GeneralInsuranceCustomer	1.0.0	inquireCustomer	56		.0019	1522	271139	1	0
GeneralInsuranceCustomer	1.0.0		83		.0022	1503	304025	1	0
GeneralInsuranceCustomer	1.0.1	addCustomer	18		.0031	1632	394601	1	1
GeneralInsuranceCustomer	1.0.1	inquireCustomer	39		.0020	1634	282194	1	0
GeneralInsuranceCustomer	1.0.1		57		.0023	1633	317691	1	0
GeneralInsuranceCustomer			140		.0023	1556	309589	1	0
Total			140		.0023	1556	309589	1	0

Example 7-4 shows the CICS Performance Analyzer V5.1 parameters used to generate the example report.

Example 7-4 CICS PA report

```

CICSPA IN(SMFIN001),
LINECNT(60),
FORMAT(':', '/'),
PRECISION(4),
SUMMARY (OUTPUT (SUMM0001),
EXTERNAL (CPAXW001),
SELECT (PERFORMANCE(
INC (ACAPPLNM (General*)))),
TOTALS (8),
INTERVAL (00:01:00),

```

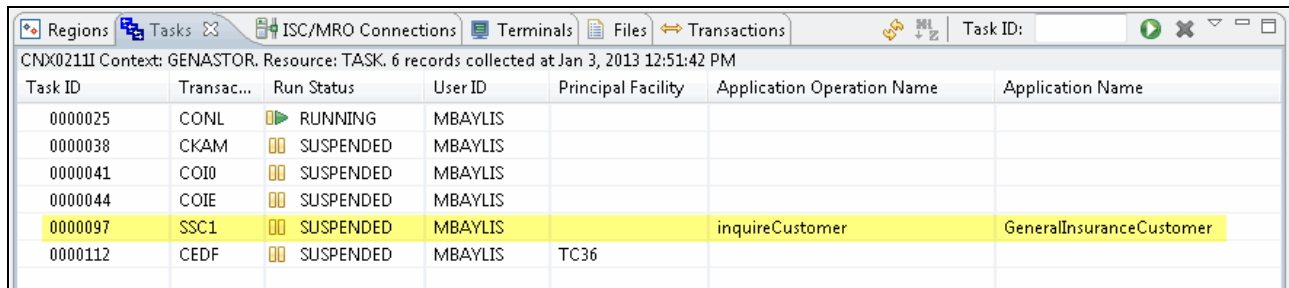
```

FIELDS(ACAPPLNM(ASCEND),
        ACAPPLVR(ASCEND),
        ACOPERNM(ASCEND),
        TASKCNT,
        CPU(TIME(AVE)),
        SC24UHW(AVE),
        SC31UHW(AVE),
        DB2REQCT(AVE),
        FCTOTAL(AVE)))

```

7.2.3 Diagnosing problems in GENAPP using CICS Explorer

Understanding the purpose of a task that is causing problems in a CICS TS region becomes easier when the task is assigned an application context. Figure 7-1 demonstrates that task 97 is the common GENAPP transaction SSC1, which is suspended. However, because the task has been assigned an application context, it is easy to determine that it is an inquire customer operation.



Task ID	Transac...	Run Status	User ID	Principal Facility	Application Operation Name	Application Name
0000025	CONL	▶ RUNNING	MBAYLIS			
0000038	CKAM	⏸ SUSPENDED	MBAYLIS			
0000041	COI0	⏸ SUSPENDED	MBAYLIS			
0000044	COIE	⏸ SUSPENDED	MBAYLIS			
0000097	SSC1	⏸ SUSPENDED	MBAYLIS		inquireCustomer	GeneralInsuranceCustomer
0000112	CEDF	⏸ SUSPENDED	MBAYLIS	TC36		

Figure 7-1 Suspended Task 97: GENAPP transaction



Managing by policy

Chapter 5, “Applying a policy” on page 97 introduced IBM Customer Information Control System (CICS) Threshold Policy, and showed the steps involved in defining and deploying policies. This chapter provides information about more general aspects of CICS Policy, and describes how policy can be used to manage IBM CICS TS applications and platforms:

- ▶ 8.1, “Benefits of CICS Policy” on page 182
- ▶ 8.2, “Reviewing CICS Threshold Policy” on page 182
- ▶ 8.3, “Policy scope” on page 186
- ▶ 8.4, “Setting policy” on page 190
- ▶ 8.5, “Policy scenarios” on page 191
- ▶ 8.6, “Concluding thoughts” on page 192

8.1 Benefits of CICS Policy

CICS Policy provides a way of controlling the behavior of CICS TS applications running in a CICS TS platform that is declarative, and that can be managed separately from the applications or platforms themselves. CICS Policy provides a step toward automating the behavior of a CICS TS environment. CICS Policy can be used to set an expectation for the resource usage of an application. CICS Policy can also be used to protect a CICS TS platform from unexpected resource consumption by the applications running on that platform.

Unexpected resource use might arise because the application has been written in such a way that it does not meet the guidelines for resource usage. Alternatively, the resource use might be because some error has occurred, causing a task to loop or behave in an undesirable way, or it could perhaps be a result of how a particular workload is driving the application.

The actions to be taken when a CICS Policy rule is contravened can be purely informational, or can lead to some kind of corrective action.

8.2 Reviewing CICS Threshold Policy

CICS Threshold Policy controls the behavior of applications by setting constraints on the resource usage by tasks running within an application on a platform. A *CICS Policy* contains one or more *policy rules*, where a policy rule defines a threshold condition, and the action to be taken if the threshold is exceeded. CICS TS supports several *policy rule types*, which determine the types of *threshold conditions* that can be defined. The tasks to which a policy rule applies depend on the *scope* into which that policy rule has been deployed.

The threshold limits apply to each of the tasks that fall within that scope, so that when any given task exceeds the resource usage specified by the policy rule, then the specified action is taken. In other words, the policy rules apply to individual tasks and not to an aggregated set of tasks.

This section provides some details about the policy rule types and threshold conditions that can be specified, and the policy actions that can be taken.

8.2.1 Policy rule types

The policy rule types that are supported by CICS TS are based on customer input about the types of resource usage that it would be most useful to be able control. Each of the rule types has a set of associated *rule items*, which enable the policy rule to be more granular. For example, the storage policy rule type has rule items ranging from 24-bit task storage to 64-bit shared storage. A rule item can be thought of as a qualifier to the overall rule type. Some of the rule types have a single rule item.

The policy rule types enable you to control the usage of the following areas:

- **Storage.** Storage can be managed based on the amount of storage used by a task (rule type of *Storage*), or the number of requests for storage made by a task (rule type of *Storage Request*). Within these rule types, there are rule items to enable you to specify the type of storage used or requested, whether it is task or shared storage, and whether it is 24-bit (below the line), 31-bit (above the line), or 64-bit (above the bar) storage.

This makes it possible to control, for example, the use of below-the-line storage by the tasks in an application. In some cases, the Storage rule type is most appropriate, because it manages the quantities of storage used by a task.

In other cases, it is a better fit to specify the number of requests to get storage that a task is allowed before the policy action should be taken. As an example, one of the uses of CICS Policy is to detect “rogue” tasks that are unexpectedly using too many resources, and a high number of storage requests might indicate that a task is looping.

- ▶ CPU or Elapsed time. The *Time* Rule type has Rule items for *CPU limit* and *Elapsed time*. The CPU limit rule item enables you to control tasks in an application or platform based on their usage of CPU. The Elapsed time rule item enables you to control tasks in an application or platform based on the amount of elapsed time for which they have been running.

CPU limit and Elapsed time are further examples of thresholds that might be used to detect rogue tasks. Elapsed time could also be used to help ensure that user response times do not become excessive.

- ▶ Data access requests. It is possible to control how many data access requests of various types are made by a task, either to a database using Structured Query Language (SQL) commands or to a file using CICS TS file control commands. A rule type of *Database request* enables you to specify the number of SQL commands made by a task.

The *File request* rule type provides control over the number of file accesses made by a task, and enables you to do so at the granularity of the types of request, whether these are read (*Read*), browse (*Start browse*, *Read next*, or *Read previous*), Read for update (*Read update*), *Write*, *Rewrite*, or *Delete* requests.

- ▶ Program links. The *Program request* rule type, with its rule item of *Link*, enables you to control how many programs a task links to. This rule type might be used to set an expectation about the complexity of applications.
- ▶ Start requests. The *Start request* rule type, with its rule item of *START command*, enables you to control how many other CICS TS transactions a task can asynchronously start. This rule type might be used to detect rogue tasks, such as a task that is looping.
- ▶ Sync point requests. The *Sync point request* rule type, with its rule item of *SYNCPOINT command*, enables you to control how many sync points, or sync point rollbacks, a task can take. This rule type might be used to detect rogue tasks.
- ▶ Transient data requests. The *TD Queue request* rule type has two rule items for *READQ TD commands* and *WRITEQ TD commands*. These rule items enable you to control how many requests are made to read or write transient data. This rule type might be used to detect rogue tasks, such as a task that is looping.
- ▶ Temporary storage requests. Temporary storage can be managed based on the amount of temporary storage used by a task (rule type of *TS Queue bytes*) or the number of requests for temporary storage made by a task (rule type of *TS Queue request*). Within the *TS Queue bytes* rule type, there are rule items to enable you to scope the policy to the particular type of temporary storage that it is, main or auxiliary.

Within the *TS Queue request* rule type, there are rule items to enable you to control how many read or write requests are made to temporary storage, and how many write requests are made to auxiliary or main storage. These rule types might be used to detect rogue tasks that are unexpectedly using too many resources, or to indicate that a task is looping.

The condition portion of a policy rule consists of a rule type (and optional rule item), an operator, a value, and the units of that value. For the CICS TS V5.2 threshold policies, the operator is always greater than, and the possible units that can be specified depend on the rule type. A policy rule also has a name, a description, and an action to be taken when the condition is met. Examples of policy rules are given in Chapter 5, “Applying a policy” on page 97.

8.2.2 Policy actions

The set of actions that can be specified when a policy threshold is exceeded provide an increasing amount of intervention into the behavior of the task. Therefore, this intervention implies an increasing degree of trust that is placed in the policy management's ability to control the platform and applications for you.

The least intrusive action is to send a message that provides details about the task that has exceeded the policy threshold, and the threshold value that was exceeded. This is purely informational. The next action in terms of the amount of intervention is to emit an event, which could be used to drive additional processing within CICS TS or externally to CICS TS, sent to an event monitor, or combined with other events using an event engine such as IBM WebSphere Operational Decision Manager.

The most intrusive action is to cause an abend of the task that exceeded the policy threshold. The message action informs an operator that the policy threshold has been exceeded, and enables the operator to choose what further action, if any, should be taken. The event action introduces the possibility of automating any further action that should be taken.

Finally, the abend action automates the removal of the offending task from the system. Only one action can be specified for each policy rule, but a combination of policy rules could be used to take increasingly stringent action as a sequence of increasing threshold values is exceeded.

Each of the actions has predefined information associated with it. For the *Issue message* action, message DFHMP3001 is issued. For the *Emit event* action, a predefined event is formatted and emitted using the specified Event Processing (EP) adapter or EP adapter set. For the *Abend task* action, the task is ended with abend code AMPB and message DFHMP3002 is issued. The following subsections provide more details about each action, and the predefined information associated with it.

Send message policy action

Message DFHMP3001 contains information about the following items:

- ▶ The task number (trannum) and transaction ID of the task that exceeded the policy threshold.
- ▶ The name of the CICS bundle containing the policy definition that was used to deploy the policy.
- ▶ The name of the policy definition.
- ▶ The name of the policy rule that was exceeded.
- ▶ Details of the policy rule that was exceeded, including the RuleType, the Category (rule item), and the Threshold with its value and units. The Threshold is the actual limit derived from the value and the units. For example, for a policy rule that specifies that the amount of 24-bit task storage used by a task should not exceed 20 kilobytes (KB), the RuleType in the message would be Storage, the Category would be task24, the Value would be 20, the Units would be K, and the Threshold would be 20480.
- ▶ The CurrentCount, which is the amount of the resource used by the task at the time that CICS TS detected the threshold had been exceeded. The CurrentCount will often be a larger number than the Threshold, because of how CICS TS validates the task against the applicable policy rules.

The following list provides examples of CurrentCount:

- For a storage policy rule, the storage usage is determined when storage is allocated, so if a task acquires an additional 10 KB of storage when it already had 15 KB, then the CurrentCount will show 25600 although the Threshold might be 20480.
- For a CPU time policy rule, the CPU time is determined each time the task is dispatched, so it might have exceeded the CPU threshold by some amount.

Figure 5-10 on page 108 shows an example of message DFHMP3001.

Emit event policy action

When the Emit event action is specified in a policy rule definition, the EP adapter or EP adapter set to be used to emit the event must also be specified. The EP adapter or EP adapter set does not need to have been defined or installed when the policy rule is created. However, if it has not been installed and enabled by the time the policy first fires (or if there is a problem with the EP adapters), the event emission fails, and one of message DFHMP3003, DFHMP3004, or DFHMP3005 is issued with an explanation of the failure.

The EP adapter defines the event format and transport that are used to emit the event to an event consumer. If an EP adapter set is specified, multiple events can be emitted to multiple event consumers as a result of exceeding a policy threshold.

The contents of the event (or events) that are emitted for CICS TS policies are predefined, with standard event context information, including an event binding name of CICS Policy Event, a business event name of CICS_Policy_Event, and a capture specification name of CICS_Policy_Event_Capture_Spec. The event data contains similar information about the task and the policy rule as that provided in message DFHMP3001, but also contains details about the platform, application, and application version, and operation.

Abend task policy action

When the abend action is specified, the event is ended with abend code AMPB, preceded by issuing message DFHMP3002. Message DFHMP3002 contains the abend code plus the same detailed information as message DFHMP3001. Figure 5-10 on page 108 shows an example of message DFHMP3002 and the messages associated with abnormally ending a task.

If more than one policy rule applies for a particular task and threshold value, the actions associated with the rules are processed so that any messages are issued first, then any events are emitted, and finally the task abend occurs if any rules specify an abend action.

8.3 Policy scope

The scope to which a policy applies is determined by how the policy is deployed. Figure 8-1 shows a simplified platform, which has a single region type, and onto which two applications have been deployed, each with two operations. This figure is used as a basis for illustrating the different scopes at which a policy can be deployed, and the user tasks to which the policy then applies.

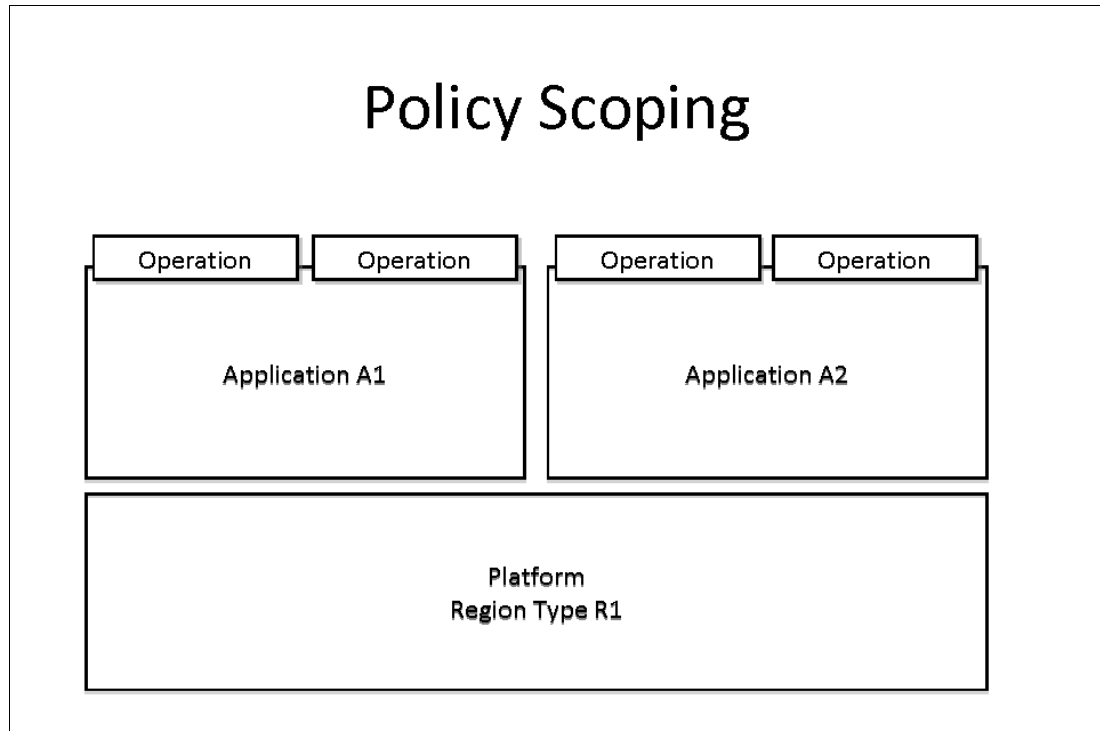


Figure 8-1 Simple platform schematic to illustrate policy scoping

Policy definitions are created in CICS bundles, and the CICS bundle is the unit of deployment for policies. The term *CICS Policy Bundle* is used to refer to a CICS bundle containing one or more policy definitions (which themselves contain one or more policy rules). Policies can be deployed in a cloud enabled CICS TS environment to the following scopes:

- ▶ To a platform scope by including the CICS bundle containing the policy as part of the CICS TS platform project, so that the policy is deployed when the platform is deployed.
- ▶ To a platform scope by adding the CICS bundle containing the policy to an already active platform. The Add bundle operation from the Cloud Explorer enables a CICS Policy Bundle to be added to one or more region types within the CICS TS platform.
- ▶ To an application scope by including the CICS bundle containing the policy as part of the CICS TS application project, so that the policy is deployed when the application is deployed.
- ▶ To an application scope by including the CICS bundle containing the policy in the application binding, so that the policy is bound to the application when it is deployed to a specific platform.
- ▶ To an operation scope by specifying a policy scope within the bundle manifest of the CICS Policy Bundle. This relates the policy to a specific application operation, which has been defined by specifying an application entry point for this application. A policy scope names the policy resource and the operation.

Figure 8-2 illustrates *platform scope* for a policy. In this example, the policy has been deployed into region type R1 in the platform, either by deploying the policy as part of the platform, or by adding the policy bundle to the region type in the platform. The policy applies to user tasks that run as part of any application that is deployed to the platform region type, and to any operation in those applications.

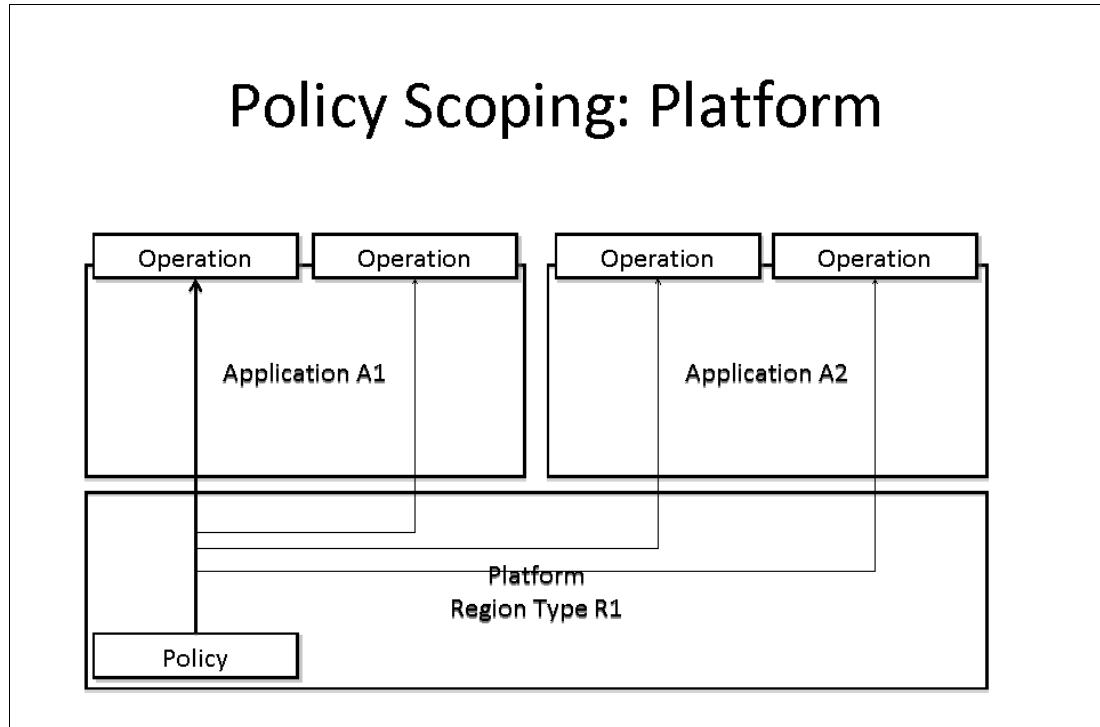


Figure 8-2 Simple example of platform policy scope

Figure 8-3 illustrates *application scope* for a policy. In this example, the policy has been deployed into Application A1 by including the policy bundle in Application A1 when it was deployed. The policy applies to user tasks running as part of Application A1, and to both operations of Application A1, but it does *not* apply to user tasks running as part of Application A2.

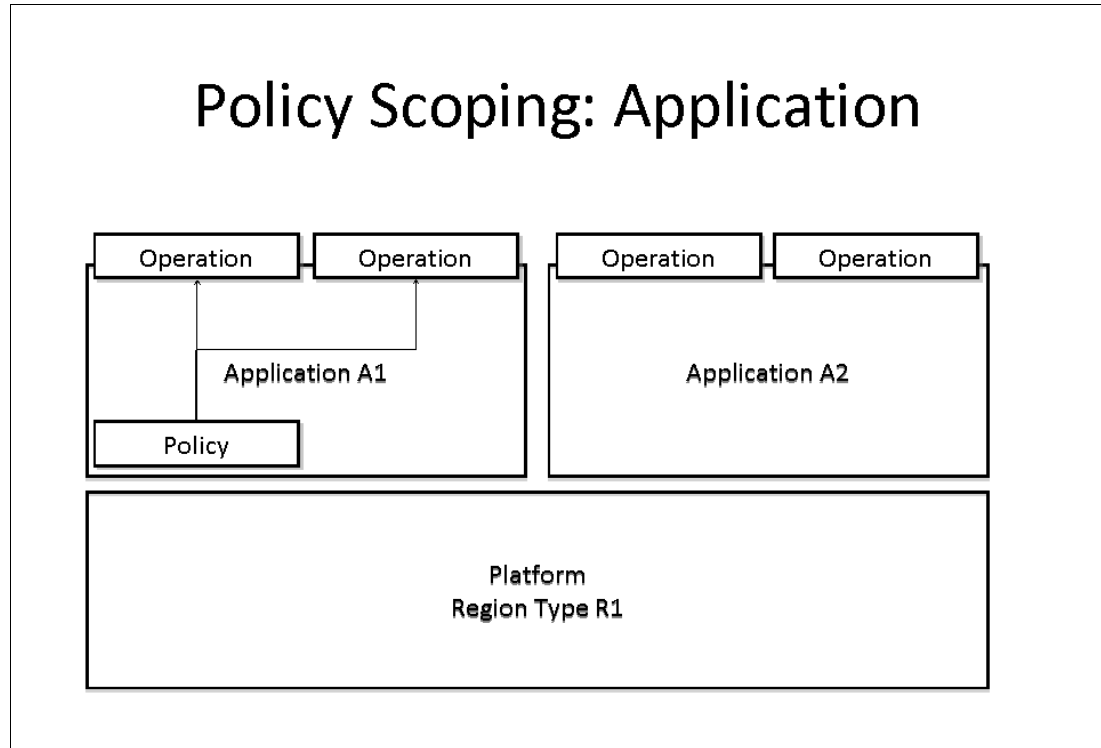


Figure 8-3 Simple example of application policy scope

The policy can also be deployed to the application scope by including it in an application binding used to deploy Application A1 to the Platform Region Type R1. The next section describes when it might be appropriate to include the policy in the application itself, and when it makes more sense to include it as part of the deployment of the application.

Finally, Figure 8-4 illustrates *operation scope* for a policy. In this case, a policy scope has been included in the CICS bundle manifest that associates the policy specifically with one of the operations of Application A2. The policy therefore applies to user tasks running for this particular operation of Application A2, but not to the other operation of Application A2, or to any user tasks running under Application A1.

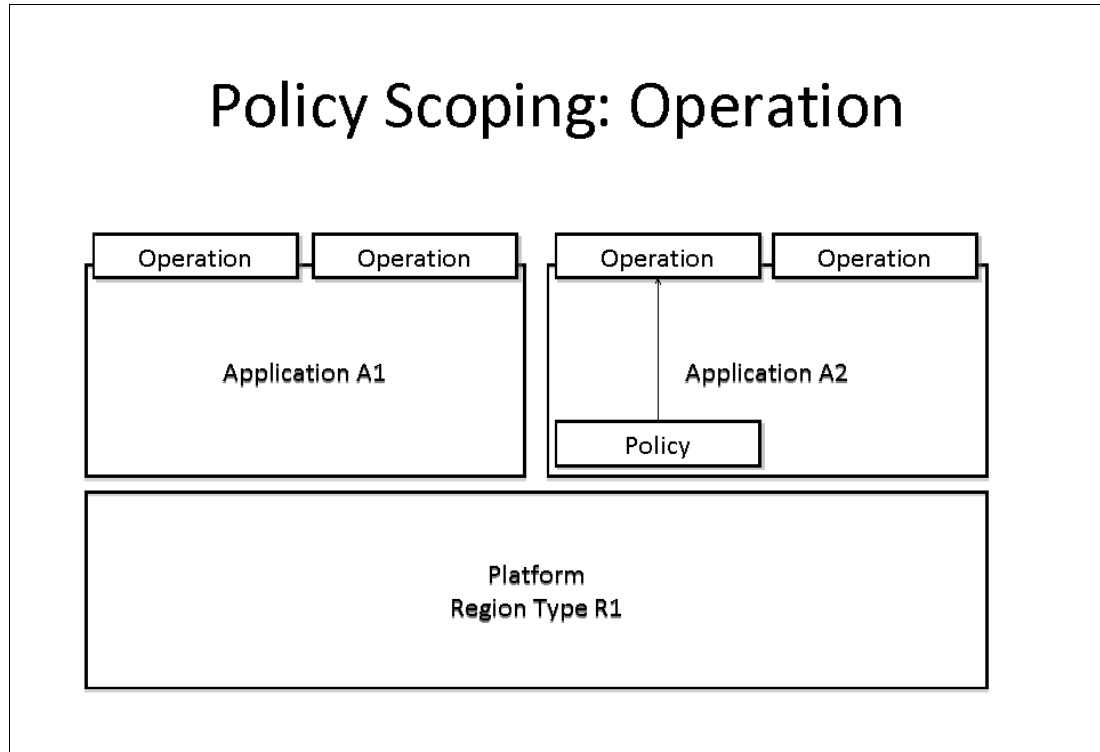


Figure 8-4 Simple example of operation policy scope

When any given task is running in the platform, it might be possible that it is subject to several policy rules:

- ▶ Rules that apply to this task because it is running in a platform region type to which a policy has been deployed
- ▶ Rules that apply to this task because it is running as part of an application to which those rules apply
- ▶ Rules that apply specifically to the operation that the task runs under

Where multiple rules apply, those which specify a lower threshold are checked first.

When a task is first attached, the policy rules that apply to it in its environment are established. Those rules are applied to that task, even if at some point the flow of control passes to a different application scope.

8.3.1 Policy for applications

Policy can be deployed at an operation or application scope to control how an application behaves, or to set the expectation for its resource usage.

The application developer can make a statement about the expected limits within which the application should operate, by adding policies into the application. This means that the policies are an integral part of the application, and apply regardless of the environment into which it is deployed. Some policies might relate to any task running within the application, but others would be specific to particular operations.

An *inquiry* operation might have a policy limiting its use of below-the-line storage, where an *update* operation might be known to require a certain amount of storage below-the-line. It is quite simple to add entry points to an application, and then use those to specify policies at an operation scope.

There might also be some policies that vary depending on the environment into which the application is deployed. These policies should not be added as part of the application, but should be included in the application binding, to enable them to be varied, perhaps to introduce more stringent policies for QA testing.

8.3.2 Policy for platforms

Policy can be deployed to a platform scope to determine the behavior expected or required of applications deployed to the platform, or to “protect” the platform from applications that might experience difficulties.

For example, the platform might deploy a set of policy rules that place upper limits on the amount of storage, the number of program calls, and the amount of CPU used by applications deployed to the platform, to protect the platform from any tasks that might be looping.

Alternatively, in a platform that has distinct regions for accessing data, a policy might be deployed to the AOR region type that limits the number of data access requests that user tasks can make. This ensures that applications deployed to this platform have been suitably structured to isolate the pieces that access data.

8.4 Setting policy

There are various considerations to be taken into account when deciding whether to apply policy rules, what policy rules to apply, and what the threshold values should be.

For platform scope rules, one consideration is whether this is a development, test, or production platform, and another consideration is the nature of the applications that are to run on this platform. For a development platform, there might not be any policies deployed, so that there are no limitations on how the applications behave while they are being developed, or alternatively there might be some quite restrictive policies in place so that any problems can be quickly discovered.

In a production platform, policies are used to ensure that all applications can run successfully on the platform, with some policies ensuring that certain resources are not over-committed, and others perhaps limiting particular resources to quite strict thresholds.

For application and operation scopes, policies can be used to enable the application to have sufficient resources for its needs without exceeding the use that would be expected. For example, if an application needs to make a significant number of database accesses, a Database request policy might be included with the application that has a high threshold value.

If that application is deployed to a region type within a platform that is more restrictive in the database access that it allows, it will soon be discovered that the region type is not suitable for this application. If an application is not intended to make any updates to files, a policy with appropriate File request policy rules could be deployed to ensure that it does not do so.

This last example, of a policy to limit file updates, is a good example where multiple policy rules might be included in a single policy definition. There could be a policy rule for each of the file rule items that relate to file update requests.

Guidance from tools like CICS Performance Analyzer, from knowledge of the application, and so on, should be used to set appropriate policy values. CICS Performance Analyzer V5.1 and later provide information about the platform, application, and operation to which data relates. This information can be used to establish the normal use of resources by the application, and then inform the setting to be used in policy rules.

Another general consideration when defining and deploying policies is governance of the policy definitions. Policies should be regarded as similar to source code or system configuration settings, and controlled and managed using similar techniques, such as source code management. The packaging of policies with applications or platforms helps to provide the structure for doing this, but the choice of governance technique depends on the installation.

8.5 Policy scenarios

This section describes a few examples of policy rules, and how they could be used to manage the behavior of our example GeneralInsuranceCustomer application.

Use of the term *policy* in this book to refer to both the CICS TS threshold mechanism and the insurance policies managed by the GENAPP might be a little confusing. However, the CICS Policy mechanism is actually somewhat like an insurance policy, which spreads the risk across the CICS TS platform and is activated when problems occur.

The operation to Inquire on an Insurance Policy needs to be responsive (it might, for example, be issued by an insurance agent when a customer needs to make a claim). A policy can be deployed that checks the CPU time of tasks performing the Inquire operation, and emit an event to a monitoring dashboard, enabling the IT department to keep an eye on this time use.

Deleting an Insurance Policy is not critical to the overall performance of the application, so perhaps a policy could be used to limit its virtual storage usage, ensuring that it does not use a disproportionate amount at times when the CICS TS platform is busy.

The development platform might be used to catch some poor performance practices in the GeneralInsuranceCustomer Application before new versions of the application are made available to test. For example, policies installed in the development platform could check for excessive use of 24-bit storage, or for use of shared storage, which might make it difficult to deploy parts of the application across different cloned regions in a region type.

8.6 Concluding thoughts

This chapter has described how threshold policy rules can be used to control and automate the behavior of applications in a cloud-enabled CICS TS environment. The scope to which policies can be applied ranges from the tasks running on region types within a CICS TS platform, to tasks running as a result of individual operations within an application.



Application lifecycle management

In Part II, we created the GeneralInsuranceDev platform and the GeneralInsuranceCustomer IBM Customer Information Control System (CICS) Transaction Server (CICS TS) application. In Chapter 4, “Creating an application” on page 51, we saw how, by defining entry points for our application, we can enable application-level monitoring and threshold policies.

In Chapter 5, “Applying a policy” on page 97, we added a new policy to the application and updated its version to 1.0.1, so that new messages emitted as a result of the policy addition could be attributed to the changes in version 1.0.1. Then in Chapter 6, “Packaging an application for multiversion deployment” on page 127, we saw how we could enable multiple versions of the same application to run in the same CICS TS regions at the same time, by defining private CICS TS program and library resources within the application.

We updated the application to version 1.0.2 when we repackaged the applications program and library resources to enable multi-versioning, and then we created version 1.100.0 of the application and deployed it alongside version 1.0.2 on the GeneralInsuranceDev platform.

In this chapter, we describe why we chose the version numbers that we did in Part 2, “How to cloud-enable your IBM CICS TS” on page 23. We discuss semantic versioning, which we suggest for the new cloud resources, and which was used in previous chapters. We examine in more detail the benefits of being able to run multiple versions of an application at the same time, and we discuss how these techniques can be applied to the GeneralInsuranceCustomer application around which we have based our examples.

This chapter provides information about the following topics:

- ▶ 9.1, “Managing application change through versioning” on page 194
- ▶ 9.2, “Managing application deployment through versioning” on page 195
- ▶ 9.3, “Applying versioning to GeneralInsuranceCustomer” on page 200
- ▶ 9.4, “Evolving the GeneralInsuranceCustomer application” on page 202
- ▶ 9.5, “Promoting GeneralInsuranceCustomer from dev to test” on page 203
- ▶ 9.6, “GeneralInsuranceCustomer deployment lifecycle” on page 203
- ▶ 9.7, “Outcomes” on page 204

9.1 Managing application change through versioning

Applications evolve in response to business needs. Developers fix bugs, architects design new capabilities, and system programmers use platform enhancements. We need a way of expressing the nature of these changes, in addition to understanding their effect on existing users, applications, and data.

We also need a way to package and manage the *lifecycle* of these changes, including reverting to a previous version if we encounter problems. We want users of our applications to experience a seamless transition to a new version. We might also want to phase in a new version of an application, enabling a subset of our users to try out the new version before rolling everyone onto it.

9.1.1 Semantic versioning

CICS TS V5.1 introduced a new CICS TS application resource that requires a *version* and a *name*. A version must also be specified for a CICS bundle that is to be installed as part of either a CICS TS application or a CICS TS platform. CICS application bindings have a version too. Why is this version needed, and how should you use it?

We need to separate describing *what* an application does and its name, which does not change, from *how well* it does it and its version, which does change. We are all familiar with software product versions, from operating systems and mobile phone applications to CICS TS.

The versioning system suggested for CICS TS cloud enablement, and the one used in this IBM Redbooks publication, is described in the technical white paper *Semantic Versioning*, by the Open Service Gateway initiative (OSGi Alliance):

<http://www.osgi.org/wiki/uploads/Links/SemanticVersioning.pdf>

It is widely used in OSGi-based projects, such as Eclipse, and was first seen in CICS TS V4.2 with the Java class library for CICS (JCICS) application programming interface (API) in both the CICS Explorer software development kit (SDK) and Java virtual machine (JVM) server run time. It enables the author of an API or service to clearly describe the nature of a change in the implementation, so clients can understand any compatibility issues with earlier versions.

The version attribute used for a CICS TS application, a CICS bundle, or an OSGi bundle takes the form `<major>.<minor>.<micro>`, for example, 2.3.1. The version should be updated manually by whoever is responsible for maintaining the resource concerned, for example, the application developer. The version should be the first change made as part of a new activity. The version part changes should be used in the following ways:

- ▶ Micro, for example, from 1.0.0 to 1.0.1. No external change (for example, a bug fix).
- ▶ Minor, for example, from 1.0.1 to 1.100.0. Compatible with an earlier version or an external change (for example, existing clients are unaffected).

Note: You might be wondering why we chose to increase the minor version by 100 rather than by 1. This technique is valuable when you want to run multiple versions of the application simultaneously. Imagine a scenario where version 1.0.0 is in production and you add in version 1.1.0. For awhile, both versions run without problems at the same time.

Then, a bug is found in version 1.0.0 that requires a small change to the application externals. Ideally, we would increase the minor version number, because the externals have changed, but V1.1.0 is already being used. If instead we had used version 1.100.0 when we added the new version of the application, we could have fixed the bug in V1.0.0 and increased its minor version number to reflect the fact that a change to the externals of the application was made.

This technique is used by CICS TS to version the OSGi bundles included with the JCICS APIs.

- Major, for example, from 1.100.0 to 2.0.0. Change is incompatible with earlier versions (for example, a different file record format, or an operation is removed).

9.1.2 Development lifecycle

The version is also extremely important for change management and problem diagnosis. You might ask the question, “Which version of the GeneralInsuranceCustomer application is running on the GeneralInsuranceTest platform?” You can track changes back through the lifecycle, to a user workspace and eventually to version control (for example, IBM Rational® Team Concert).

The versions of the individual CICS bundles that compose the application are shown in CICS Explorer, and the CICS TS application version is recorded in monitoring data. In Chapter 7, “Measurement by application” on page 175, we show how this can be used to compare the behavior and performance of different versions of the same application.

9.2 Managing application deployment through versioning

CICS TS V5.2 introduces the ability to run multiple versions of the same CICS TS application on the same CICS TS platform (for example, the same set of CICS TS regions). If an Application A defines a CICS TS program called ADD, version 1.0.0 and version 1.0.1 of Application A can each run their own different private version of the ADD program. Deploying multiple application versions was demonstrated in Chapter 6, “Packaging an application for multiversion deployment” on page 127.

This ability to run multiple application versions simultaneously, enables CICS TS applications to be updated with no outage of the application for customers. It enables different users to start different versions of the application, and it makes it simple to switch back to a stable version of an application if a new version turns out to have errors.

9.2.1 Switching application versions with no outage

As demonstrated in Chapter 6, “Packaging an application for multiversion deployment” on page 127, you can now Install, Enable, and Make Available multiple versions of the same application. When CICS TS detects that two or more versions of the same application are Available, it automatically routes requests to the version with the highest version number.

Figure 9-1 demonstrates how EXEC CICS LINK requests to an application entry point program always get routed to the highest available application version. GeneralInsuranceCustomer V1.0.2 is Enabled and Available. GeneralInsuranceCustomer V1.100.0 is ready for deployment. It is Installed and Enabled. At this point, work still flows to V1.0.2. Now V1.100.0 is made Available. New requests flow to V1.100.0, while requests already being processed in V1.0.2 complete in V1.0.2.

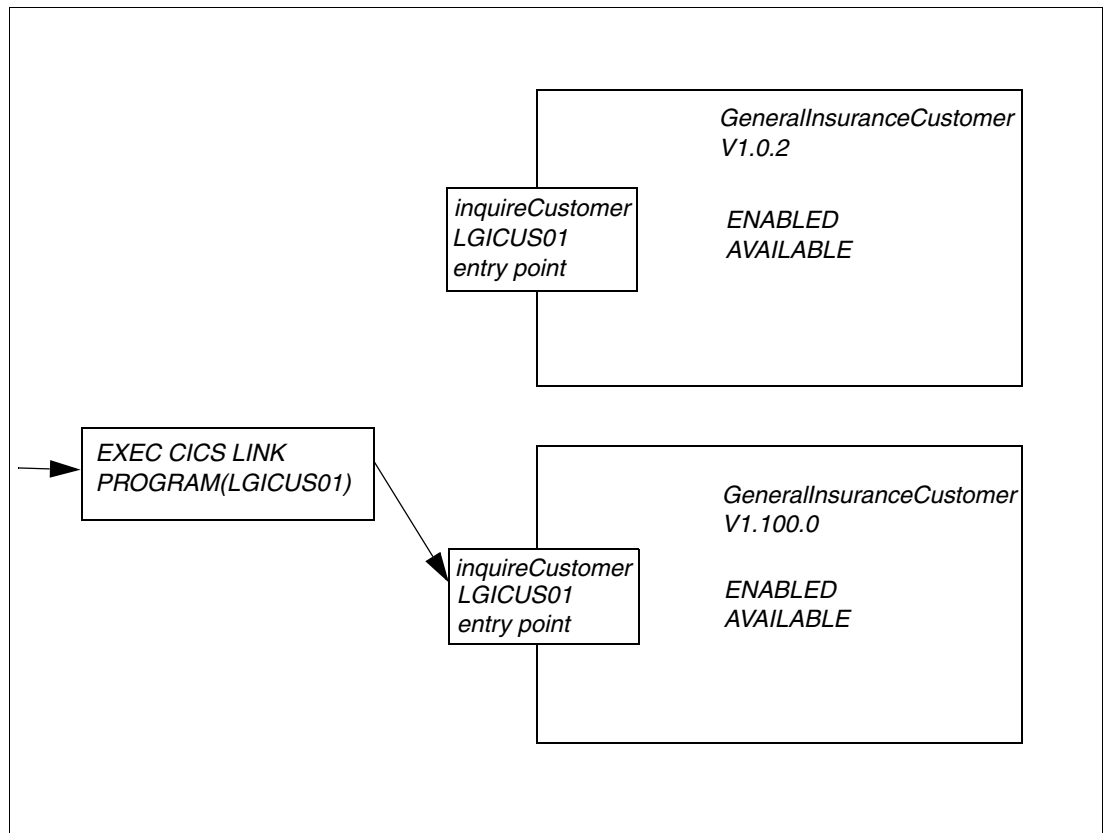


Figure 9-1 Linking to an entry point program always picks the highest AVAILABLE application version

If, having made a new version of the application available, you find that there is a problem with it, simply set it to Unavailable.

Figure 9-2 gives an example. GeneralInsuranceCustomer V1.100.0 has a problem. The decision is made to revert to V1.0.2. GeneralInsuranceCustomer V1.100.0 is Made Unavailable. Now, new requests flow to V1.0.2, while requests already being processed in V1.100.0 complete in V1.100.0.

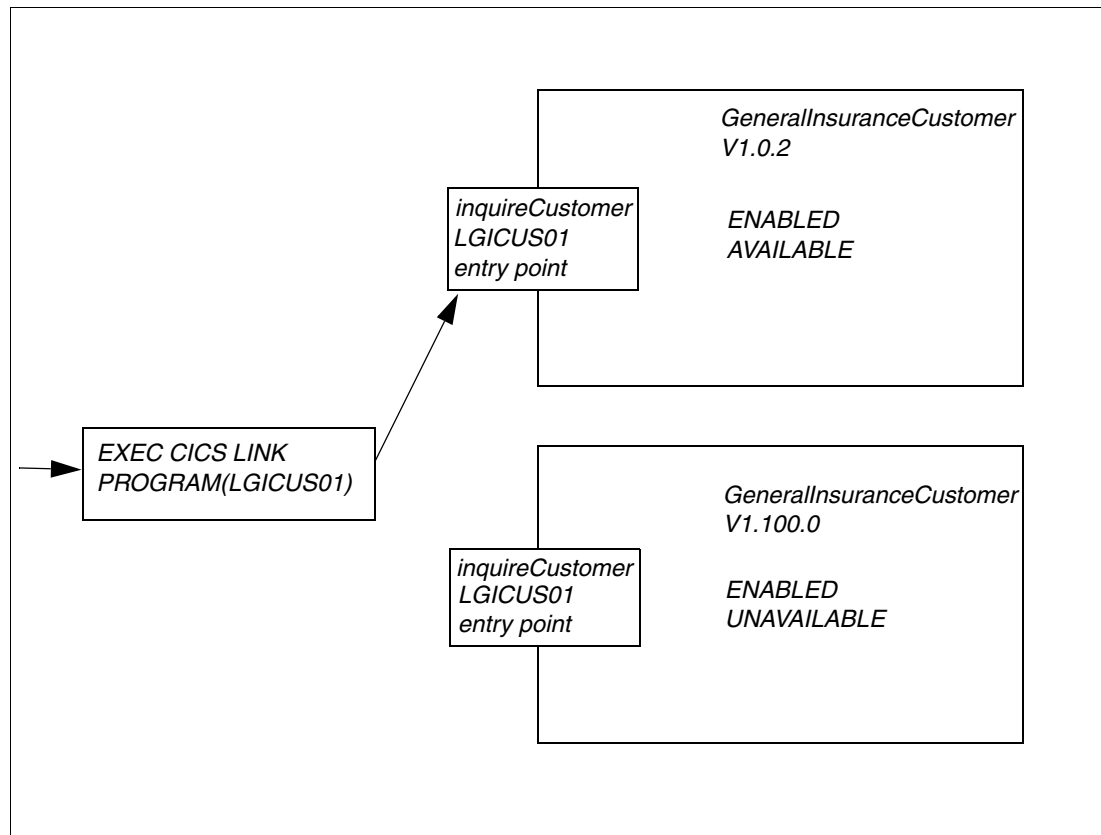


Figure 9-2 Version 1.100.0 is UNAVAILABLE so EXEC CICS LINKs revert to V1.0.2

Whether moving to a new version, or removing a version, the transition occurs without the need for an application outage.

9.2.2 Phasing in a new application

The ability to run multiple versions of the same application in parallel can enable a model whereby a subset of your customers gets to try out the latest version of the application, while the majority continues to use an earlier version. This can be achieved in different ways:

- A router program is in front of your application, and decides which version of the application a particular request should be sent too. It uses a new CICS TS API command, **EXEC CICS start APPLICATION**, to call an explicit application version. The **start APPLICATION** command enables the user to specify the target major version and minor version of the application. The latest micro version (bug fix) is always used.

Figure 9-3 shows an example. GeneralInsuranceCustomer V1.100.5 and V1.200.0 are both Enabled and Available. The router program decides that an inbound request should be sent to V1.100. It issues the following command (one line):

```
EXEC CICS start APPLICATION(GeneralInsuranceCustomer)
OPERATION(inquireCustomer) MAJORVERSION(1) MINORVERSION(100) EXACTMATCH
```

The request is routed to V1.100.5 of the application.

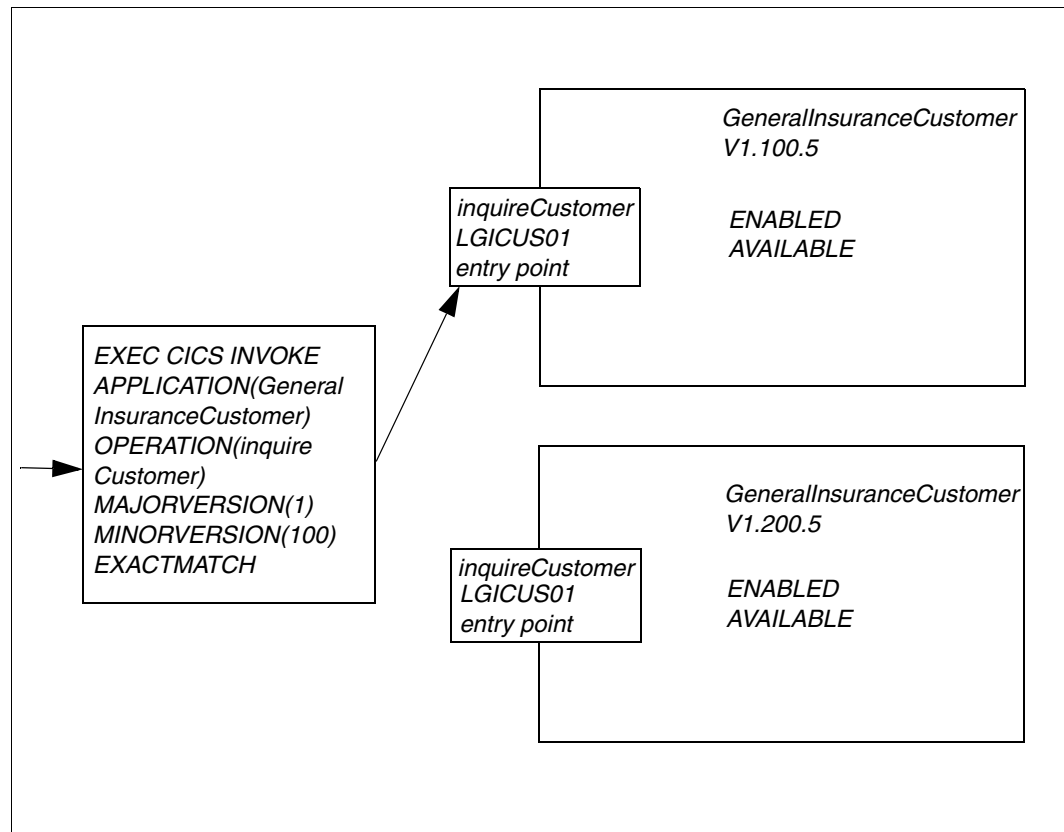


Figure 9-3 A router program explicitly targets a specific application version

- The different versions of the application expose different application entry points, providing external routes into the different versions of the application.

Figure 9-4 shows an example. GeneralInsuranceCustomer V1.300.0 includes a web services front end to the application. A URIMAP resource is defined in the application binding, and is enabled as an application entry point. When inbound Hypertext Transfer Protocol (HTTP) requests match the URIMAP, they are directed into V1.300.0 of the application.

Now a new version, V1.400.0, is deployed. In its application binding, it defines a different URIMAP as its entry point. Both versions of the application can be installed, enabled, and made available at the same time. The web service requester can choose which version of the application they start by altering the Uniform Resource Identifier (URI) that they send the request message to.

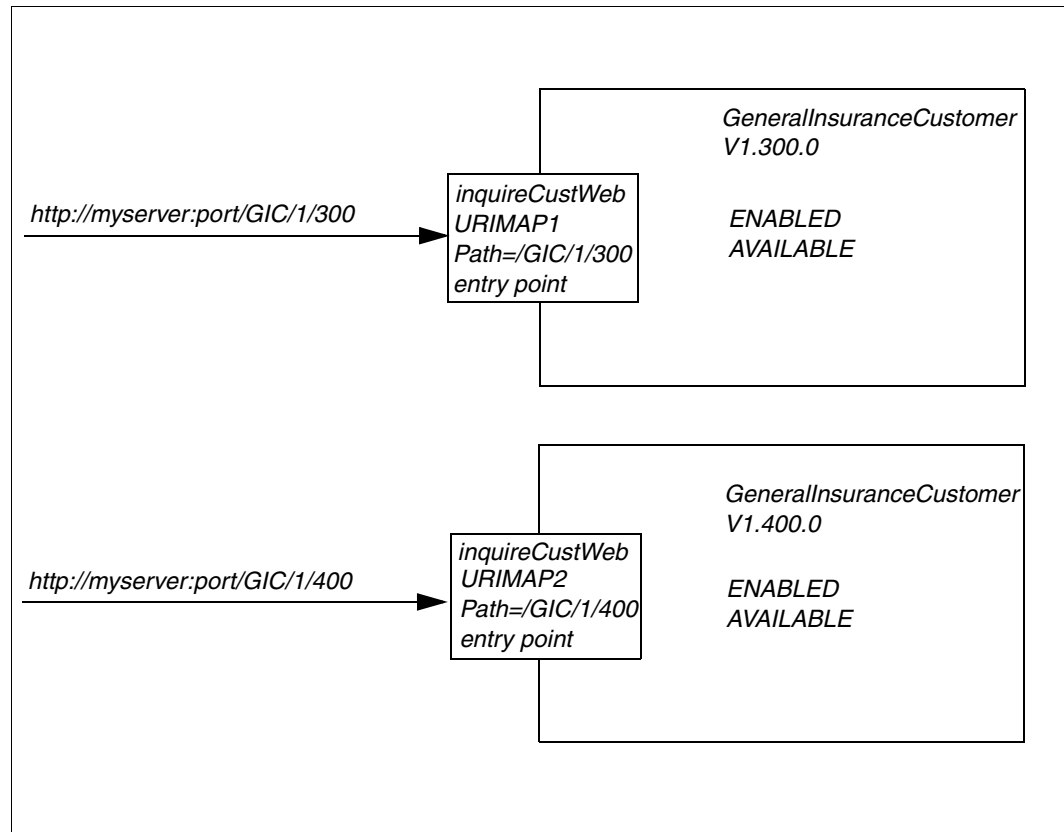


Figure 9-4 URIMAP entry points match different web requests, and associate them to an application version

9.2.3 CICS TS region consolidation

Continued improvements to the scalability of CICS TS have enabled customers to consolidate regions, save costs, and minimize the number of CICS TS regions that need to be managed. This can be a challenge, because different applications can use the same CICS TS program names, making it impossible to run both applications on the same CICS TS region without changes.

By packaging these applications as CICS TS applications, you can resolve this problem. CICS TS program and library resources defined within a CICS TS application are considered private to that application. Therefore, different Applications can be installed, enabled, and made available to run on the same CICS TS regions, even if they use the same CICS TS program names.

Figure 9-5 shows an example where Application GeneralInsuranceCustomer V2.0.0 defines a CICS PROGRAM called ADD. The IT department wants to run the GeneralInsuranceCustomer Application on the same CICS TS platform (the same CICS TS regions) as another Application Calculator V1.0.0. The Calculator program also defines a CICS PROGRAM called ADD.

This is acceptable, though. Because the programs are defined as part of CICS TS applications, both can be installed and run at the same time, and each uses its own private copy of PROGRAM ADD. This would not have been possible if the programs were defined in the CICS system definition (CSD).

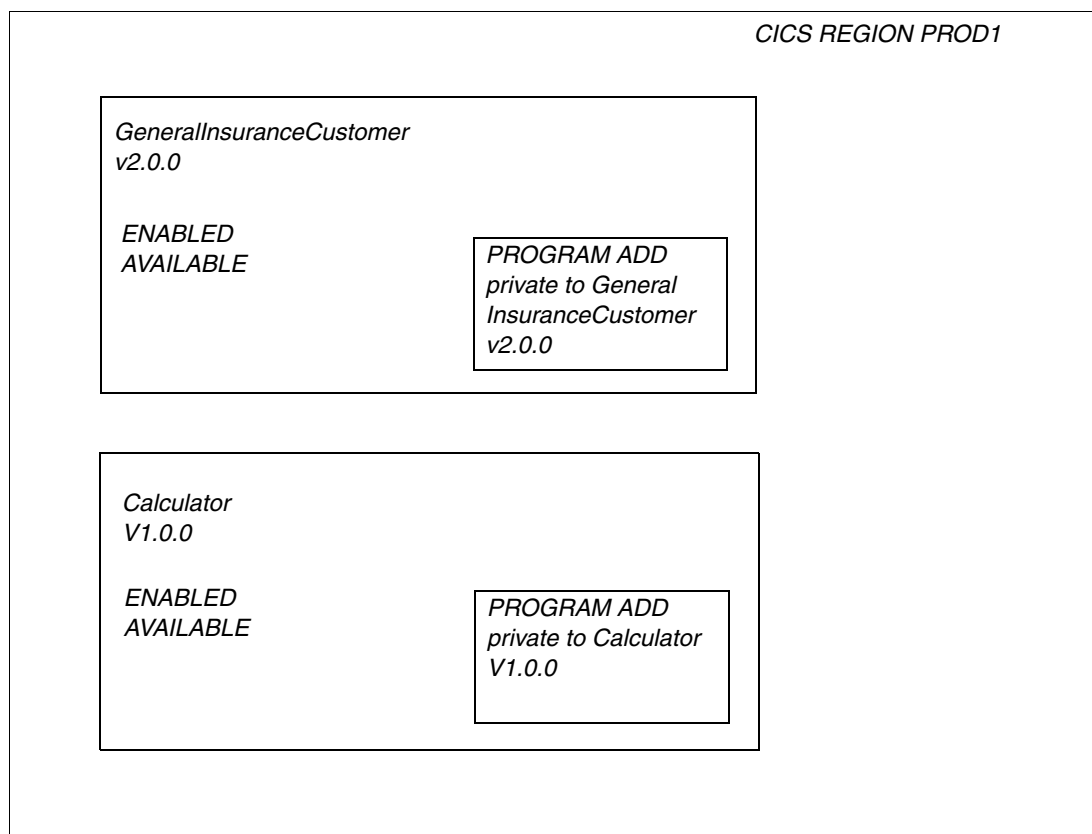


Figure 9-5 Enabling region consolidation for applications with clashing PROGRAM resources

9.3 Applying versioning to GeneralInsuranceCustomer

In this section, we discuss how semantic versioning can be applied to the GeneralInsuranceCustomer application, both during the application *development* lifecycle, and the application *deployment* lifecycle.

9.3.1 Using semantic versioning

In Chapter 4, “Creating an application” on page 51, we created version 1.0.0 of the GeneralInsuranceCustomer application, including version 1.0.0 of all of the CICS bundles that it consists of. In Chapter 5, “Applying a policy” on page 97, we added a new policy to the application, and updated its version to 1.0.1, so that new messages emitted as a result of the policy addition could be attributed to the changes in version 1.0.1.

In Chapter 6, “Packaging an application for multiversion deployment” on page 127, we repackaged the application to define a subset of the resources, to enable multiple versions of the application to be made available at the same time, on the same CICS TS platform. Before making the change, we updated the CICS bundle micro version and the CICS TS application micro version (version 1.0.2).

This is because the changes had no compatibility issues with earlier versions, and no external application changes. For example, a user would see no change in the capability of the application.

We then demonstrated creating a new version of the application to implement a new requirement to protect certain customer data fields from unauthorized users. We updated the application minor and micro version numbers to 1.100.0. This is because the change introduced new functionality, and made a change to the externals of the application, but was compatible with earlier versions.

It is important to track changes through both the development and deployment lifecycles, so that if a problem did occur, it would be possible to trace the change back to the user that made it. For example, suppose that you repackage the GeneralInsuranceApplication as shown in Chapter 6, “Packaging an application for multiversion deployment” on page 127.

If you forgot to discard a LIBRARY resource or remove its definition from the CSD group that is installed at start, there would be a name clash, and the CICS TS application would fail to install completely. Knowing the version of both the CICS TS application and the failing CICS bundle would help to resolve the problem, if you know the changes that have been made in each version.

Another change that would require an update to both the CICS bundle and CICS TS application is a bug fix. Although such a change might not affect the externals, it can affect application behavior. Then, an analysis of any monitoring data can take advantage of the full application name and version.

9.3.2 Semantic versioning in the development lifecycle

Application change is an ongoing process requiring an appropriate update to version information for each separate piece of work. For example, it might not be possible to repackage all resources that make up an application at the same time, due to restrictions in the set of resources that support multiple versions of an application running simultaneously on the same CICS TS platform.

If and when additional resources are supported by CICS TS in a future release, you might want to revisit the application and make changes to it. In each case, the version of the individual CICS bundles and overall CICS TS application must be updated again.

In addition to considering the overall CICS TS application version, changes at the CICS bundle and, if using Java, the OSGi bundle level, can also be determined. This is because the version number is central to the application lifecycle. When assembling an application, the developer chooses the specific version of each CICS bundle used.

9.3.3 Semantic versioning in the deployment lifecycle

When creating a CICS TS application definition, the version of both the CICS TS application and CICS application binding are specified. Finally, when the application is installed, the service provider chooses which versions of the application to make available.

Version information can be extremely helpful when deciding where or whether an application should be deployed after it is promoted through the development lifecycle. The following list describes some examples:

Micro	Low risk. Deployment might be allowed or even required outside the normal change window.
Minor	Requires regression testing for existing clients, and might require infrastructure change, for example, new platform dependencies.
Major	Can require platform separation, for example, data format change.

9.4 Evolving the GeneralInsuranceCustomer application

The GeneralInsuranceCustomer CICS TS application we have used for the examples in this IBM Redbooks publication has a traditional IBM 3270 terminal interface. We could modernize it in an update, introducing a web services interface, a capability that is available in the General Insurance Application (GENAPP). According to the definition we used in 9.1.1, “Semantic versioning” on page 194, this would be a change to the externals, but one that is compatible with earlier versions (existing IBM 3270 clients are unaffected).

There are no changes to the customer records, so there are no compatibility issues for the database (or the Virtual Storage Access Method (VSAM) files). This means that we could deploy the new version onto the same platform as the previous one at the same time if we chose to. If any problems were encountered, the update could be backed out without any risk of data corruption.

To add the web service interface, we would create a new CICS bundle, GeneralInsuranceCustomerPresentationWeb, so that there are no changes to existing bundles, and all updates to the application are kept in one place. This makes it clear when developing, deploying, and managing the application that we are not changing the existing IBM 3270 interface.

In the future, this approach would help us to distinguish between updates made to the IBM 3270 interface and those made to the web services. Also, we could more easily remove the traditional interface in the future if it is no longer required.

The next task would be to determine the new CICS TS application version. The new GeneralInsuranceCustomerPresentationWeb Bundle would have default version 1.0.0, which is acceptable because the bundle is new. We are making a change to the application that is compatible to an earlier version, so we increment the minor version of the application, for example, 1.200.0.

We would also need a Transmission Control Protocol/Internet Protocol (TCP/IP) service. In CICS TS V5.2, we can define TCP/IP services within CICS bundles. However, we might not want to include the TCP/IP service in the application, because TCP/IP services are often resources that are shared by multiple applications. In this case, it makes more sense to define the TCP/IP service as a part of the CICS TS platform, and to assert a dependency on the TCP/IP service in the application.

We could define a new CICS bundle which would be installed into the ApplicationServices region type. This can be added to the GeneralInsuranceDev platform so that when it is next installed, the TCP/IP service is defined. To save having to restart the platform and disrupt any other applications it might be hosting, we could use the technique described in Chapter 5, “Applying a policy” on page 97 to add a CICS bundle to a running platform.

9.5 Promoting GeneralInsuranceCustomer from dev to test

Suppose that unit testing of version 1.200.0 of GeneralInsuranceCustomer, using the single managed region GeneralInsuranceDev platform, is complete. The next step is to promote the application to the GeneralInsuranceTest quality assurance platform for further testing before promotion to production.

This platform is more representative of the production environment in that it has separate regions for the Terminals, ApplicationServices, and DataAccess region types. Deployment to this platform helps ensure that multiregion operation (MRO) is working correctly.

We need to create `general.insurance.customer.to.test.platform.binding`, a new application binding. The `general.insurance.customer.to.dev.platform.binding` application binding is similar, in that the GeneralInsuranceTest Platform has the same region types as GeneralInsuranceDev. However, the new application binding gives us the opportunity to manage the application lifecycle differently, and to make certain adjustments to the externals. In particular, we could use a different Uniform Resource Locator (URL) for the web services.

9.6 GeneralInsuranceCustomer deployment lifecycle

In this section, we describe how we could replace the existing GeneralInsuranceCustomer version 1.100.0 with version 1.200.0, and then back out the changes by simply reverting to the previous version. Chapter 4, “Creating an application” on page 51 introduced the application deployment lifecycle. In this section, we expand upon that capability.

You now have the new version 1.200.0 of GeneralInsuranceCustomer, which provides a web services interface. You need to deploy the application, then you can install it, enable it, and make it available to users.

The first step is to export version 1.200.0 of GeneralInsuranceCustomer as originally described in Chapter 4, “Creating an application” on page 51, creating a new APPLDEF in the process. This is a decision point for whoever is deploying the new application version. If, as described in Chapter 6, “Packaging an application for multiversion deployment” on page 127, we have merely implemented an internal structural change reflected in an increase in the micro version (for example, 1.0.2), there should be little risk installing the new version.

However, if we were about to install version 2.0.0 of the application, it would be prudent to disable and discard the old version, and to delete the old APPLDEF. This is because an increase in the major version indicates that we have introduced a change that is not compatible with an earlier version of the application, including perhaps a change in record formats. Removing the old application version prevents possible corruption to the database.

In this example, an increase in the minor version indicates a change in the application that is compatible with an earlier version. There are new externals to test, and new dependencies on the platform that must be validated, but it should be safe to revert to the previous version if necessary.

By keeping the older application installed and available to users, we can ensure that our new version can be successfully installed and enabled without any outage to our customers. When we are confident that the new version is enabled, we can make it available. New requests for the application will now be routed to the new version.

If we encounter any problems, we can simply make the new version unavailable, which reverts new requests back to the old version. When we are comfortable with the new version, and we are confident that all requests being processed by the old version of the application have completed, we can disable and discard it. We have successfully transitioned to a new application version across multiple CICS TS regions, with no application outage.

9.7 Outcomes

In this chapter, we described in detail the suggested system of semantic versioning, and how it can be used to record and track changes to GeneralInsuranceCustomer, in addition to communicating how those changes affect clients, data, and other applications running on the same platform. We introduced scenarios demonstrating the value of being able to run multiple versions of the same CICS TS application on the same CICS TS platform at the same time.

Then, we discussed how we could move version 1.200.0 of GeneralInsuranceCustomer, unchanged, from a single region development environment to a multi-region test environment using a different application binding. This binding allows for changes in the underlying platform, and differences in operational procedures between the environments. Finally, we showed how you can transition between different versions of an application, to either upgrade capability or back out a change, without causing an outage of the application.



Part 4

Appendixes

This part of the book provides detailed instructions about how to set up the initial General Insurance Application (GENAPP) and the IBM Customer Information Control System (CICS) Explorer tooling.



A

Setup and environment

This appendix provides a detailed description of how to set up the General Insurance Application (GENAPP) in a single managed region, and in an IBM Customer Information Control System (CICS) Transaction Server (CICS TS) topology.

This appendix provides information about the following topics:

- ▶ “Installing the general insurance application” on page 208
- ▶ “Building the application environment” on page 212
- ▶ “Build the CICS TS environment” on page 213
- ▶ “Testing the general insurance application” on page 217

Installing the general insurance application

The general insurance application is available to download from the IBM SupportPac website, and contains the jobs you need to set up the application.

Before you begin

For the support of the CICS TS examples in this IBM Redbooks publication, the implementation of GENAPP requires the following setup, as outlined in Figure A-1 on page 209:

- A single managed CICS TS region. GENAPP can be installed into a stand-alone CICS TS region System Management Single Server (SMSS) that is not managed using IBM CICSplex System Manager (CICSplex SM). However, the concepts of applications and platforms discussed in this Redbooks publication *require* the region to be managed by CICSplex SM, which is referred to as *single managed region*.

This region is used in Chapter 3, “Creating a platform” on page 25, to demonstrate how to create a development CICS TS platform with three region types that all map to the same underlying CICS TS region. The platform is used for all further examples in this book.

- Optionally, a CICS TS topology. Chapter 3, “Creating a platform” on page 25, demonstrates how to create a test platform, with three region types, that map to three different CICS TS regions. Chapter 9, “Application lifecycle management” on page 193, describes how an application could be promoted from the development platform to the test platform.

However, the test platform is *not* required for the examples in this book. For this configuration, connectivity is required from the terminal-owning region (TOR) to the application-owning region (AOR), and from AOR to the data-owning region (DOR). The connection names are used on some program definitions in the @CDEF122 member.

This topology needs to be managed by CICSplex SM. As part of the customization, sample job control language (JCL) is built for the CICSplex SM address space (CMAS), web user interface (UI), and optional coupling facility components.

Tip: Although it is possible to share a CICS system definition (CSD) between the single managed region and the CICS TS topology, we suggest that you use separate CSDs to avoid conflicts.

You must also have a supported version of IBM DB2 and a Common Business Oriented Language (COBOL) compiler installed.

Remember: If you do not have DB2 or a COBOL compiler, you can download a version of GENAPP called *GENAPP Lite* from the CICS TS SupportPac website. This can be used for all but one of the examples. In Chapter 6, “Packaging an application for multiversion deployment” on page 127, we demonstrate making a small change to the application source and deploying a new version of the application, which does assume that you have the application source and can compile it using a COBOL compiler.

The CICS TS components, a named counter server and a Shared CICS TS Queue server, are optional.

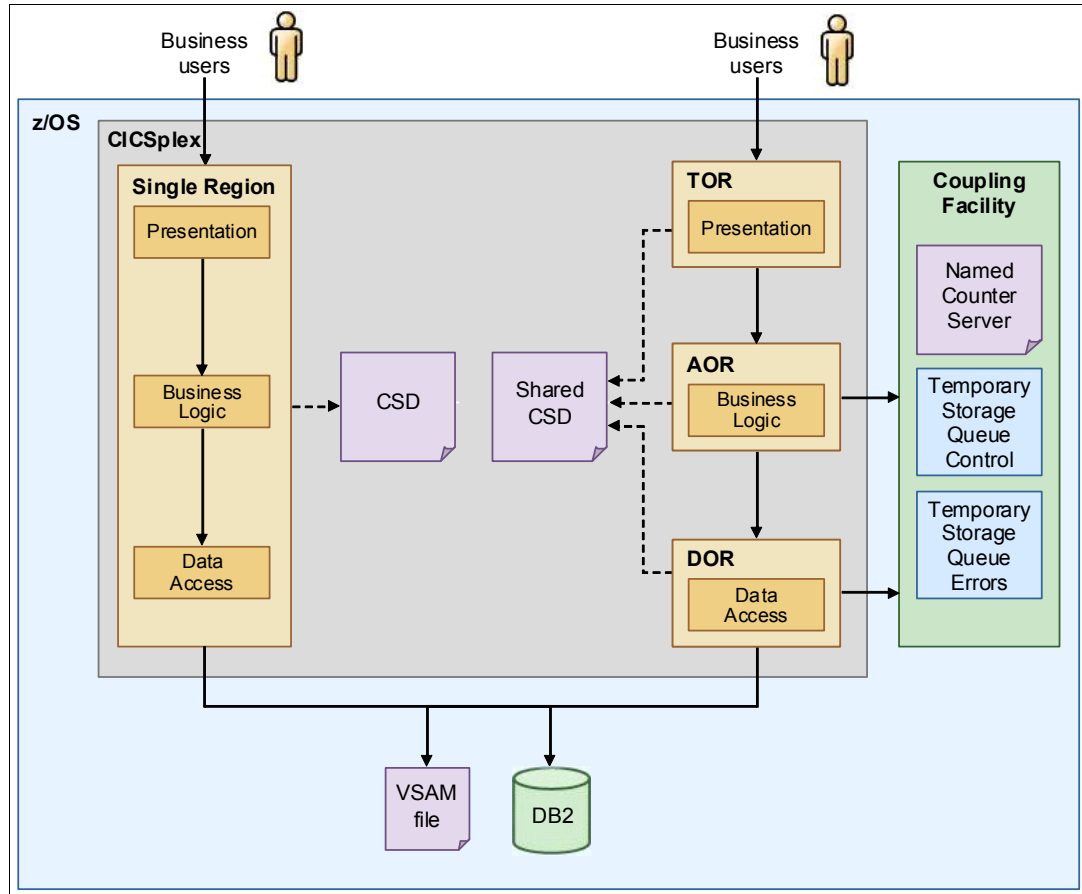


Figure A-1 Single managed region and CICS TS topology

Important: You can also find the instructions for installing GENAPP to a single CICS TS region and to a CICS TS topology in the CICS TS Knowledge Center at the following link:

http://www-01.ibm.com/support/knowledgecenter/SSGMCP_5.2.0/com.ibm.cics.ts.samples.doc/topics/genapp.html

Note that if you follow those instructions, you should not reuse the TOR as a single region as suggested. In addition, do not define the GENAPP load libraries using request parameter lists (RPLs), but use dynamic libraries as described in “Add GENAPP load library to CICS TS as a dynamic LIBRARY” on page 215.

Procedure to customize

Follow these instructions to install GENAPP:

1. Go to the IBM support and downloads website and download the `genapp.zip` file to your workstation. The website Uniform Resource Locator (URL) can be obtained by searching on the web for GENAPP SUPPORTPAC CB12.
2. Extract the files to a suitable directory.

The following files are extracted, as shown in Table A-1.

Table A-1 List of extracted files

File	Description
CNTL.XMIT	JCL to build the environment for the application
EXEC.XMIT	REXX code that customizes the JCL
KSDSCUST	Sample customer information for the application
KSDSPOLY	Sample policy information for the application
SOURCE.XMIT	Source programs for the general insurance application
WSIM.XMIT	Sample workload simulator scripts

3. Use File Transfer Protocol (FTP) to transfer the files to the System z machine and put them in IBM Multiple Virtual Storage (IBM MVS) data sets. Call the data sets `<userid>.CB12.<filename>`, where `<filename>` is the name of the extracted file (for example, `CICSUSR.CB12.SOURCE.XMIT`):
 - a. Transfer the files with the suffix `.XMIT` in binary mode.
 - b. Enter `bin` on the command line to change to binary mode.
 - c. Transfer the `KSDSCUST` file in American Standard Code for Information Interchange (ASCII) mode.
 - d. Enter `ascii` on the command line to change to ASCII mode.If necessary, set the record length of the data set for the `KSDSCUST` file before transferring it. Enter the following command to set the record length:
quote site lrecl=80 blksize=3120
 - e. Transfer the `KSDSPOLY` file in ASCII mode.
 - f. Set the record length of the data set for the `KSDSPOLY` file before transferring it. Enter the following command to set the record length:
quote site lrecl=64 blksize=6400
4. Extract the `SOURCE`, `WSIM`, `CNTL`, and `EXEC` files into partitioned data sets by using the following `RECEIVE` command:
RECEIVE INDSN('<userid>.CB12.SOURCE.XMIT')

This command identifies the input data set. You must enter a response to the **RECEIVE** command to identify the name of the destination partitioned data set, as shown in the following example:

DA('userid.CB12.SOURCE')

You do not have to perform this step for the `KSDS` files, because they are already in a readable format.
5. Customize the `CUST1` member in the `CB12.EXEC` data set. The values in this member include settings for the CICS TS topology and CICSplex SM.

Replace the values used in Example A-1 with yours. Table A-2 explains the parameters.

Example A-1 userid.CB12.EXEC(CUST) member for customization

```

PDSMEMin = 'userid.CB12.CNTL'
CICSHLQ   = 'CTS510.CICS'
CPSMHLQ   = 'CTS510.CPSM'
CSDNAME    = 'userid.GENAPP.DFHCSO'
USRHLQ     = 'userid'
COBOLHLQ   = 'PP.COBO1390.V420'
DB2HLQ     = 'SYS2.DB2.V910'
CEEHLQ     = 'CEE'
DB2RUN     = 'DSN910PM'
SQLID      = 'STTESTER'
DB2SSID    = 'DHM1'
DB2DBID    = 'GENASA1'
TORAPPL    = 'CICSTOR1'
AORAPPL    = 'CICSAOR1'
DORAPPL    = 'CICSDOR1'
TORSYSID   = 'TOR1'
AORSYSID   = 'AOR1'
DORSYSID   = 'DOR1'
CMASAPPL   = 'CICSCMAS'
CMASYSID   = 'ICMA'
WUIAPPL    = 'CICSWUI'
WUISYSID   = 'IWUI'
WSIMHLQ    = 'WSIM.V110'

```

Table A-2 Description of the customization parameters

Parameter	Description
PDSMEMin	Enter the location of the CNTL library, <code>userid.CB12.CNTL</code> . Replace <code>userid</code> with your user ID.
CICSHLQ	Enter the CICS TS high-level qualifier to customize the CICS TS data sets.
CPSMHLQ	Enter the CICSplex SM high-level qualifier to customize the CICS TS data sets.
CSDNAME	Enter the fully qualified name of the CSD for the CICS TS regions that will run the general insurance application.
USRHLQ	Enter a high-level qualifier for the application data sets.
COBOLHLQ	Enter the high-level qualifier for the COBOL compiler.
DB2HLQ	Enter the high-level qualifier for DB2 libraries.
CEEHLQ	Enter the high-level qualifier for IBM Language Environment®.
DB2RUN	Enter the high-level qualifier for the DB2 runtime library.
SQLID	Enter the IBM Resource Access Control Facility (IBM RACF®) user ID that is authorized to create objects in DB2.
DB2SSID	Enter the subsystem ID of the DB2 instance that you want to use.
DB2DBID	Enter a name for the database that is going to contain the general insurance application data. You can use any value.
WSIMHLQ	Enter the high-level qualifier for the Workload Simulator for IBM z/OS libraries. If you do not use this product, you can ignore this parameter.

6. Run the **EXEC** job to customize the JCL in the CNTL data set. The job copies the members and updates them with your values. Each member name is prefixed with @ to indicate that it has been customized.

Results

You have successfully installed and customized the GENAPP jobs.

The next step is to build the environment for the application.

Building the application environment

Run the customized jobs to create the DB2 database, application files, and CICS TS resources.

About this task

To set up GENAPP, run each of the customized jobs shown in the following sections from the CB12.CNTL data set.

Some jobs are optional, and depend on whether you want to use a coupling facility to share application data across CICS TS regions.

Procedure

Follow these steps to set up GENAPP:

1. Submit the @ADEF121 job to create the Virtual Storage Access Method (VSAM) application files for customer details and policy details. This job populates the KSDSCUST and KSDSPOLY files with data.
2. Submit the @ASMMAP job to build the basic mapping support (BMS) maps for the IBM 3270 interface. This job has a return code of 4.
3. Submit the @COBOL job to compile the COBOL application programs. The compiled programs are put in the <USERHLQ>.CB12.LOAD library. This job has a return code of 4.
4. Verify that the name of the DB2 plan is correct in @DB2CRE, then **submit** the job to create the DB2 database for the application. It creates a storage group, database, tables, and indexes. This job also populates the database with sample data.
Note: @DB2DEL will drop the database that is created in the @DB2CRE job.
5. Submit the @DB2BIND job to bind the application to the DB2 objects.
6. Optional: If you want to use a named counter server with the application, create a structure in the coupling facility for a named counter server. The pool name is GENA and the structure name is DFHNCLS_GENA.
7. Optional: If you want to use a shared temporary storage queue, create a structure in the coupling facility for a shared temporary storage queue. The shared temporary storage queue is also called GENA and the structure name is DFHXQLS_GENA.

8. Optional: Submit the @SAMPNCS job to create a named counter server called GENA. You can customize this sample to change the pooled name if appropriate. The job is long-running. You can check that the job initialized successfully by looking in the job log for the following CICS TS message:

DFHNC0102I Named counter server for pool GENA is now active.

9. Optional: Submit the @SAMPTSQ job to create a temporary storage queue server called GENA. Again, you can customize the sample to change the pooled name if appropriate. This job should show successful initialization with the following CICS TS message:

DFHXQ0102I Shared TS queue server for pool GENA is now active.

Build the CICS TS environment

In the following steps, you make the required modifications in CICS TS to run GENAPP.

Single managed region

The following steps must be taken against a CICS TS region that is managed by CICSplex SM. The SupportPac does not contain explicit configuration for a single managed region. For further information about configuring CICSplex SM, see the CICS TS product documentation.

Perform the following steps to set up the single managed region:

1. Submit the @CDEF121 job to add all of the required resource definitions to the CSD of your single managed CICS TS region. The generated list GENALIST includes groups that contain the following components:
 - a. Transaction definitions
 - b. Local program definitions for the presentation, business, and data access logic
 - c. Definitions for data to be accessed (that is, VSAM files, DB2Connection, and DB2Entry)

Hint: The local program definitions can be omitted if the CICS TS auto-install program function is enabled.

2. Update the system initialization parameters to provide the following non-default values:
 - a. CPSMCONN=LMAS
 - b. GRPLIST=(DFHLIST,GENALIST)
 - c. DB2CONN=YES
 - d. Optional: NCPLDFT=GENA
3. Start the CICS TS region.

CICS TS topology

The following steps assume that you already have three CICS TS regions interconnected and managed by CICSplex SM.

Tip: For help configuring GENAPP for a CICS TS topology, see *Creating a CICS topology that is managed by CICSplex System Manager* in the IBM Knowledge Center:

http://www-01.ibm.com/support/knowledgecenter/SSGMCP_5.2.0/com.ibm.cics.ts.scenarios.doc/genapp_cpsm/scenario_overview.html

Perform the following steps to set up the CICS TS topology:

1. Submit the job @CDEF122 to add all of the required resource definitions to the shared CSD of your CICS TS topology. We suggest using a separate CSD from the single managed region.

Important: Verify that the Remote system parameter on the program definitions match your connection names for routing purposes.

2. The definitions are separated into three different lists to distribute the resources among the presentation, business logic, and data access regions:
 - a. The list for the presentation region (TORLIST) contains the following components:
 - Transaction definitions
 - Local program definitions for the presentation logic
 - Remote program definitions for routing requests to the business logic layer
 - b. The list for the business logic region (AORLIST) contains the following components:
 - Local program definitions for the business logic
 - Remote program definitions for routing requests to the data access layer
 - c. The list for the data access layer region (DORLIST) contains the following components:
 - Local program definitions for the data access logic
 - Definitions for data to be accessed

Hint: The local program definitions can be omitted if the CICS TS auto-install program function is enabled.

If you want to use CICSplex SM Workload Manager to manage where the work is routed, the remote programs are defined as DYNAMIC(YES) using the provided @CDEF123 job. If not, they are defined with a specific REMOTESYSTEM using the @CDEF122 job.

3. Various SIT overrides are necessary, in the following ways:
 - a. In all of the regions:
 - i. CPSMCONN=LMAS
 - ii. Optional: NCPLDFT=GENA
 - b. In the JCL for the presentation region, add the following system initialization parameter values:
 - i. GRPLIST=(DFHLIST,TORLIST)
 - ii. DB2CONN=NO
 - iii. ISC=YES

- c. In the JCL for the business logic region, add the following system initialization parameter values:
 - i. GRPLIST=(DFHLIST,AORLIST)
 - ii. DB2CONN=NO
 - iii. ISC=YES
- d. In the JCL for the data access region, add the following system initialization parameter values:
 - i. GRPLIST=(DFHLIST,DORLIST)
 - ii. ISC=YES
 - iii. DB2CONN=YES
4. CICS TS startup JCL must be modified. The data access region needs to have DB2 libraries adding to the STEPLIB and DFHRPL.
5. It is also necessary to add or modify the EYUPARM DD card to include CICSplex(GNAPPLEX) and CMASYSID(cmasysid) parameters.

Add GENAPP load library to CICS TS as a dynamic LIBRARY

The original GENAPP design suggests adding a single load library to the DFHRPL concatenation of the CICS JCL to make GENAPP load modules available to CICS TS. This leads to a disadvantage, because when CICS TS is running, changes to the DFHRPL data set names are not possible without stopping and restarting CICS TS. This is not usually an option in a continuously available environment.

Since CICS TS V3.2, program LIBRARY concatenations can be defined to CICS TS dynamically. Using dynamic LIBRARY concatenations provides several advantages for the system programmer and the organization:

- ▶ They contain one or more data sets from which program artifacts can be loaded.
- ▶ New applications for deployment can be brought into service at any time without affecting continuous availability.
- ▶ Existing applications in dynamic LIBRARY concatenations can easily be withdrawn from service without affecting continuous availability.
- ▶ Patches to existing applications can be installed easily, by installing them in a LIBRARY concatenation with a higher ranking than the existing LIBRARY, without affecting continuous availability.
- ▶ Data sets in dynamic LIBRARY concatenations can easily be taken offline for compression without affecting continuous availability.

Therefore, you should define and install your GENAPP load library to all CICS TS regions using a CICS TS LIBRARY resource. The following steps show how to do this with CICS Explorer:

1. Connect CICS Explorer to your CICSplex. See Appendix B, “IBM CICS Explorer setup” on page 219 for further information about how to do this.
2. Open the LIBRARY Definitions view. If it is not already inside your workspace, you can open it using one of the following methods:
 - a. From the menu bar, select **Window** → **Show View** → **Other**, and then select **CICS SM Definitions** → **LIBRARY Definitions**.
 - b. Press **Ctrl+3** and type LIBRARY Definitions until the Quick Access Menu offers **LIBRARY Definitions - CICS SM Definitions**.
3. Right-click anywhere inside the LIBRARY Definitions view and select **New**.

4. Enter the following attributes, as shown in Figure A-2:
 - CICSplex: The name of the CICSplex that manages your GENAPP region(s).
 - Region (CSD): The name of your single managed region. The library will be installed to its underlying CSD.

Remember: If you are configuring the single managed region and the CICS TS topology, and have used different CSDs for both, you need to repeat these steps to define the library onto both CSDs.

- Resource Group: GENASALI.
 - Name: GENALIB.
 - Ranking: 50 (the default).
 - Data Set Name 1: Specify the load library where GENAPP is installed. In Figure A-2, this is CICSSEM.CB12.LOAD.
 - Clear **Open Editor**.
5. Click **Finish**.

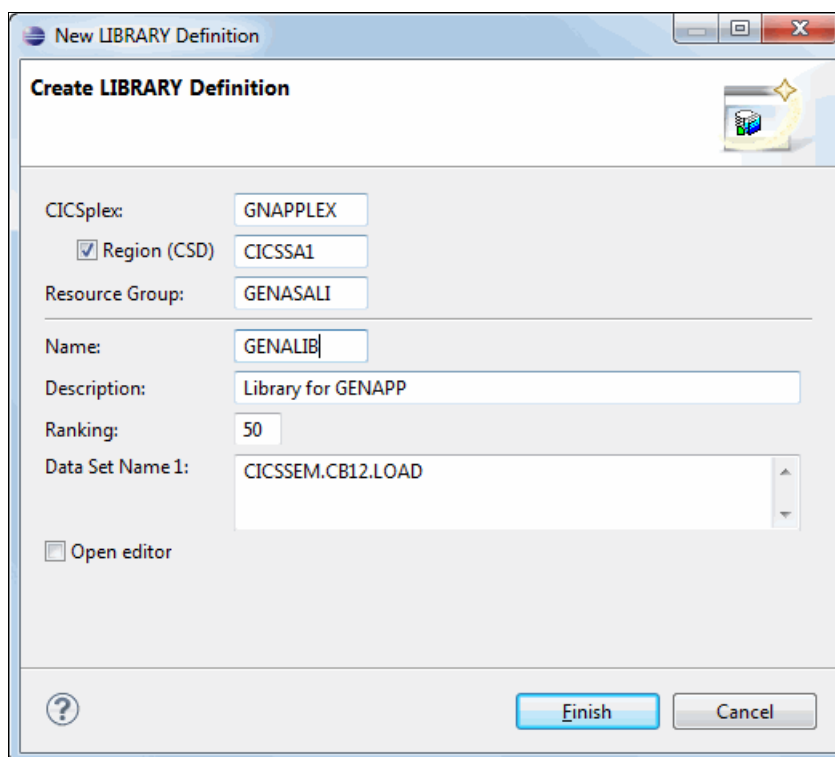


Figure A-2 Example definition of the GENALIB dynamic library

Remember: Although this library only contains one data set, it is possible to add up to 16 data set names within one library.

6. Right-click the newly defined GENALIB library and select **Install**.
7. Select your single managed region in the Perform Install Operation window and click **OK** to install the library.

8. If you want to install this library at CICS TS startup, add the group GENASALI to your CICS TS startup lists defined in the GRPLIST SIT parameter.
9. Install the library to all other CICS TS regions where GENAPP runs. This includes the multitier topology.

Testing the general insurance application

To validate that your setup is correct, run GENAPP using its IBM 3270 interface. Run the test from your single managed CICS TS region, and from the presentation region of the three region topology.

The application has several transactions that you can run to access customer information and policies:

1. Run the LGSE transaction to build the control tables for the application. The transaction also clears out the shared temporary storage queue and the named counter server.
2. Run the SSC1 transaction to open the application customer menu.
3. In the Cust Number field, enter a number between 1 and 10 to browse the customer records and validate that the application can access the DB2 database. The application contains 10 customer records.
4. In the Select Option field, enter 1 to inquire on the customer. The customer information is returned from DB2.
5. Exit the application by pressing F3.
6. Run the SSC1 transaction again to add a customer to the database:
 - a. Enter the details for a new customer.
 - b. In the Select Option field, enter 2 to add the customer record to the database.

A unique customer number is allocated from the named counter server. The application adds the customer record to DB2 and the VSAM file in a two-phase commit. If the VSAM file update fails, the DB2 update is rolled back.

The application is successfully set up and running. It can access DB2 and write to VSAM files.



IBM CICS Explorer setup

This appendix provides an overview of the setup steps needed for IBM Customer Information Control System (CICS) Explorer:

- ▶ “Download and Install CICS TS” on page 220
- ▶ “Install the CICS TS resources required for CICS Explorer” on page 220
- ▶ “Define credentials to CICS Explorer” on page 220
- ▶ “Configure a CMCI connection to CICS TS from CICS Explorer” on page 221
- ▶ “Configure an FTP connection to CICS TS from CICS Explorer” on page 221
- ▶ “Change to the CICS Cloud perspective” on page 221

Overview

CICS Explorer is a system management tool available at no initial cost that offers a simple, integrated, and intuitive way of managing one or more CICS Transaction Server (CICS TS) systems. It is based on the Eclipse platform, enables you to view and manage CICS TS regions, integrates CICS TS tools, and provides the visibility and control of the CICS TS run time and its resources.

The IBM Redbooks publication *IBM CICS Explorer* describes, in detail, how to install and configure the CICS Explorer. It is available on the following website:

<http://www.redbooks.ibm.com/redpieces/abstracts/sg247778.html>

In the following sections, we discuss the key steps involved.

Download and Install CICS TS

The CICS Explorer can be downloaded by following a link from the CICS Explorer product page. The page also includes information about the CICS Explorer system requirements and supported platforms:

<http://www-03.ibm.com/software/products/en/cics-explorer/>

Follow the instructions to download and install the CICS Explorer.

Remember: The examples in this publication require CICS Explorer V5.2 or later.

Install the CICS TS resources required for CICS Explorer

The CICS cloud capabilities require IBM CICSplex System Manager (CICSplex SM). You can connect the CICS Explorer to CICSplex SM using a CICS Management Client Interface (CMCI) connection. To enable this, you need to configure your CICSplex SM Web User Interface (WUI) to support inbound CMCI connections. Details on how to do this are described in the CICS TS documentation:

http://www-01.ibm.com/support/knowledgecenter/SSGMCP_5.2.0/com.ibm.cics.ts.clientapi.doc/topics/clientapi_setupcpism.html

Define credentials to CICS Explorer

Before defining a connection from CICS Explorer to CICSplex SM, we must first create a set of credentials (user ID and password) to use for the connection. Details about how to do this are described in the CICS TS documentation:

http://www-01.ibm.com/support/knowledgecenter/SSGMCP_5.2.0/com.ibm.cics.zos.help/topics/tasks/task_configure_credentials.html

Configure a CMCI connection to CICS TS from CICS Explorer

The next step is to connect your CICS Explorer to the CMCI connection exposed through your CICSplex SM WUI. To do this, you need to know the Transmission Control Protocol/Internet Protocol (TCP/IP) port on which the WUI is listening for CMCI connections. Details about how to do this are described in the CICS TS documentation:

http://www-01.ibm.com/support/knowledgecenter/SSGMCP_5.2.0/com.ibm.cics.core.help/topics/tasks/task_configure_connection.html

Configure an FTP connection to CICS TS from CICS Explorer

Next, we create a File Transfer Protocol (FTP) connection to enable your CICS Explorer to export resources to the IBM z/OS UNIX File System. Details about how to do this are described in the CICS TS documentation:

http://www-01.ibm.com/support/knowledgecenter/SSGMCP_5.2.0/com.ibm.cics.zos.help/topics/tasks/task_configure_ftp.html

Change to the CICS Cloud perspective

Now that you have an active connection from the CICS Explorer to CICSplex SM, your setup is complete. The CICS Explorer has the concept of *perspectives*, where a perspective defines a subset of the CICS Explorer capabilities, that are applicable to a particular task. The guides in this IBM Redbooks publication use the CICS Cloud perspective. To change CICS Explorer to the CICS Cloud perspective, perform the following steps:

1. Click the menu called **Window**.
2. Select **Open Perspective → Other**.
3. Select **CICS Cloud** and click **Ok**.

The configuration of CICS Explorer is complete.

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:

- ▶ *IBM CICS Explorer*, SG24-7778
- ▶ *Event Processing with CICS*, SG24-7792
- ▶ *IBM CICS and the JVM server: Developing and Deploying Java Applications*, SG24-8038
- ▶ *Architect's Guide to IBM CICS on System z*, SG24-8067

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

ibm.com/redbooks

Other publications

CICS Transaction Server Application Architecture, TIPS0922

Online resources

These websites are also relevant as further information sources:

- ▶ CB12: General Insurance Application (GENAPP) for IBM CICS TS:
<http://www.ibm.com/support/docview.wss?uid=swg24031760>
- ▶ The National Institute of Standards and Technology (NIST) Definition of cloud computing:
<http://csrc.nist.gov/publications/nistpubs/800-145/SP800-145.pdf>
- ▶ IBM CICS Family:
<http://www.ibm.com/cics>

Help from IBM

IBM Support and downloads

ibm.com/support

IBM Global Services

ibm.com/services



Redbooks®

Cloud Enabling IBM CICS

Discover how to quickly cloud enable a traditional IBM 3270-COBOL-VSAM application

Use CICS Explorer to develop and deploy a multi-versioned application

Understand the benefits of threshold policy

This IBM Redbooks publication takes an existing IBM 3270-COBOL-VSAM application and describes how to use the features of IBM Customer Information Control System (CICS) Transaction Server (CICS TS) cloud enablement. Working with the General Insurance Application (GENAPP) as an example, this book describes the steps needed to monitor both platform and application health using the CICS Explorer CICS Cloud perspective.

It also shows you how to apply threshold policy and measure resource usage, all without source code changes to the original application. In addition, this book describes how to use multi-versioning to safely and reliably apply and back out application changes.

This Redbooks publication includes instructions about the following topics:

- ▶ How to create a CICS TS platform to manage and reflect the health of a set of CICS TS regions, and the services that they provide to applications
- ▶ How to quickly get value from CICS TS applications, by creating and deploying a CICS TS application for an existing user application
- ▶ How to protect your CICS TS platform from erroneous applications by using threshold policies
- ▶ How to deploy and run multiple versions of the same CICS TS application on the same CICS TS platform at the same time, enabling a safer migration from one application version to another, with no downtime
- ▶ How to measure application resource usage, enabling a comparison of the performance of different application versions, and chargeback based on application use

This book describes how CICS TS cloud enablement uses existing operational facilities, including monitoring, events, transaction tracking, CICS TS bundles, and IBM CICSplex System Manager (CICSplex SM), to integrate with existing deployment and management processes.

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, clients, and IBM Business 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