IBM

Flexible Decision Automation for Your zEnterprise with Business Rules and Events

Understand the benefits of operational decision management

Build dynamic solutions with business events and business rules

Learn by example with practical scenarios

Fiona Crowther
Andy Flatt
Guy Hindle
Mike Johnson
Tim Wuthenow

Redbooks

ibm.com/redbooks

International Technical Support Organization

Flexible Decision Automation for Your zEnterprise with Business Rules and Events

November 2013

SG24-8014-01

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

Second Edition (November 2013)

This edition applies to Version 8.0.1 IBM Operational Decision Manager for z/OS.

© Copyright International Business Machines Corporation 2013. 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

© Copyright IBM Corp. 2013. All rights reserved.

Figures

© Copyright IBM Corp. 2013. All rights reserved.

Tables

© Copyright IBM Corp. 2013. All rights reserved.

Examples

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:

AIX®	ILOG®	Redbooks (logo) ®
CICS®	IMS™	System z®
CICS Explorer®	RACF®	WebSphere®
CICSPlex®	Rational®	zEnterprise®
DB2®	Redbooks®	z/OS®
IBM®	Redpaper™	

The following terms are trademarks of other companies:

Linux is a trademark of Linus Torvalds in the United States, other countries, or both.

Microsoft, 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.

Preface

The IBM® Operational Decision Manager product family provides value to organizations that want to improve the responsiveness and precision of automated decisions. This decision management platform on IBM z/OS® provides comprehensive automation and governance of operational decisions that are made within mainframe applications. These decisions can be shared with other cross-platform applications, providing true enterprise decision management.

This IBM Redbooks® publication makes the case for using Operational Decision Manager for z/OS and provides an overview of its components. It is aimed at IT architects, enterprise architects, and development managers looking to build rule-based and business event-based solutions. Step-by-step guidance is provided about getting started with business rules and creating business events by using a scenario-based approach. This book provides detailed guidelines for testing and simulation and describes advanced options for decision authoring. Finally, it describes and documents multiple runtime configuration options.

Second edition: This second edition, SG24-8014-01, of this IBM Redbooks publication updated the information presented in this book to reflect function available in IBM Operational Decision Manager for z/OS Version 8.0.1. It is also important to note that the product name has changed from IBM WebSphere® Operational Decision Management for z/OS to IBM Operational Decision Manager for z/OS.

Authors

This second edition of this IBM Redbooks publication was produced by a team of specialists from around the world working at the International Technical Support Organization, Raleigh Center.

Fiona Crowther is part of the development team for IBM Operational Decision Management on z/OS in Hursley, UK. She has a Masters degree in Information Systems from the Robert Gordon University in Aberdeen, Scotland, and has worked as a software engineer in IBM for 16 years. She moved to Hursley in 2000, where she has worked on various products including IBM WebSphere Message Broker, WebSphere Enterprise Service Bus, and WebSpere Service Registry and Repository.

Andy Flatt is part of the development team for Operational Decision Management on z/OS. He has delivered key z/OS features, such as the support of multiple execution environments and rule execution from IBM CICS® applications. He is based in the IBM Hursley Development laboratory in the UK.

Guy Hindle is the team lead for Operational Decision Manager on z/OS development based in the IBM Hursley development laboratory in the UK. He has worked in IBM for 17 years and in IT for 24 years. Previous IBM roles include team lead, software development, test, and education roles on projects including application software development, middleware, and services engagements. Prior to joining IBM, he developed application generation tools for HR solutions and tools for corporate management accounting. His expertise is in usability, software application design, and database development.

Mike Johnson is a senior developer for Operational Decision Manager on z/OS in the IBM Hursley development laboratory in the UK. In this role, he also acts as an ambassador for Operational Decision Manager at various technical conferences. He has a number of patents and publications and was one of the authors of the first edition of this publication. Mike holds a Combined Honours degree in Computing and Statistics from Aston University.

Tim Wuthenow is an IT Specialist for WebSphere on IBM System z®. He holds a degree in Chemical Engineering from North Carolina State University. His area of expertise is in the configuration of WebSphere Operational Decision Management on z/OS and CICS. He has developed numerous client demonstrations using the new capabilities of WebSphere Operational Decision Management within the CICS and z/OS environments. He was also one of the authors of the first edition of this publication.

Thanks to the authors of the first edition of this book: Janet Wall, Chris Backhouse, Sebastien Brunot, Pierre Feillet, Mark Hiscock, Youngxin Pan, and Wand Wenjie. We thank them for their excellent work, much of which is still contained in this second edition.

The second edition of this IBM Redbooks project was led by:

Debbie Landon
International Technical Support Organization, Raleigh Center

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

Chris Backhouse
Operational Decision Manager on z/OS Architect

Kieron Brear
WebSphere Operational Decision Management on z/OS Development Manager

Now you can become a published author, too!

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

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

ibm.com/redbooks/residencies.html

Comments welcome

Your comments are important to us!

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

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

  ibm.com/redbooks

- Send your comments in an email to:

  redbooks@us.ibm.com

- Mail your comments to:

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

Stay connected to IBM Redbooks

- Find us on Facebook:

  http://www.facebook.com/IBMRedbooks

- Follow us on Twitter:

  http://twitter.com/ibmredbooks

- Look for us on LinkedIn:

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

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

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

- Stay current on recent Redbooks publications with RSS Feeds:

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

Part 1

zEnterprise with business rules and events

This part describes Operational Decision Manager for z/OS and contains the following chapters:

1

The case for Operational Decision Manager

This chapter introduces operational decision management and describes using IBM Operational Decision Manager for z/OS to address the agility needs of an organization’s CICS, IBM IMS™, and batch COBOL applications.

The following topics are covered in this chapter:

- 1.1, “What is Operational Decision Manager” on page 4
- 1.2, “When to think about Operational Decision Manager” on page 5
- 1.3, “Why Operational Decision Manager in z/OS applications” on page 6
- 1.4, “Where Operational Decision Manager can be used” on page 7
- 1.5, “Who is involved in deploying Operational Decision Manager” on page 8
- 1.6, “How business rule and event externalization enables application modernization” on page 9
- 1.7, “Key concepts to understand decision management” on page 10
- 1.8, “Overview of the scenario used in this book” on page 11

© Copyright IBM Corp. 2013. All rights reserved.

1.1 What is Operational Decision Manager

Smarter business outcomes require the ability to quickly adapt to change. But, corporate leadership in every industry is struggling to keep pace with change especially in this less predictable, complex economic environment. These changes directly affect the corporation's business policies and those business decisions that are required to consistently apply business policies.

Business policy: A *business policy* is a statement of guidelines governing business decisions.

For example, a bank might have a lending policy stating, "Customers whose credit rating is above average are entitled to a discounted rate on their loan". The traditional application development lifecycle requires a business analyst to document the detailed requirements to design and develop this policy into one or more business applications. Then, one or more developers take those requirements and code or embed the requirements into the various application programs. The development is then followed by a lengthy testing process. Unfortunately, the decisions are now hidden in the code in one or more programs, and over time as additional changes are added to the business policy, the code becomes more complex, making it difficult for change and traceability.

Decision management is emerging as an important capability for delivering agile business solutions. *Decision management* is the "business discipline, supported by software that enables organizations to automate, optimize and govern repeatable business decisions improving the value of customer, partner and internal interactions." Decision management is the tool to help corporations accelerate their reaction to the pace of the growing complexity of business changes.

Accurate real-time business decisions provide many benefits within an organization. Better decisions help companies identify opportunities for increased revenue and profitability, such as in marketing and sales. Better decisions also help companies enforce compliance with external and internal policies, such as in claims processing or eligibility determination. Finally, better decisions help companies manage and reduce risk, such as with fraud detection and credit approvals.

Figure 1-1 on page 5 shows the two emerging forms of decision management technology:

- Operational decision management
- Analytical decision management

The objective of this book is to describe how Operational Decision Manager capabilities (a combination of business rules and business events) can address the agility needs of the CICS, IMS, and batch COBOL applications of an organization.

For more information about operational decision management, see the IBM Redpaper™ publication *Making Better Decisions using WebSphere Operational Decision Management*, REDP-4836.

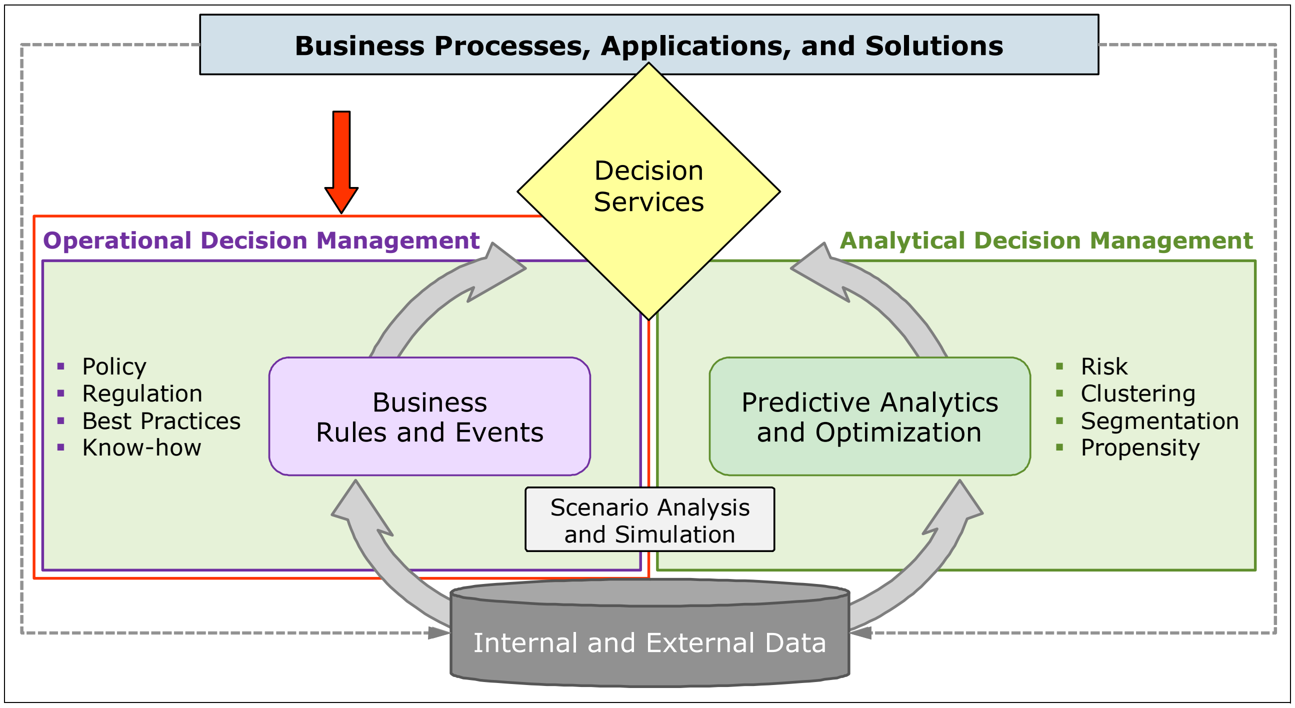

Business Processes, Applications, and Solutions
Decision Services
Operational Decision Management
Analytical Decision Management
Policy
Regulation
Best Practices
Know-how
Business Rules and Events
Predictive Analytics and Optimization
Risk
Clustering
Segmentation
Propensity
Scenario Analysis and Simulation
Internal and External Data

Figure 1-1 What is decision management

1.2 When to think about Operational Decision Manager

Organizations today have their core business processes automated in application systems that were developed over the course of years or even decades, making the applications difficult for a person to comprehend. As these software assets mature, they tend to become increasingly complex. Unfortunately, this complexity is compounded by a decline in technical and business understanding of how these assets support business goals and priorities.

Corporations need to identify solutions that support business change and cycles far more effectively than traditional application development methods and take direct advantage of business expertise. A decision management approach separates decision logic from application code so that the business logic can be modified without affecting the rest of the application. This approach enables users to assess, implement, test, and deploy changes quickly, allowing them to react to market conditions or new regulations in a timely manner.

Business decisions are made every day in business transactions incorporated in web or online business applications and batch applications. These business decisions can be categorized in three categories:

- The first type is identifying opportunities to increase revenue and profitability. This type includes decisions that are used by marketing and sales to make targeted offers based on customer profiles, demographics, and analytical models.

- The second type of automated decision concerns consistency and compliance. This type of automated decision can be found in all industries, such as financial, insurance, and government sectors. Examples are claims validation and *straight-through processing* (the ability to process a claim without manual intervention). A key goal for many organizations is not only to automate the claims process but generally to improve the quality of claims validation, while simultaneously speeding up response time, saving money, and improving customer satisfaction. The same goal is true for compliance-related decisions, such as eligibility determination or payment processing.
- The third type of automated decision includes those decisions that help reduce and mitigate risk. Examples are credit decisioning, fraud detection, and insurance underwriting.

1.3 Why Operational Decision Manager in z/OS applications

Organizations embark on application modernization projects to focus on how their core System z business applications can respond rapidly to emerging opportunities. To manage agile solution delivery, it is essential to be able to understand these business applications in terms of the business decisions they implement and the effect of decision changes on key business processes.

Organizations can rapidly and efficiently advance into their application modernization projects by incrementally externalizing their business decisions from COBOL applications and moving them into a decision management system. Most companies begin using Operational Decision Manager with one or possibly two business decisions at a time. Examples are deciding when to reorder products in a specific region and identifying the eligibility of a new customer. Taking an incremental approach with decision management in your core business applications provides organizations with a return on investment (ROI) in their first phase of their projects. An incremental approach avoids embarking on a lengthy, labor-intensive “rip and replace” project. It also enables the team to understand the design and management techniques of decision management.

Operational Decision Manager combines the authoring, testing, and management of business rules and business events that are required for implementing business decisions. Operational Decision Manager enables organizations to adapt incrementally the business decisions in their System z while avoiding lengthy application development cycles.

Operational Decision Manager offers these features:

- A set of tools for business users, administrators, and developers to edit and manage rules
- A powerful decision engine to execute business decisions
- A robust decision repository to tie everything together
- An extensive library to define and extend the decision execution and management environment

Applying Operational Decision Manager to application modernization projects can incrementally address projects in the following areas:

- Effective application maintenance: z/OS development teams need to address their long list of maintenance projects for their core COBOL business applications. If a maintenance project requires updates to the decisions that are implemented in a specific application (for example, calculating preferred customer discounts or determining credit fraud), redesign those rules in Operational Decision Manager for enhanced ongoing management.

- Consolidating or restructuring existing applications: Most organizations have duplicate functionality in multiple applications, which is a high cost whenever changes occur. Duplicated functionality causes a company to spend more time and resources maintaining applications than is necessary. Consolidation combines the same functionality into a core business application. Incorporating Operational Decision Manager technology into those modernization projects centralizes the business decisions that were redundant in the duplicated function into one source for ease of change and management.
- Sharing business decisions across applications and platforms. This area is an effective way to obtain a higher ROI. Operational Decision Manager for z/OS provides the tooling to design business decisions for your COBOL applications that can be reused or shared for execution in Java applications on z/OS or distributed platforms.

1.4 Where Operational Decision Manager can be used

Operational Decision Manager combines the management of business events and business rules and, in this combination, enables intelligent and responsive decision automation. It enables organizations to flexibly build solutions that can detect and react to event patterns as they occur within a specified time period. Then, it provides the appropriate automated decision response to transactional and process-oriented business systems.

Business event processing focuses on sense and response. Business event processing executes a continuous evaluation of events from multiple sources. It attempts to detect patterns of events that occur or do not occur as expected that represent situations to which the business wants to respond. Business event processing is primarily focused on situational awareness by detecting patterns in the overall flow of events in the enterprise.

Business rules are the specific statements that enforce a policy. The policies are translated into business rules, which are the detailed conditions and actions that unambiguously enforce the policy. The business rules expand upon the policy by stating in detail the circumstances under which the policy is applicable and the actions that enforce it. Business decisions can be defined that result in new application behaviors, offering a quick and productive route to enhance the business responsiveness of CICS and IMS operations. Also, by putting the business owner in control of the management of business decisions and at the same time retaining good governance and change management.

Designing, maintaining, testing, implementing, and managing business decisions within Operational Decision Manager provides these benefits:

- A convenient communication channel between IT and business teams
- Easy implementation and reuse of business decisions across the enterprise
- Flexible options for progressive IT modernization

Operational Decision Manager can benefit your organization in many ways:

- Customer relationship: By improving customer interaction and personalization:
  - Achieve finer-grained personalization in customer interactions. Business rules enable business users to implement more tailored promotions, pricing, risk models, and so on, therefore, increasing the precision and personalization of operational decisions.
  - Move decision-making to the point of contact with customers and enable enterprises to deploy decisions at the contact point with customers and improve consistency in decisions about customers and customer interaction.

- Enterprise processes: By improving business alignment, compliance, and transparency:
  - Achieve high pass-through rates in process automation. Centrally managing business decisions enables you to streamline processes and helps you achieve higher levels of automation and higher pass-through rates by externalizing decisions and automating more complex decisions.
  - Maximize decisions for resources, risk, and value. Managed business decisions enable businesses to tie sources of insight (from historical data, predictive knowledge, simulation, and events) and decision automation capabilities to achieve consistently better business outcomes and maximize resources and value.
- Business agility and speed: By improving business-led agility and responsiveness:
  - Empower business users to manage and improve decisions. Managed business decisions provide an agile platform to enable business users to manage decisions and changes in a short time frame.
  - Shorten response time to changing market conditions and events.
  - Increase enterprise responsiveness to unforeseen events, as well as shortened response time and time-to-business due to higher levels of automation.

1.5 Who is involved in deploying Operational Decision Manager

For a team to be effective, it is necessary to have the right set of skills on the team or available to the team for consultation. As stated in the IBM Redpaper publication *Making Better Decisions using WebSphere Operational Decision Management*, REDP-4836, the responsibility for specifying and managing the business needs to be considered from the point of view of the business roles.

Business analysts are responsible for specifying how the business needs to behave, identifying key performance indicators (KPIs) that reflect how well the business is doing, and defining the processes and decision points that are needed to manage the business.

Line-of-business (LOB) users are responsible for the day-to-day management of the business using the solutions. They are responsible for monitoring the KPIs and modifying the way that decisions are made in order to optimize the business. In a decision management solution, these roles have the responsibility for optimizing decisions to meet the business need.

Users are responsible for using the solution and need to consider the solution from a consumability and process-efficiency perspective. In many cases, the user role might be the subject of KPIs that the solution is designed to support.

The responsibility for the delivery and maintenance of these systems lies with the IT department. When embarking on a project using Operational Decision Manager technology, there are new and expanded project roles. There are several key roles to include in these projects:

- Application subject-matter expert (SME)

  A SME on the application that is being mined is an essential member of the team. This individual provides awareness of the programming styles in use and an understanding of the role that the application serves. Ideally, this SME is aware of the application's programming history, including the original purpose and design and how the application changed over time.

- Enterprise architect

  The enterprise architect can provide valuable context for the application’s role in the business decisions that are managed.
- Business rule analyst/rule designer

  This role understands the business decisions and rules for the new implementation. This person provides input to the new design of the rules. This role is normally combined with the business rule designer/implementer.
- Business object modeler

  This role defines the business object model for the target application and maps the COBOL structures to business-friendly vocabulary.
- Business rule repository administrator

  This role is responsible for ensuring that the business object model and rules are defined consistently for all phases of the project and ensures that the rules can be shared across platforms.
- Business rule miner

  This role is optional. Normally, this person is the COBOL developer or a technical business analyst responsible to “drive through the code” to identify the candidate business rules. This person enters them in the rule authoring user interface.

1.6 How business rule and event externalization enables application modernization

Companies can more rapidly and efficiently advance their application modernization projects by incrementally externalizing business rules and events from COBOL applications and moving them into a decision management system. Operational Decision Manager for z/OS can play a major role in application modernization initiatives:

- Refactor/application structure modernization. Business rules and events are externalized from core application code into a layered and modular architecture for better maintainability, reuse, and service enablement. Business rules can then be executed through a call to the appropriate Decision Server run time.
- Documentation modernization. After the business rules and events are externalized and managed using Decision Center, Decision Center continues to manage and document all the changes through its version control and audit management features. The natural-language format of the rule authoring allows the rule implementation to be the documentation. Through the Decision Center’s Rule Solution for Office component, business rules and their associated metadata can also be integrated into Microsoft Word and Excel documents.
- Reuse modernization. Business rule and event externalization helps identify the reusable components of the existing application and facilitates the reuse of business decisions that span applications. Extracting business rules from application code empowers business user subject matter experts (SMEs) to manage the rules in a manner that focuses on how decision logic is used to support overall business objectives, regardless of individual applications. This method enables a decision logic change to be implemented one time and then easily deployed by any application that requires it.

Operational Decision Manager for z/OS is designed to increase business agility by helping to revitalize existing application portfolios of an organization. By providing multiple rule and event execution capabilities, Operational Decision Manager for z/OS offers flexible options when adopting a business rules approach for application modernization on mainframe systems.

1.7 Key concepts to understand decision management

IBM Operational Decision Manager for z/OS provides comprehensive automation and governance of operational decisions that are made within mainframe applications.

To explain how Operational Decision Manager for z/OS works, this section explains its nomenclature. Every organization has people responsible for setting the policies by which they do business. In this context, a *business policy* is a statement of guidelines governing business decisions. An insurer might have an underwriting policy, for example, that states “an underage applicant for insurance on high-powered sports cars is ineligible for coverage”.

A general policy statement is not sufficient to form the basis of an automated decision. A policy manager must translate the policy into more specific statements that specify the details of how the policy is enforced. In this insurance example, the policy manager is a SME for the automobile underwriting domain.

The specific statements that enforce the policy are *business rules*. Business rules are translations of the policies into detailed conditions and actions that unambiguously enforce decision outputs. To begin, you must start with an understanding of the data, interactions, and terminology included in the specific domain of the business policy. The understanding of this information leads to the definition of a vocabulary that is used to write the rules. These rules are used by various business systems that require them through an object model, which is developed by IT and mapped to specific data sources within the software infrastructure. To the policy manager, the interaction of the rules with the object model is through the non-technical vocabulary, allowing the policy manager to author and maintain rules using a business syntax. Business rules expand on policies, by stating the detailed circumstances under which the rule is applicable and the actions that enforce it. One policy can translate into many business rules.

Another concept relating to policies is *business events*, which focus on occurrences of significance in a process or transaction that result in a change of state. Examples are, in the auto insurance scenario, the submission of an application into the underwriting system, or the change of an application from “pending” to “accepted”. Definitions describing patterns of interest across multiple business events can be defined to enforce business policies or other organizational objectives. These definitions also take the form of conditions and actions, although the conditions tend to be based on the aggregation of multiple occurrences or correlating event patterns that span a period of time. Business rules, however, tend to be based on an individual occurrence at a specific point in time.

Because both the definition of business rules and business events patterns use a common condition-action form, they are both referred to as *rules*. The terms “business rules” and “event rules” are used to distinguish between them.

Even one policy domain, such as personal auto insurance underwriting in our example, can require hundreds or thousands of rules. The rules frequently change over time and differ throughout jurisdictions, customers, products, channels, or any other partition of a business policy domain. The process by which a company manages and governs changes to policies is called the *rule lifecycle*.

1.8 Overview of the scenario used in this book

The scenario that illustrates how Operational Decision Manager can be implemented in an existing core CICS, IMS, and batch COBOL business application is based on a fictional insurance company. The insurance company needs to provide greater agility to sense, respond to, and decide when and what to do, when a customer needs to contact the insurance company. In addition, the company currently is unable to detect when the same customer or household contacts it from multiple channels of the company. This issue caused customer dissatisfaction, as well as many fraud situations of multiple requests for quotes from the same customer or household.

There are disparate business applications that manage customer channels. The call center uses a web application, and the branch offices use a CICS application to manage the insurance company's claims business processes. The insurance company initiated an application modernization project to focus on its business decisions that relate to customer contact within these channel applications.

The use of business rules ensures more consistent results and can detect fraud when a customer household submits similar quote requests. Business rules for customer validation and fraud detection need to be shared across the applications. Business rules can be used for customer profiling: identify the customer, its products with the company, and the previous quotes provided to the customer on certain products.

Business events in the CICS application and web applications can be used to identify whether there are similar requests for quotes and alert the insurance officer of possible fraud. The applications can reject a request for a quote if the number of events in a specific amount of time on the same product is reached.

The chapters in this book use this scenario and describe how to use Operational Decision Manager for z/OS for both the business rules and business events for the web and CICS COBOL applications to address the insurance company application modernization project.

Figure 1-2 depicts this scenario.

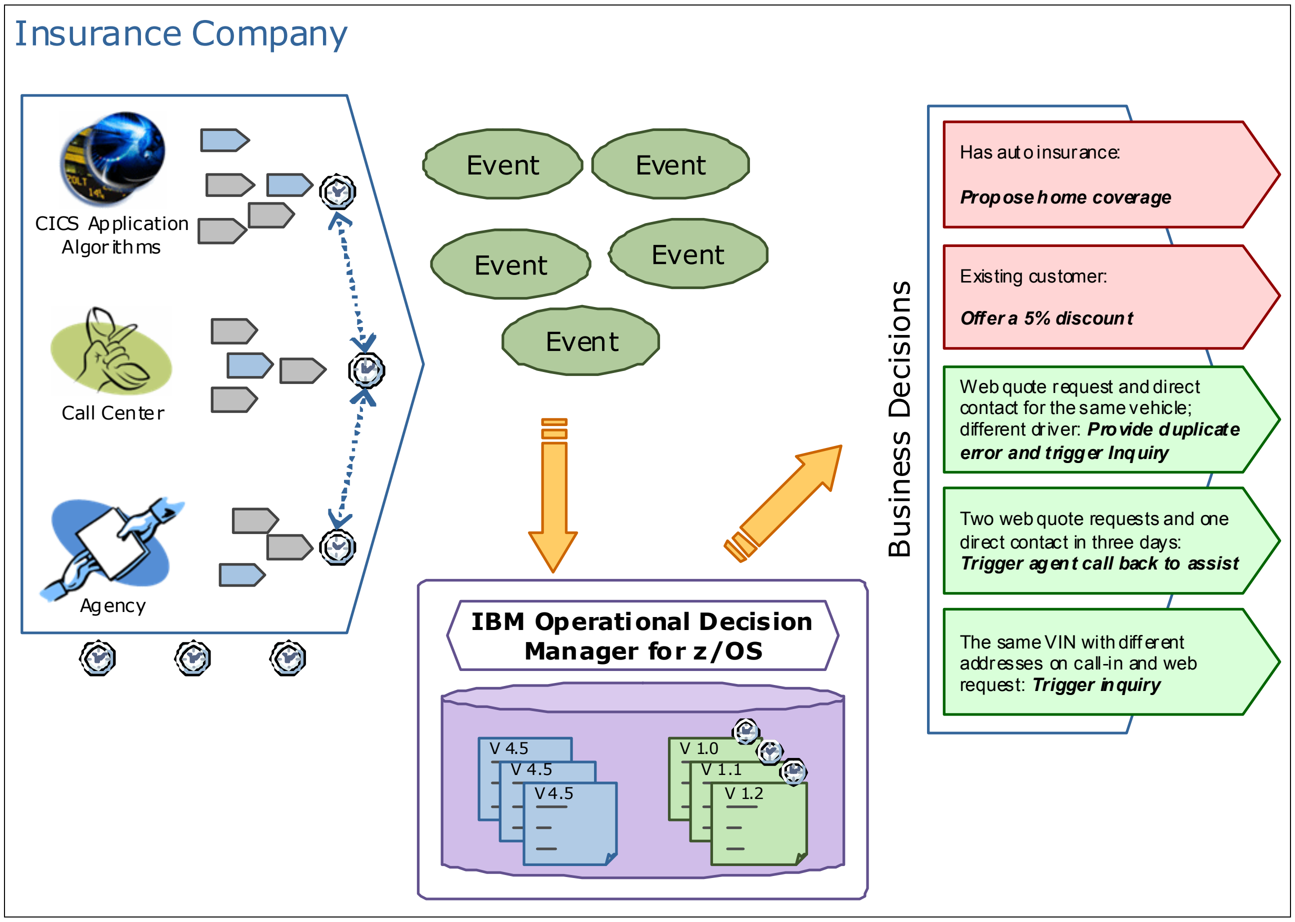

Insurance Company
CICS Application Algorithms
Call Center
Agency
Event
Event
Event
Event
Event
IBM Operational Decision Manager for z/OS
V 4.5
V 4.5
V 4.5
V 1.0
V 1.1
V 1.2
Business Decisions
Has auto insurance: Propose home coverage
Existing customer: Offer a 5% discount
Web quote request and direct contact for the same vehicle; different driver: Provide duplicate error and trigger Inquiry
Two web quote requests and one direct contact in three days: Trigger agent call back to assist
The same VIN with different addresses on call-in and web request: Trigger inquiry

Figure 1-2 Sample insurance company application modernization project

2

Operational Decision Manager on z/OS

This chapter provides an overview of IBM Operational Decision Manager on z/OS.

The following topics are covered in this chapter:

© Copyright IBM Corp. 2013. All rights reserved.

2.1 Operational Decision Manager for z/OS overview

The IBM Operational Decision Manager product family provides value to organizations that want to improve the responsiveness and precision of automated decisions on z/OS and distributed applications. On z/OS, this decision management platform provides comprehensive automation and governance of the operational decisions that are made within mainframe applications.

Operational Decision Manager for z/OS consists of two orderable products, illustrated in Figure 2-1 on page 15, which together form a platform for the management and execution of business rules and event rules:

- IBM Decision Center for z/OS provides an integrated repository and management components for line-of-business (LOB) subject matter experts (SMEs) to directly participate in the governance of business rule-based and business event-based decision logic. Through the capabilities of the Decision Center, business and IT functions can work collaboratively. They align the entire organization in the implementation of automated decisions and accelerate the maintenance lifecycle as they evolve, based on new external and internal requirements.

  Decision Center provides the following features:

  - Comprehensive decision governance, including role-based security, custom metadata, multiple branch release management, non-technical testing and simulation, and historical reporting
  - Team collaboration through multiple user access for business users and integrated synchronization between IT and business user environments

  Decision Center packaging includes these environments and tools:

  - Decision Center console
  - Decision Center repository
  - Rule Solutions for Office

  For additional information about Decision Center for z/OS, see 2.3, “Decision Center for z/OS” on page 17.

- IBM Decision Server for z/OS provides the runtime components to automate event and rule-based decision logic on mainframe systems. This product enables the detection of actionable business situations and the response of precise decisions based on the context of each interaction.

  With Decision Server for z/OS, an organization can monitor a business network to discover and act on event-based data patterns. Then, an organization can process this information against hundreds or even thousands of business rules to determine how to respond within both front-end and back-end systems.

  This product includes these components:

  - Specific run times

    These run times are designed to handle the unique aspects of business rule and business event execution. For business rule execution, this product offers several mainframe runtime options. These options allow development teams to choose a deployment strategy that best fits their mainframe applications and architecture.

  - Eclipse-based development tooling

    Rule Designer and Event Designer provide application development environments, sharing a similar high-level approach and technology.

For additional information about the Decision Server for z/OS, see 2.4, “Decision Server for z/OS” on page 19.

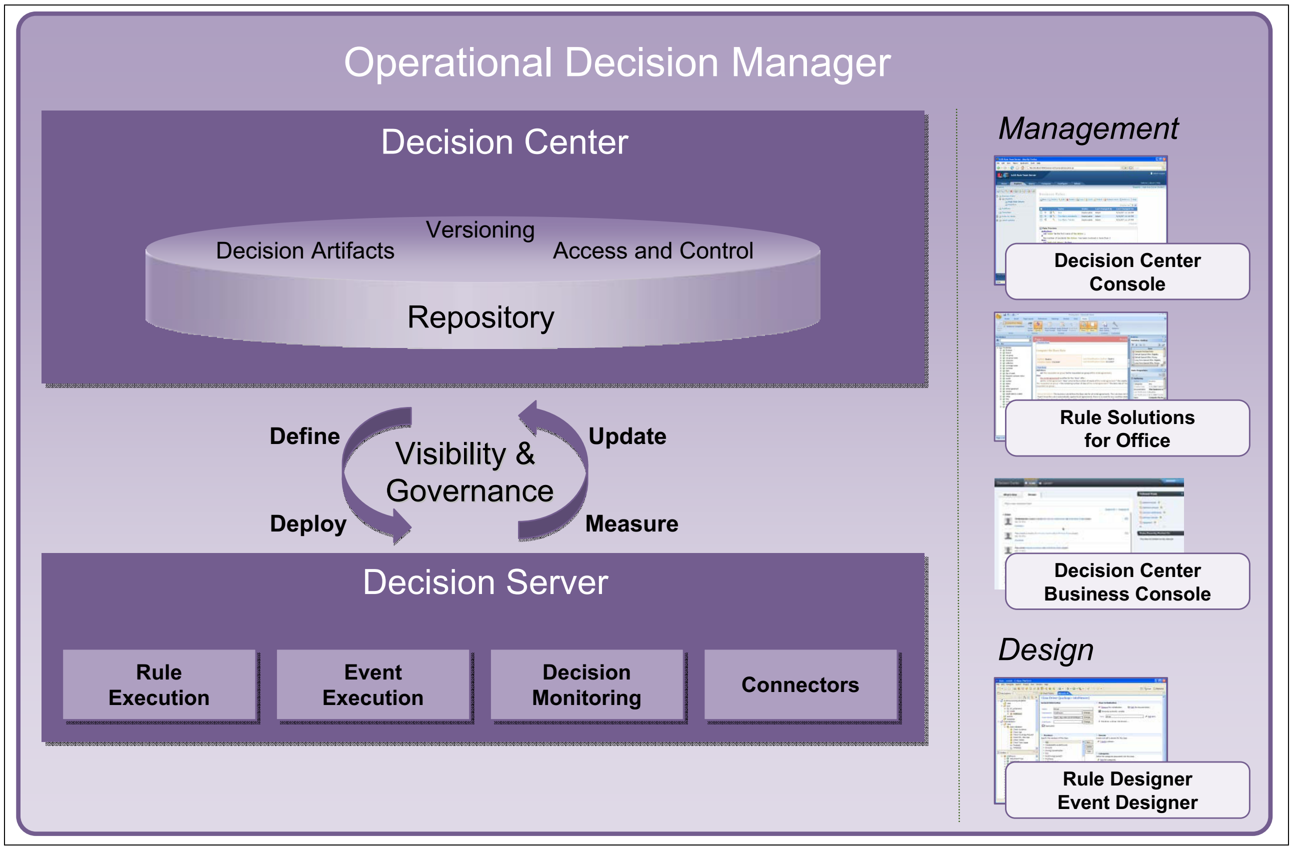


Figure 2-1 Overview of Operational Decision Manager for z/OS

2.2 Operational concepts

Operational Decision Manager for z/OS delivers the decision management features and applies them on mainframes.

Operational Decision Manager for z/OS includes these products and modules:

- Rule Designer is used as the starting point to create the model on which you author the business rules. Rule Designer is the Eclipse-based development toolkit for business rules. It is installed on a workstation.
- Event Designer is used as an entry point to develop event rules. It consists of the Eclipse-based development toolkit for event rules and is installed on a workstation.
- Decision Center is used as the team repository to govern the business rules and event rules, and to author them through a web interface. Decision Center runs on WebSphere Application Server on z/OS, Linux for System z, or a distributed environment.

- Decision Server run times are split into two types:
  - For Business rules, three approaches are possible: zRule Execution Server for z/OS, Rule Execution Server running on WebSphere Application Server on z/OS, or COBOL source generation. The core runtime stack is common across Rule Execution Server (RES) and zRule Execution Server for z/OS (zRES), which share the rule engine. zRES is a runtime solution to manage and operate decision services that are invoked from COBOL applications running in batch, CICS, or IMS on z/OS.
  - For Event rules, a run time exists on WebSphere Application Server on z/OS to process business events.

Figure 2-2 shows the relationships among the various modules and the separation between development and execution.

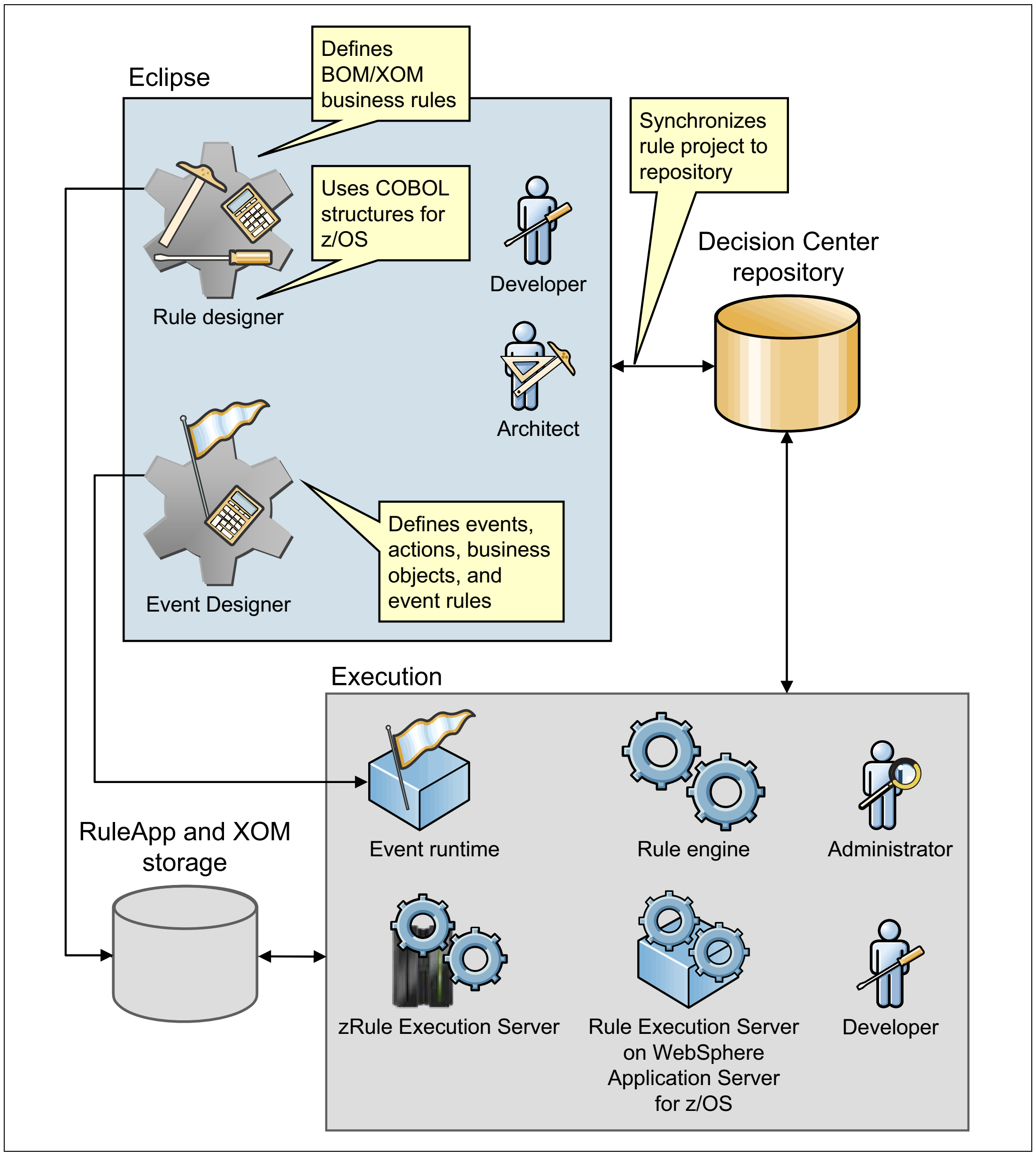

Eclipse
Defines BOM/XOM business rules
Uses COBOL structures for z/OS
Rule designer
Developer
Architect
Event Designer
Defines events, actions, business objects, and event rules
Synchronizes rule project to repository
Decision Center repository
Execution
Event runtime
Rule engine
Administrator
RuleApp and XOM storage
zRule Execution Server
Rule Execution Server on WebSphere Application Server for z/OS
Developer

Figure 2-2 Operational concepts

Table 2-1 shows the tasks across Operational Decision Manager tools and environments.

Table 2-1 Tasks across Operational Decision Manager tools and environments

Tasks	Business rules	Event rules
Synchronizing	Rule Designer	Event Designer
Authoring	Decision Center console Decision widget Rules Solution for Office	Decision Center console Decision widget
Reviewing and managing	Decision Center console Decision widget	Decision Center console Decision widget
Validating	Decision Center console	Testing widgets: ► Event Tester ► Event Capture ► Event Replay Monitoring widgets: ► Event Chart ► Event Chart Manager ► Event Layout
Deploying	Decision Center console Decision widget	Decision Center console Decision widget
Administering	Decision Center repository Decision Center console	Decision Center repository Decision Center console

2.3 Decision Center for z/OS

Decision Center for z/OS provides an integrated repository and the management components for LOB SMEs to directly participate in the governance of decision logic. Through the capabilities of Decision Center, business and IT functions can work collaboratively. Decision Center helps you to align the entire organization in the implementation of automated decisions. It helps you to accelerate the maintenance lifecycle as the automated decisions evolve based on new external and internal requirements.

2.3.1 Features

Decision Center is the central hub that coordinates the decision lifecycle across the business and IT parts of your organization. It provides the following features for business users to manage their decisions:

- Rule authoring

  Decision Center includes editors for both business rules and event rules. These editors are available in a web console. They are also available in Microsoft Office for business rules only.

- Rule synchronization between users and developers

  Synchronization is the key to collaborative work between business and IT users. You can adopt a developer-centric or a business user-centric approach to managing synchronization.

- Rule review and management

  In the Decision Center console and in the Decision widget, business users can run queries and publish reports on the content of their projects. Decision Center provides ways to customize how business users can view the items in their projects with smart folders. Business users can also manage releases and work in progress with branches and baselines.
- Rule validation

  Decision Center provides tools for validating that decisions are implemented as expected. For business rules, Decision Center provides testing and simulation of rulesets. For event rules, the Testing and Monitoring widgets help business users analyze the processing of business events.
- Rule deployment

  Following verification, you can deploy your decision logic as business rules or event rules to the production system.
- Administration

  After you configure Decision Center, you perform several regular administrative tasks to provide optimum service to the business users.

Decision Center can reside on z/OS or Linux for System z. It can be deployed, as well, in a distributed environment to edit the business rules and event rules. It can be connected to a Decision Server running on z/OS.

2.3.2 Directory structure

The following top-level directories are created under the `<`*`InstallDir`*`>` directory for each component in Decision Center for z/OS. These components correspond to modules or other services that provide related functionality or data:

- `events`: This directory contains the necessary resources to configure the Event widgets:
  - `config`
  - `widgets`
- `teamserver`: This directory contains the necessary resources to deploy and configure Decision Center on the WebSphere Application Server on z/OS:
  - `applicationservers`
  - `bin`
  - `lib`
- `shared`: This directory contains the shared third-party tools.

2.3.3 Decision Center console

Decision Center console is a web application that is shared across business rules and business events. This web application empowers business users to author and manage their rules for business rules and events. All decision authoring assets are versioned and persisted in an underlying database repository (Figure 2-3 on page 19).

Authoring artifacts: The authoring artifacts are transformed into executable assets and then deployed to a runtime repository. The console can be deployed on z/OS and run on WebSphere Application Server on z/OS. The console deployment is also allowed on Linux for System z and distributed environments.

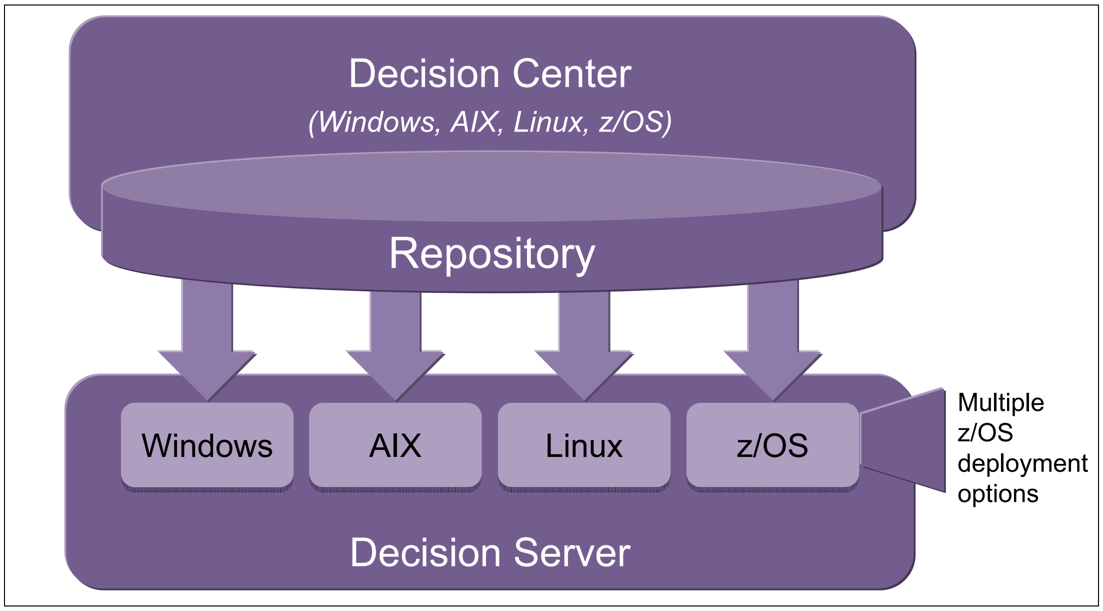


Figure 2-3 Operational Decision Manager on z/OS environment options

2.3.4 Rule Solutions for Office

Rule Solutions for Office is used for sharing business rules offline. Policy managers and rule authors can author business rules in Rule Solutions for Office. With Rule Solutions for Office, rule authors write rules in Microsoft Word and edit decision tables in Microsoft Excel. They can create mixed rule and non-rule content in a *RuleDoc*, which retains semantic information together with the actual implementation content of the rules.

You can integrate business rule authoring and management extensions that are developed in Rule Designer into Rule Solutions for Office. The RuleDocs can be synchronized with the Decision Center console.

2.4 Decision Server for z/OS

Decision Server for z/OS provides the runtime components to automate decision logic. These components enable the detection of actionable business situations and the response of precise decisions based on the specific context of an interaction. With Decision Server, organizations can monitor a business network to discover and act on event-based data patterns. Then, organizations process this information against business rules to determine how to respond within both front-end and back-end systems.

Decision Server is split into the following independent components, as illustrated in Figure 2-4 on page 20:

- Decision Server Rules:
  - Rule Designer, which is an Eclipse-based integrated development environment (IDE) for business rule development
  - Rule Execution Server on WebSphere Application Server on z/OS
  - zRule Execution Server (zRES)

- Decision Server Events:
  - Event Designer, which is an Eclipse-based IDE for business event development
  - Event Execution Runtime, which runs on WebSphere Application Server on z/OS

Decision Server Rules and Events have their own dedicated stacks and executable artifact repositories.

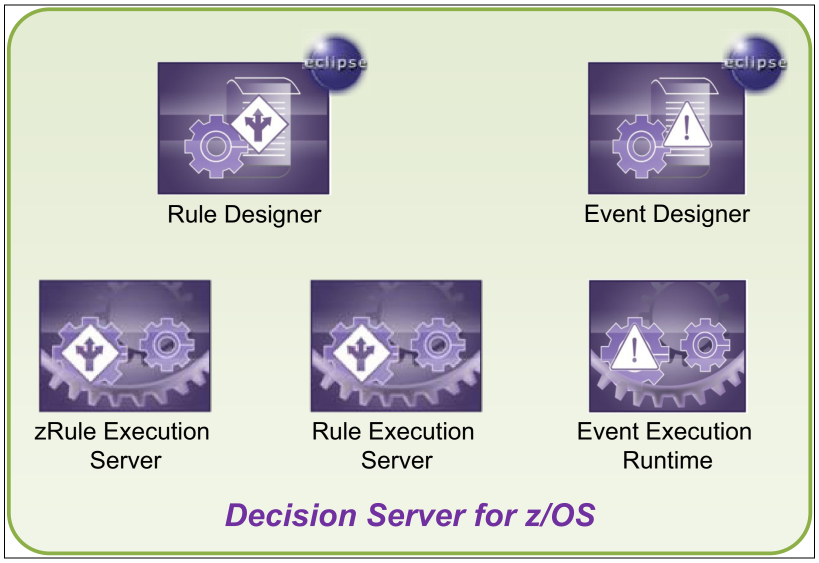


Figure 2-4 Decision Server for z/OS components

2.4.1 Directory structure

The following top-level directories are created under the `<InstallDir>` directory for each component in Decision Server for z/OS. These components correspond to modules or other services that provide related functionality or data:

- `events`: This directory contains the necessary resources to deploy and configure the event run time.
- `executionserver`: This directory contains the necessary resources to deploy and configure Rule Execution Server on the WebSphere Application Server on z/OS:
  - `applicationservers`:
    - `WebSphere7`
    - `WebSphere8`
    - `WebSphere85`
    - `WOLA`
  - `bin`
  - `lib`
- `shared`
- `zexecutionserver`: This directory contains the necessary libraries and files to configure a zRES for a z/OS instance.

The following set of Decision Server for z/OS data sets is added to your system during installation:

- HBRHLQ.SHBRAUTH: Authorized program facility (APF)-authorized modules
- HBRHLQ.SHBRCICS: CICS modules

- HBRHLQ.SHBRCOBC: COBOL copybooks
- HBRHLQ.SHBRCOBS: COBOL sample code
- HBRHLQ.SHBREXEC: Installation-specific exec
- HBRHLQ.SHBRINST: Install and configuration jobs
- HBRHLQ.SHBRJCL: Runtime sample jobs
- HBRHLQ.SHBRLOAD: Product modules
- HBRHLQ.SHBRPARM: Runtime configuration parameters
- HBRHLQ.SHBRPROC: Runtime JCL procedures
- HBRHLQ.SHBRWASC: Generated properties file for WebSphere Application Server configuration

2.4.2 Features

Depending on their role (architect, COBOL developer, QA tester, business user, policy manager, and so on), users are interested in various aspects of developing a business rule application and integrating it with the calling application on z/OS. Decision Server for z/OS offers these features:

- Business rule application development for z/OS

  To develop a business rule application for z/OS, you design business rules independently from the application logic. Using Rule Designer, you develop rule projects from which you extract a ruleset. Then, you create a contract between the application and the ruleset. An application can call the ruleset in a number of ways using various execution options.

- Validation of ruleset execution on z/OS

  With Rule Execution Server on WebSphere Application Server on z/OS, you can test the ruleset execution and simulate scenarios.

- Integrating ruleset execution into z/OS

  You have a number of options to execute a ruleset on z/OS. When you need a managed execution environment on z/OS, you must choose between WebSphere Application Server or zRES. In both cases, you can audit and monitor the performance of the application using the administrative console. In a Java Platform, Enterprise Edition (Java EE) environment, you can also use Decision Warehouse.

Decision Server on z/OS offers multiple deployment options with and without a WebSphere Application Server base.

See 2.4.3, “Decision Server rules” on page 21, which focuses on the business rule aspects of Decision Server.

2.4.3 Decision Server rules

This section describes Decision Server rules.

Rule Designer

Rule Designer is used for the base rule authoring. This tooling is Eclipse-based and installed on a workstation. It cooperates with the Decision Center, which is installed on a distributed operating system or on WebSphere Application Server installed on z/OS.

This split-platform configuration enables users to implement full business rule management system (BRMS) functionality with the ability to define a business object model from COBOL definitions, run and test rules on COBOL data, and call the rule engine from existing CICS, IMS, and batch applications. Users who want to eliminate the duplication of application functionality on z/OS and distributed applications can consolidate applications and identify the rules that they can share between the two platforms.

The inclusion of COBOL management in Rule Designer enables users to author rules and develop object models that can be shared with COBOL and Java applications. Developers import a COBOL eXecutable Object Model from a copybook and create a Java eXecutable Object Model along with a marshaller project to provide the mapping between Java and COBOL.

You do not have to re-engineer or rewrite an entire COBOL application to start managing the business rules for your z/OS applications. You can base the scope of your rules on one, or a combination, of the following objects:

- A set of rules for a specific region, territory, or type of customer
- A process or subprocess within a z/OS application
- As a replacement for rules that might be hardcoded in your COBOL application

A rule authoring environment is set up in the Rule Designer. Rule projects are created in the Rule perspective.

This Rule Project can be synchronized with Decision Center and Rules Solution for Office to finally deploy executable rules with separate approaches.

Methods to invoke business rules from COBOL

When it comes to deployment, Operational Decision Manager for z/OS does not impose a rigid architectural or technical choice. Instead, it provides a set of options of which you can take advantage, depending on your strategy and architectural preferences.

This solution provides three options on Decision Server for z/OS for deploying business rules on System z:

- Rule Execution Server on WebSphere Application Server for z/OS

  This option brings the full power of WebSphere Application Server for z/OS to the rule execution on z/OS. Full high availability and scalability are provided by the underlying application server, and the full suite of decision management services is available. Because of the ability to consume COBOL data structures directly, the Rule Execution Server environment can be easily integrated with existing COBOL applications. This integration occurs through the use of technologies, such as the WebSphere Optimized Local Adapter or WebSphere MQ.

  For details about the decision engine for zRES, see 2.5, “New in Operational Decision Manager Version 8” on page 27.

- zRule Execution Server for z/OS (zRES)

  This option provides more local integration with existing COBOL applications. A supplied COBOL stub program provides an interface that can call COBOL directly into the rule execution. This rule execution environment provides COBOL applications with access to the full rule-authoring constructs through three available runtime options:

  – Stand-alone mode provides a rule execution address space that can be invoked from existing COBOL applications by way of the supplied callable stub.

  - In IBM CICS Transaction Server for z/OS V4.1 and V4.2, an optimized local execution option uses the new Java virtual machine (JVM) server environment to allow execution within the CICS region.
  - An option of a COBOL application with the HBR API and zRule Execution Server Runtime (RT) and the zRES Console.

- COBOL code generation

  This option is for clients who want to retain their existing application architecture and manage their business decisions within COBOL application code. However, they want a cleaner and more manageable form than through standard application development methods. The rules in a COBOL application can be incrementally migrated to a central business rule repository for external management, directly by business users. Then, the rules can be generated back into COBOL code to be inserted into and called directly from the application. This option can also be a first step toward the incremental modernization of applications.

All of the previous options provide the following values:

- Reduced risk, disruption, cost, and time to implement change
- Better visibility and maintainability of decision logic
- Improved decision logic reuse across applications

Decision Service is deployed as COBOL source code. It offers a pure COBOL approach but without giving full agility, such as the Rule Execution Server on WebSphere Application Server for z/OS or zRES options. The difference is the compiled and linked code. Additionally, this source generation option does not offer support for Decision Validation Service (DVS) and is limited to the Sequential algorithm. See Figure 2-5.

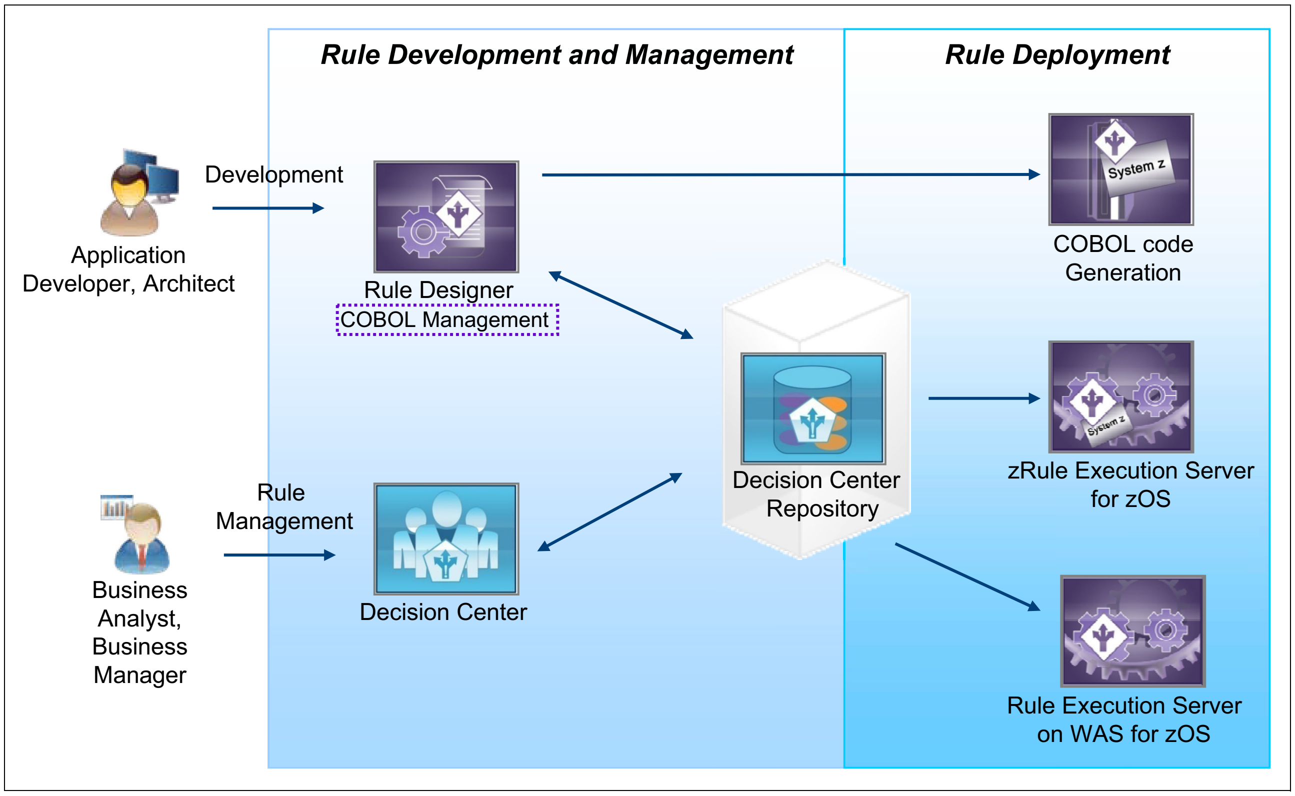

Rule Development and Management
Rule Deployment
Development
Application Developer, Architect
Rule Designer
COBOL Management
System z
COBOL code Generation
Decision Center Repository
zRule Execution Server for zOS
Rule Management
Business Analyst, Business Manager
Decision Center
Rule Execution Server on WAS for zOS

Figure 2-5 Operational Decision Manager for z/OS business rules options

A feature comparison between zRES and the Rule Execution Server on WebSphere Application Server for z/OS is shown in Table 2-2.

Table 2-2 zRule Execution Server for z/OS feature comparison

Feature	**zRule Execution Server for z/OS (zRES)**	**Rule Execution Server on WebSphere Application Server for z/OS (RES)**	**COBOL source generation**
Execution from Java	No	Yes	No
Execution from COBOL	Yes	Yes, through WebSphere Optimized Local Adapter	Yes
OOTB COBOL marshalling	Yes	No	N/A
Testing	No	Yes	No
Simulation	No	Yes	No
Hosted transparent decision services	No	Yes	No

Figure 2-6 shows the run times to invoke business rules from a COBOL application.

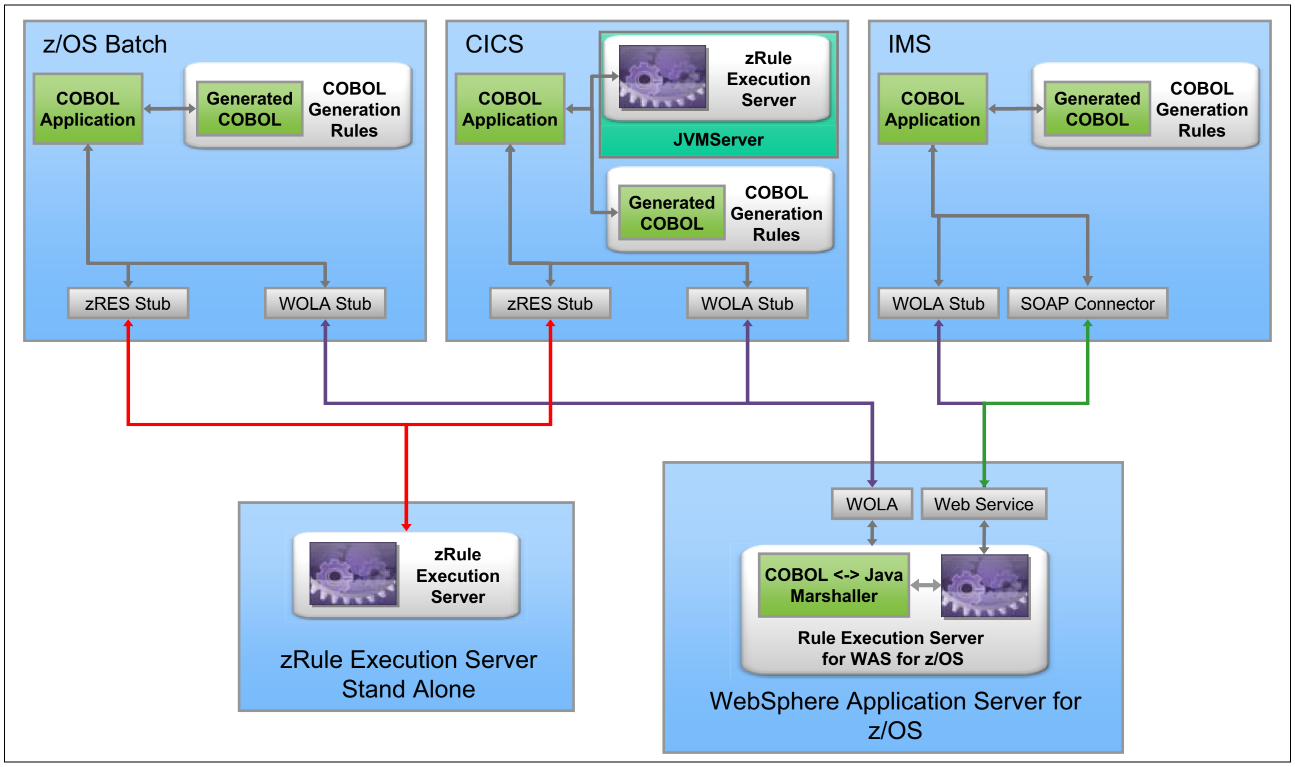


Figure 2-6 Runtime options to invoke business rules from a COBOL application

2.4.4 Decision Server events

This section describes Decision Server events.

Event Designer

Event Designer is a Decision Server Events component that supports the definition of the metadata layer that is required for business event processing (BEP). You can use Event Designer to create all the building blocks for your application, including events, business objects, actions, and event rules. Event Designer is based on Eclipse.

Event Designer brings an Eclipse event perspective, in which you manage event projects.

Event run time for WebSphere Application Server on z/OS

The event run time consists of a number of components that manage the real-time business event coordination that was defined during application development. The event run time consists of the following components (Figure 2-7 on page 26):

- The Decision Server Events application for BEP, which is called *wberuntimeear*
- A Java Message Service (JMS) message queue that is managed by a message queue server, such as the WebSphere Application Server Network Deployment service integration bus or WebSphere MQ
- A database manager to serve as a message store for persistent messages
- WebSphere Application Server Network Deployment
- A relational database to contain the event run time
- Event connectors and action connectors for touchpoint systems
- Appropriate Java Database Connectivity (JDBC) drivers for the event runtime database manager and any databases accessed in a Decision Server Events application

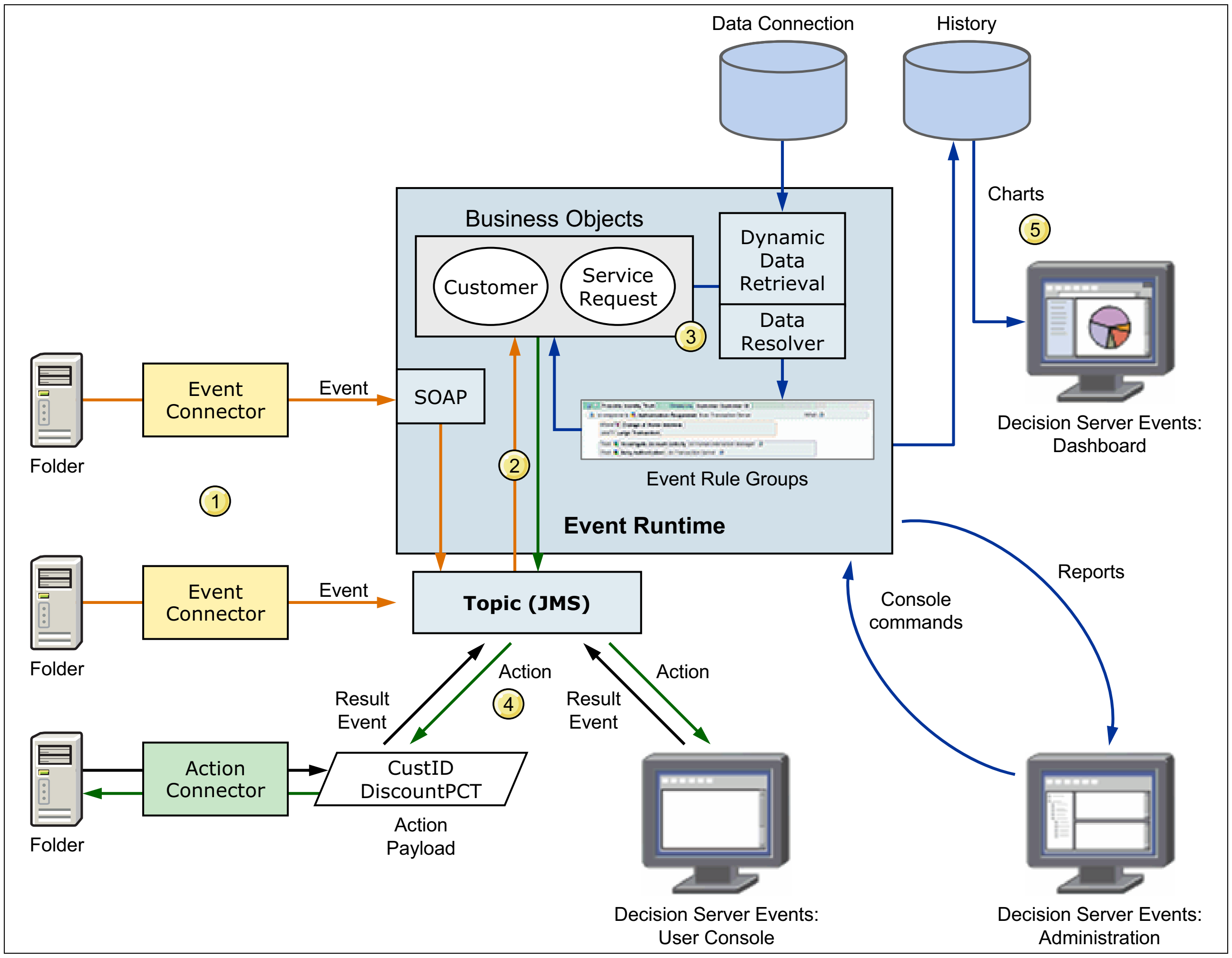


Figure 2-7 Operational Decision Manager Decision Server event

The event run time manages the real-time business event coordination that was defined during application development (the following numbers correspond to the numbers that are shown in Figure 2-7):

1. When an event occurs in a touchpoint system that potentially requires one or more actions in other touchpoint systems, the relevant data (field name, field type, and value), which is called an *event payload*, is passed to the JMS topic by using the connector of the touchpoint system. Web services environments that employ SOAP to package messages and other protocols, such as HTTP, can direct those messages through the JMS queue.
2. The event run time retrieves the message from the JMS queue and populates the appropriate business objects with the values that are contained in the event payload.
3. The event run time identifies the event rules that reference the event and determines whether any filters exist that require further evaluation.

   If a rule includes a condition, the event run time evaluates that condition to determine whether the conditions for an action are met. For complex event processing, this evaluation includes determining whether referenced events or actions occurred as described in the conditions. If any values are missing, the event run time attempts to retrieve the missing information from an external data connection. The action is triggered only if the condition is true.

   If the rule does not include any conditions, the specified action is triggered.

4. The event run time passes the relevant data (field name, field type, and value) associated with the action from business objects as an *action payload* to the outbound message queue, where it is picked up by the appropriate connector. The connector pushes the data to the appropriate touchpoint systems, initiating the appropriate activity. The connector might return a result back to the JMS queue, where it is retrieved by the event run time as a new event and processed appropriately.

   If the action requires human intervention, it is directed to the user console, where you can access it. When you respond to the information that is displayed, the response is sent back to the JMS queue as a result event, where it is retrieved by the event run time as a new event and processed appropriately.

5. The history for events, actions, and filters that are used in event rule group evaluation is stored in a History database manager.

2.5 New in Operational Decision Manager Version 8

Several new features are available in Operational Decision Manager for z/OS V8:

- 2.5.1, “High performance engine for Rule Execution Server for z/OS” on page 28
- 2.5.2, “Decision Center Business console” on page 31
- 2.5.3, “Testing and simulation support for rule-based decisions on z/OS” on page 31
- 2.5.4, “Usability improvements for COBOL management” on page 31
- 2.5.5, “Scenario Service Provider (SSP) support on zRule Execution Server” on page 31
- 2.5.6, “Revised IMS support” on page 32
- 2.5.7, “WebSphere Optimized Local Adapters” on page 32
- 2.5.8, “Decision Warehouse” on page 32

Note: The first edition of this IBM Redbooks publication, *Flexible Decision Automation for Your zEnterprise with Business Rules and Events,* SG24-8014-00, was based on Version 7, Release 5 of WebSphere Operational Decision Management for z/OS. This second edition includes updates for Version 8.

For more details, see the following links:

- For Operational Decision Manager V8.0:
  - What’s new in Decision Center for z/OS

    http://pic.dhe.ibm.com/infocenter/dmanager/v8r0/topic/com.ibm.wodm.zos.dserver.overview/topics/tpc_dsz_whatsnew.html
  - What’s new in Decision Server for z/OS

    http://pic.dhe.ibm.com/infocenter/dmanager/v8r0/topic/com.ibm.wodm.zos.dcenter.overview/topics/tpc_dcz_whatsnew.html
  - What's new in Operational Decision Manager V8.0.1

    http://pic.dhe.ibm.com/infocenter/dmanager/v8r0m1/index.jsp?topic=%2Fcom.ibm.wodm.family.overview%2Ftopics%2Fcon_whats_new.html
- The following announcement letters also contain information about new features:
  - IBM Operational Decision Manager V8.0.1 enhances the user experience for managing change and the implementation of decision logic that drives critical business systems.

    http://www-01.ibm.com/common/ssi/cgi-bin/ssialias?infotype=AN&subtype=CA&htmlfid=897/ENUS212-326&appname=USN#h2-descx

- IBM WebSphere Operational Decision Management for z/OS V8.0 provides end-to-end, rule, and lifecycle support for decisions that are used in CICS, IMS, and COBOL applications.

  http://www-01.ibm.com/common/ssi/cgi-bin/ssialias?infotype=AN&subtype=CA&htmlfid=897/ENUS212-101&appname=USN

2.5.1 High performance engine for Rule Execution Server for z/OS

A new high performance engine, which is called *Decision Engine,* is now available to execute rules on zRule Execution Server for z/OS (zRES).

Decision Engine for zRES is a new implementation of the rule engine that improves the overall performance of rule execution. The Decision Engine compiles rule artifacts into an archive that contains code that is ready to execute. Ruleset loading in the new engine is faster than the previous engine and there is no ruleset parsing at run time and no interpreted code.

What is a rules engine

A *rules engine* hosts the business rules that pertain to a given decision. These rules are organized for execution and stored in a *ruleset*. Rulesets are executable containers that correspond to a decision. Rules are atomic expressions of policies. The automation of these policies and their application to events can generate decisions.

After you develop your business rules and group them into rulesets, these rulesets are then deployed to the rules engine. The rules engine hosts these business rules.

An application wanting to ask for a decision from the rules engine makes a connection to the engine, which runs the request and returns the decision. The rule engine reads rules from a ruleset archive, evaluates rule conditions against application objects, and executes the rules for the objects that meet the conditions. You can maintain the rule engine independently from the business applications.

The rules engine has the following functions:

- Read rules dynamically at run time
- Evaluate the rules against the application objects
- Keep track of changes to application objects

The are two types of rules engine:

- Classic rule engine
- Decision engine for zRule Execution Server of z/OS

Decision Engine for zRES

Decision Engine optimizes the execution performance of your copybook-based ruleset. The Decision Engine works in a similar way to the classic rule engine, although there are differences in the compilation and loading of the rules. This gives Decision Engine better performance results but it means that support for some of the features of the classic engine is unavailable.

Table 2-3 on page 29 shows the main differences between the two rules engine technologies.

Table 2-3 Differences between the classic rule engine and the Decision Engine

Classic rule engine	Decision engine
The classic rule engine processes the rule artifacts in different ways: ▸ The ruleflow and the RetePlus rule actions are interpreted. ▸ The RetePlus rule conditions that use the useJit property are compiled to bytecode. ▸ The rules that use the sequential and Fastpath algorithms are compiled to bytecode.	The rules are compiled to bytecode before deployment, which reduces the loading time of the ruleset.
The loading into the rule engine includes the parsing of the rules, the compilation, a partial generation of the bytecode, and the loading of the Java classes.	The loading into the rule engine only involves the loading of Java classes because the rules are already compiled to bytecode.

Limitations over the Classic engine

The product documentation details the limitations in the Decision Engine:

`http://pic.dhe.ibm.com/infocenter/dmanager/v8r0m1/index.jsp?topic=%2Fcom.ibm.wodm.dserver.rules.designer.run%2Fexecuting_decision_topics%2Fcon_decision_limitations.html`

Enabling the ruleset to use the Decision Engine

To enable the Decision Engine, the “build mode” of the project needs to be changed to execute the rules with the Decision Engine.

The Decision Engine compiles rule artifacts into an archive that contains executable code. There is no ruleset parsing at run time and no interpreted code. The archive already contains all the Java classes. Therefore, the loading of the ruleset in the engine involves class loading only. The compiled rules are quickly created in memory from their binary representation, the business object model (BOM)-to-execution object module (XOM) mapping is applied, and the engine instances can be created.

The process of compilation and execution for the decision engine goes through different stages from the initial compilation of rules until the execution of the engine as shown in Figure 2-8 on page 30.

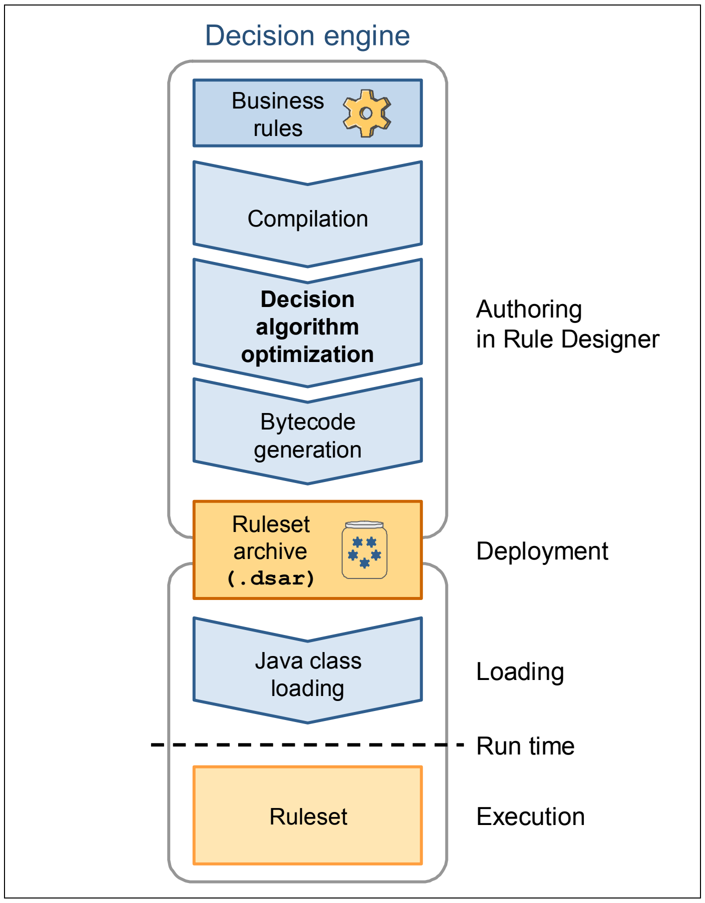
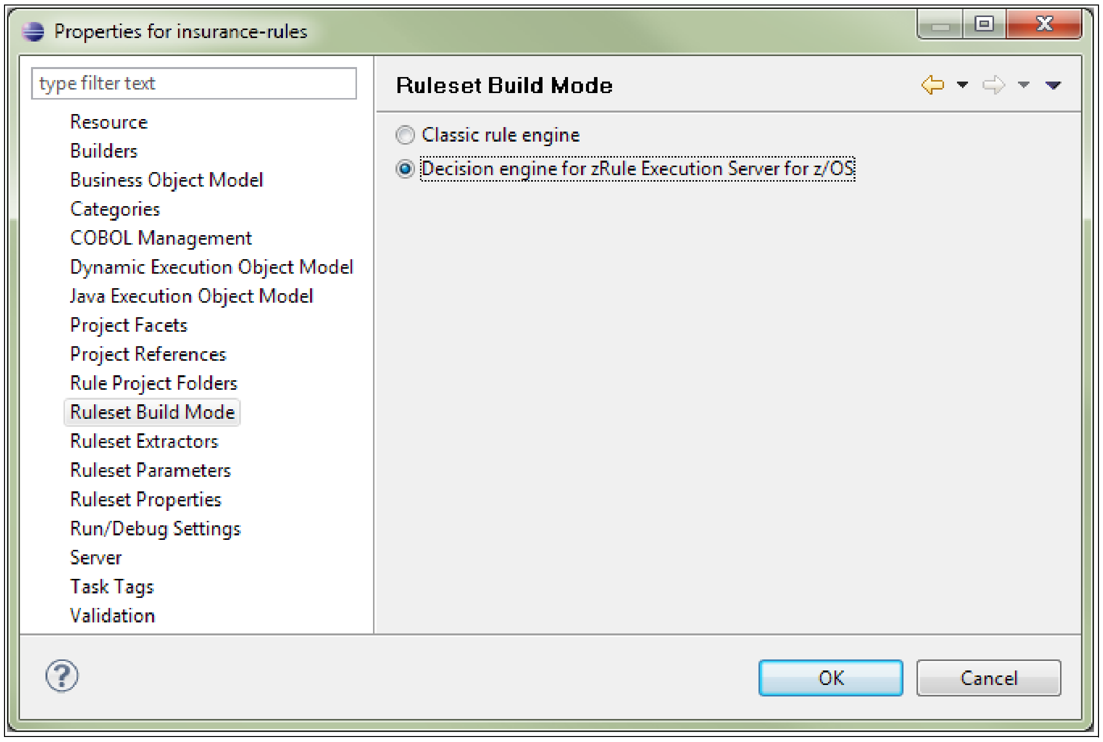


Figure 2-8 Decision engine for zRule Execution Server for z/OS

The build mode of the project can be changed in the properties of the rule project by performing the following steps:

1. Right-click the rule project and select **Properties**.
2. In the Properties dialog, click **Ruleset Build Mode**, and select one of the following rule engines to execute your rules:
   - **Classic rule engine**: The default rule engine to execute rules on both distributed and z/OS platforms.
   - **Decision engine for zRule Execution Server for z/OS**: A high performance rule engine to execute rules on zRule Execution Server for z/OS.

   See Figure 2-9 on page 31.


Properties for insurance-rules

type filter text

- Resource
- Builders
- Business Object Model
- Categories
- COBOL Management
- Dynamic Execution Object Model
- Java Execution Object Model
- Project Facets
- Project References
- Rule Project Folders
- Ruleset Build Mode
- Ruleset Extractors
- Ruleset Parameters
- Ruleset Properties
- Run/Debug Settings
- Server
- Task Tags
- Validation

Ruleset Build Mode

- ○ Classic rule engine
- ● Decision engine for zRule Execution Server for z/OS

Figure 2-9   Setting the Decision Engine for zRule Execution Server of z/OS

2.5.2 Decision Center Business console

The Decision Center Business console is targeted at non-advanced business users. This new console contains a subset of the features available in the previous console, which is now referred to as the *Decision Center Enterprise console*. The new Decision Center Business console also includes new features, such as streams and posts, that let you see and comment on activities with other users.

2.5.3 Testing and simulation support for rule-based decisions on z/OS

A new sample has been added that covers the use of a Scenario Service Provider (SSP) on zRES. This demonstrates testing data from a VSAM file.

2.5.4 Usability improvements for COBOL management

The marshaller project no longer appears as a separate project in Rule Designer. It is now possible to select multiple copybooks by using the copybook importer.

It is also possible to import copybooks and generate COBOL programs to a remote system resource when Rule Designer is integrated with IBM Rational® Developer for System z.

2.5.5 Scenario Service Provider (SSP) support on zRule Execution Server

You can now test and simulate a rule project by using COBOL data.

2.5.6 Revised IMS support

Support for IMS to call into Decision Server for z/OS has been added:

- Bean-managed persistence (BMP) and Data Language/I (DLI) programs can call into the Decision Server for z/OS to execute rules. This support was added in Operational Decision Manager V8.0.
- Applications that run in a message processing region (MPR) can call Decision Server for z/OS to execute rules. This support was added in Operational Decision Manager V8.0.1.

Note: IMS version 10 is not supported after Decision Server V8.0.

For more information about using IMS with Operational Decision Manager, see Chapter 12, “Configuring IMS to work with Operational Decision Manager” on page 221.

2.5.7 WebSphere Optimized Local Adapters

WebSphere Application Server can now be accessed using WebSphere Optimized Local Adapters (WOLA), improving the performance when using WebSphere Application Server. For more information about WOLA, see Chapter 13, “Configuring WebSphere Optimized Local Adapters support” on page 225.

2.5.8 Decision Warehouse

Decision Warehouse is a tool of the Rule Execution Server console for monitoring ruleset execution. It stores execution traces in a database.

The execution trace contains information about how a decision was made. It records the executed ruleflow, the path of executed ruleflow tasks, and the rules executed. These details are intended to help users, such as an auditor, understand what happened as a result of executing a ruleset. Users can access Decision Warehouse in the Rule Execution Server console.

For more information about Decision Warehouse, see Chapter 14, “Configuring decision warehousing” on page 241.

Getting started with business rules

This chapter describes and demonstrates the use of business rules on System z.

The following topics are covered in this chapter:

© Copyright IBM Corp. 2013. All rights reserved.

3.1 Overview of the example used in this chapter

This section provides an overview of the business scenario and the related models that are used in this chapter.

3.1.1 Business scenario

The business scenario that is used in this book describes a fictitious auto insurance company that has a solution in place for validating an insurance application for its customers. The company wants to manage and share this validation logic with other business applications on System z.

The status of the in-place application and the way in which a company wants the business rules to be executed on System z determine the approach that is chosen. This chapter uses the following typical approaches:

- Getting started from a COBOL copybook (see 3.2, “Getting started from a COBOL copybook” on page 35)
- Getting started from an existing Java-based rules project (see 3.3, “Getting started from an existing rule project” on page 73)

3.1.2 Business model

The business model used in this chapter represents an insurance quote application. This model is simplified as one insurance quote request and one insurance quote response.

Insurance quote request

The insurance quote request includes the following information:

- Driver

  This information includes personal information about the driver to assess the risk, such as age, address, and license status.
- Vehicle

  This information includes the make, model, year, and vehicle identification number, together with a categorization of the vehicle type and a vehicle value. The insurance discount policies differ based on the vehicle type.

Insurance quote response

The response to an insurance request includes the following information:

- The validation status of this insurance quote
- The validation message of this insurance quote
- The pricing and discount information of this insurance quote

3.1.3 Scenario rule model

The rules are designed to validate a customer’s eligibility for the quote application. These sets of rules validate the customer’s age or accident history and provide the validation result and possible reasons.

Both scenarios use the following rules:

- A maximum or minimum age rule, called *MaxiMinimumAge,* validates that the age of customer is between lower and upper age limits.
- A number of accidents rule, called *NumberOfAccidents*, validates that the number of accidents of the customer is below the set upper limits.

3.1.4 Project structure of a business rule on z/OS

This section illustrates a common project structure for business rule execution on zRule Execution Server for z/OS (zRES). The following typical artifacts are used in zRule Execution Server for z/OS projects:

- The *rule project*, which is used to design, debug, and manage the business rule
- The *Java Execution Module (Java XOM)*, which is used to create a rule project and which is deployed as a Java archive (JAR) resource for rule execution at run time
- The *marshaller*, which handles the conversion between COBOL data items and Java data items at run time

  Important: The marshaller is a generated artifact. Never modify it yourself.

- The *RuleApp project*, which is used to deploy business rules to the runtime configuration of the zRule Execution Server

3.2 Getting started from a COBOL copybook

This section includes detailed instructions about creating a rule application on z/OS that is started from a COBOL copybook.

3.2.1 Scenario overview

In this scenario, an insurance company has a large COBOL application that runs on z/OS. This COBOL application validates insurance applications. The company wants to manage the logic codes that are scattered throughout the COBOL application and share them with other business applications on System z. The company decides to migrate and manage the business logic as a business rule application on System z.

In an actual scenario, you identify or create a COBOL copybook that contains the COBOL data items that are required for the business rules. Example 3-1 on page 36 shows a sample COBOL copybook (INSDEMO), which is designed for the first rule application on System z in this scenario.

Example 3-1 Sample COBOL copybook [INSDEMO.cpy]

```
01  REQUEST.
      05  DREIVER.
    10  FIRST-NAME            PIC  X(20).
    10  LAST-NAME             PIC  X(20).
        10  ZIPCODE               PIC  X(8).
        10  HOUSE-NUM             PIC  9(8).
        10  AGE                   PIC  9(2) USAGE COMP-3.
        10  LIC-DATE              PIC  X(8).
        10  LIC-STATUS            PIC  X.
        10  NUMBER-ACCIDENTS      PIC  99.
      05  VEHICLE.
        10  VEC-ID                PIC  X(15).
        10  MAKE                  PIC  X(20).
        10  MODEL                 PIC  X(20).
        10  VEC-VALUE             USAGE COMP-1.
        10  VEC-TYPE              PIC  X(2).
            88 SUV     VALUE 'SU'.
            88 SEDAN   VALUE 'SD'.
            88 PICKUP  VALUE 'PU'.
   01  RESPONSE.
      05  APPROVED                PIC  X.
      05  BASE-PRICE              USAGE COMP-2.
      05  DIS-PRICE               USAGE COMP-2.
      05  MSG-COUNT               PIC 9(5)  VALUE 0.
      05  MESSAGES                PIC  X(100)
                                  OCCURS 0 TO 100 TIMES
                                  DEPENDING ON MSG-COUNT.
```

Additional resources: You can find the `INSDEMO.cpy` copybook file in the additional information that is included in this book in the `code/Chapter3/CopybookBased` directory. See Appendix C, “Additional material” on page 335.

3.2.2 Creating a rule project

You can create a rule project in Rule Designer. A rule project enables you to manage, build, and debug the items that make up the business logic of your application.

Follow these steps to create the rule project in Rule Designer:

1. Click **File** → **New** → **Rule Project**. Select **Standard Rule Project** and click **Next**.
2. In the New Rule Project dialog, Project name field, enter `insurance-rules`, as shown in Figure 3-1 on page 37. Click **Finish**.

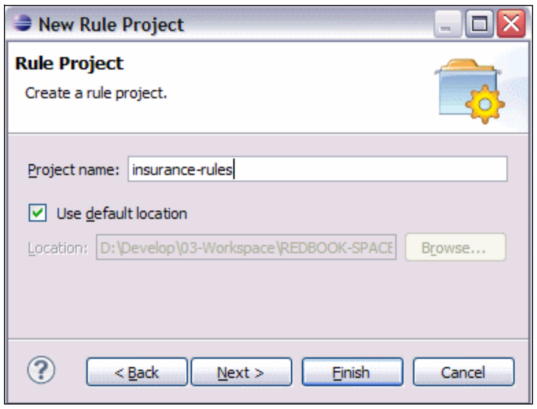


Figure 3-1 Creating a new rule project

The new rule project is created in the Rule Designer, as shown in Figure 3-2. For now, the rule project contains only empty folders.

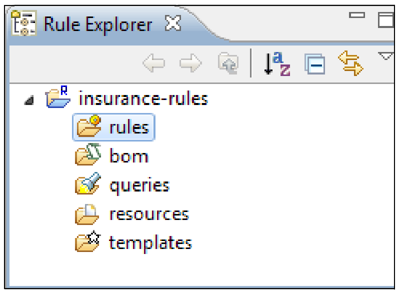


Figure 3-2 New rule project in the Rule Explorer view

3.2.3 Creating COBOL XOM from a COBOL copybook

To execute rules in a COBOL application, you generate the COBOL XOM from a COBOL copybook. A COBOL XOM provides the necessary COBOL-to-Java mapping so that you can create and execute your rules from a COBOL application.

To use the Rule Project Map to guide you through the COBOL XOM generation, follow these steps:

1. Select the **rules** folder in the newly created rule project (highlighted in Figure 3-2).
2. In the Design part of the Rule Project Map tab, click **Import XOM** (Figure 3-3).

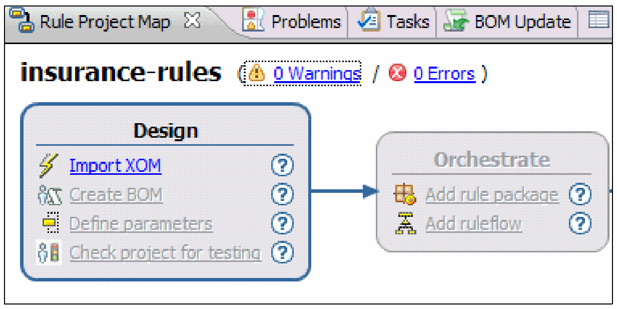


Figure 3-3 Importing a XOM into a rule project map

3. In the Import XOM dialog, select **COBOL Execution Object Model** (Figure 3-4). Click **OK**.

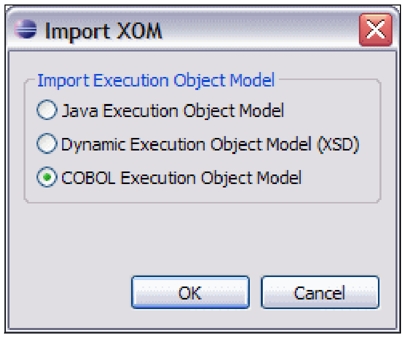

Import XOM
Import Execution Object Model
Java Execution Object Model
Dynamic Execution Object Model (XSD)
COBOL Execution Object Model
OK
Cancel

Figure 3-4 Selecting the COBOL Execution Object Model

4. On the Properties for insurance-rules dialog, click **Add** to add a COBOL Execution Object Model (Figure 3-5).

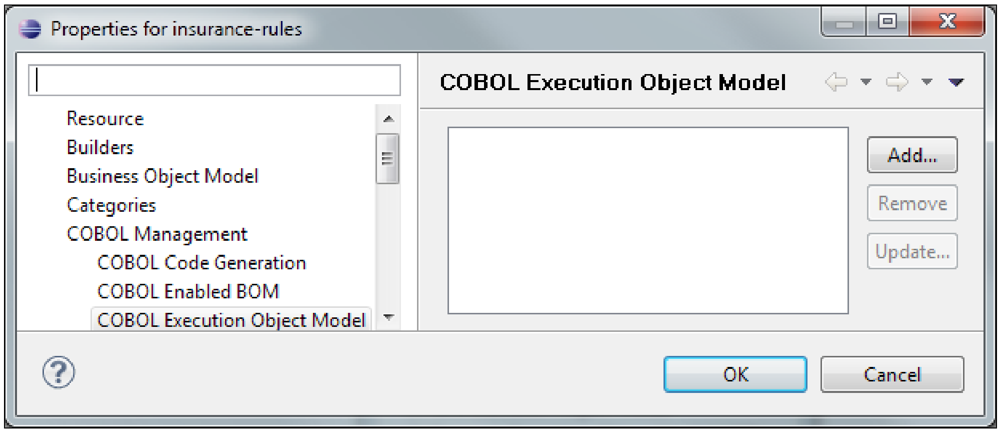
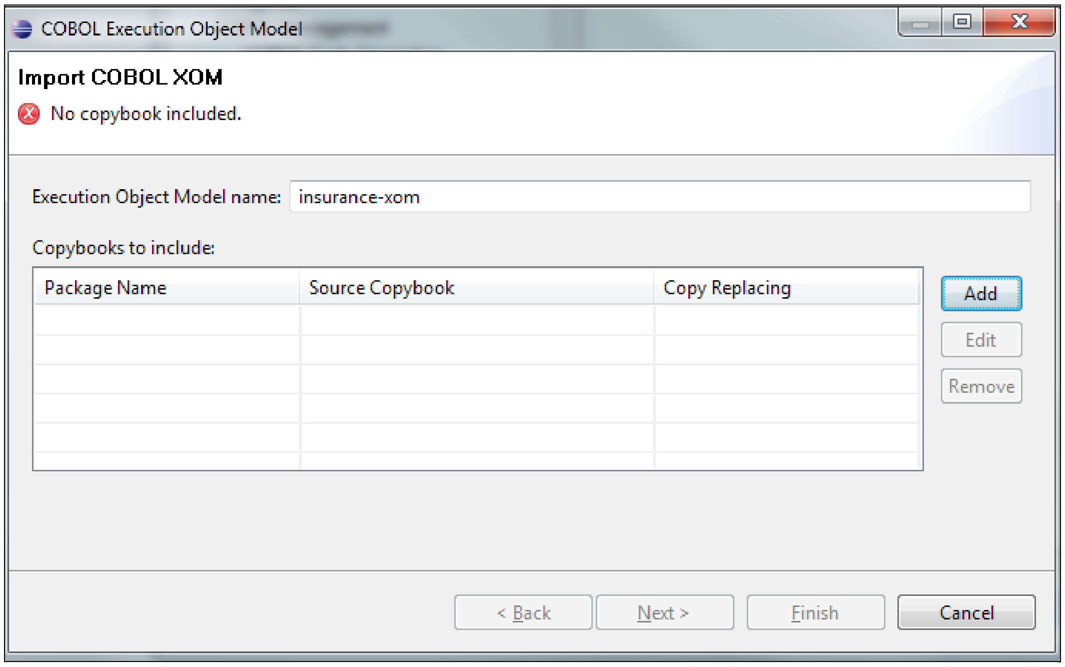


Figure 3-5 Adding a COBOL Execution Object Model

5. On the Import COBOL XOM dialog, in the Execution Object Model name field, enter `insurance-xom`, as shown in Figure 3-6. Click **Add**.


COBOL Execution Object Model

Import COBOL XOM

No copybook included.

Execution Object Model name: insurance-xom

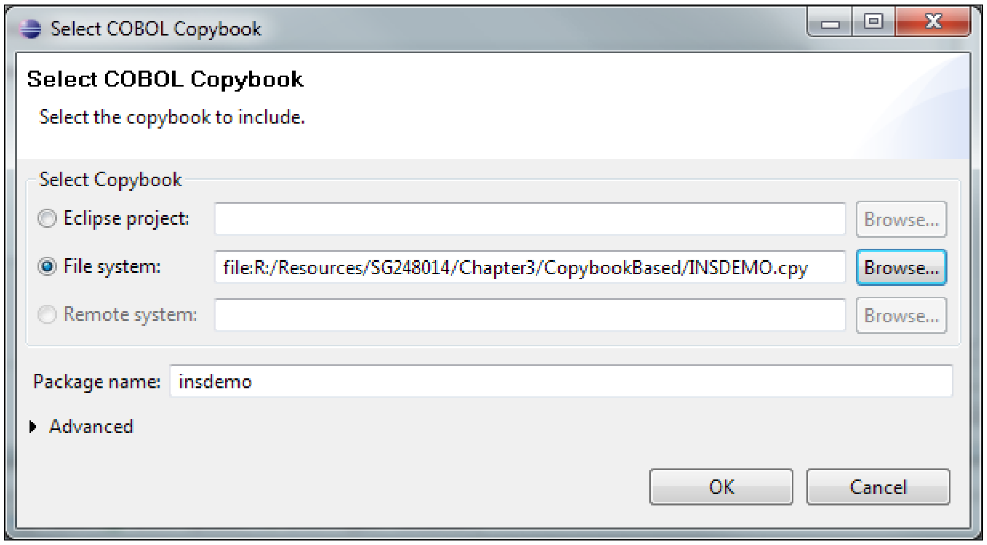
Copybooks to include:

Package Name	Source Copybook	Copy Replacing

Figure 3-6 Importing the COBOL XOM

6. On the Select COBOL Copybook dialog, select a COBOL copybook by using one of the **Browse** buttons and select the **INSDEMO.cpy** copybook (Figure 3-7). Click **OK**.

Additional resources: You can find the `INSDEMO.cpy` copybook file in the additional information that is included in this book in the `code/Chapter3/CopybookBased` directory. See Appendix C, "Additional material" on page 335.


Select COBOL Copybook

Select COBOL Copybook

Select the copybook to include.

Select Copybook

Eclipse project:

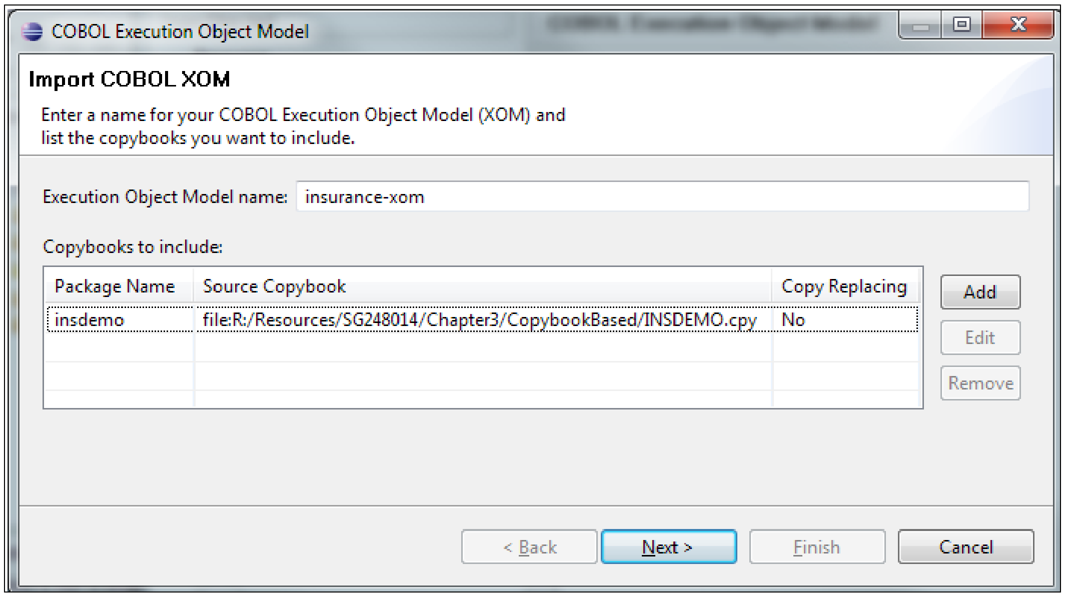
File system: file:R:/Resources/SG248014/Chapter3/CopybookBased/INSDEMO.cpy

Remote system:

Package name: insdemo

Figure 3-7 Selecting the COBOL copybook

7. On the resulting Import COBOL XOM dialog (Figure 3-8), click **Next**.



Enter a name for your COBOL Execution Object Model (XOM) and list the copybooks you want to include.

Execution Object Model name: insurance-xom

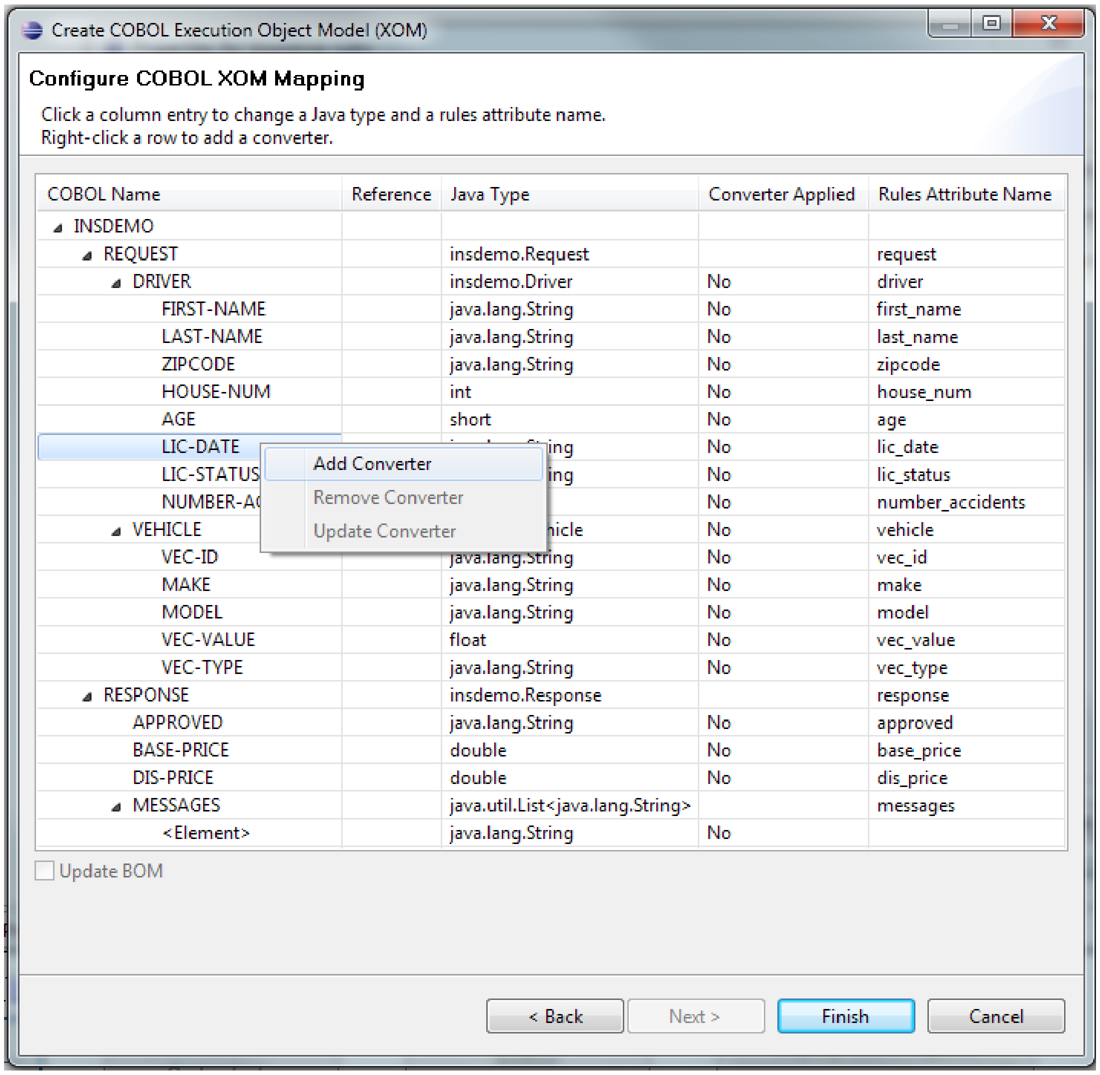
Copybooks to include:

Package Name	Source Copybook	Copy Replacing
insdemo	file:R:/Resources/SG248014/Chapter3/CopybookBased/INSDEMO.cpy	No

Figure 3-8 Importing the selected COBOL copybook

8. On the Configure COBOL XOM Mapping dialog, use the type converter to map two COBOL string items (PIC X) to a Java Date and a Boolean. Perform the following steps to change LIC-DATE from type String to Date by using the type converter:

   a. Expand the **REQUEST** item. Then, right-click the row that contains the **LIC-DATE** data item, and click **Add Converter**, as shown in Figure 3-9.



Configure COBOL XOM Mapping

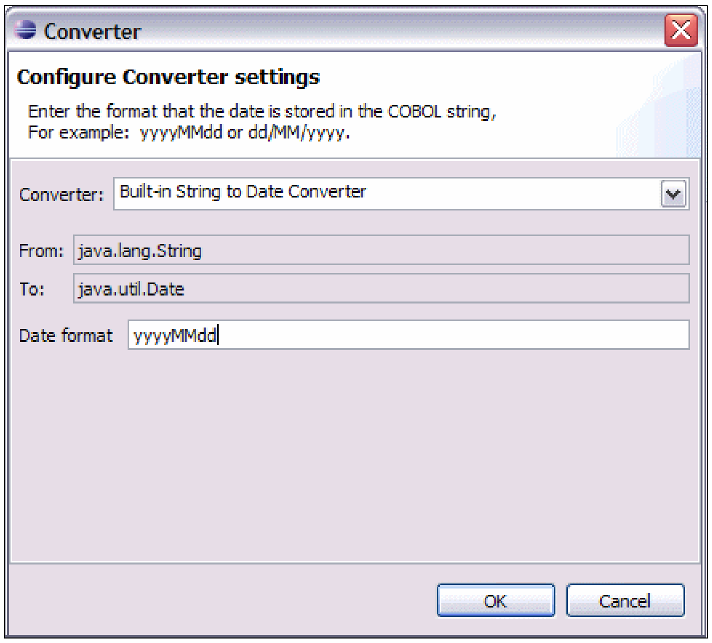
Click a column entry to change a Java type and a rules attribute name.
Right-click a row to add a converter.

COBOL Name	Reference	Java Type	Converter Applied	Rules Attribute Name
INSDEMO				
REQUEST		insdemo.Request		request
DRIVER		insdemo.Driver	No	driver
FIRST-NAME		java.lang.String	No	first_name
LAST-NAME		java.lang.String	No	last_name
ZIPCODE		java.lang.String	No	zipcode
HOUSE-NUM		int	No	house_num
AGE		short	No	age
LIC-DATE		ing	No	lic_date
LIC-STATUS		ing	No	lic_status
NUMBER-A			No	number_accidents
VEHICLE		icle	No	vehicle
VEC-ID		java.lang.String	No	vec_id
MAKE		java.lang.String	No	make
MODEL		java.lang.String	No	model
VEC-VALUE		float	No	vec_value
VEC-TYPE		java.lang.String	No	vec_type
RESPONSE		insdemo.Response		response
APPROVED		java.lang.String	No	approved
BASE-PRICE		double	No	base_price
DIS-PRICE		double	No	dis_price
MESSAGES		java.util.List<java.lang.String>		messages
<Element>		java.lang.String	No	

Figure 3-9 Adding a converter for LIC-DATE item

b. On the Configure Converter settings dialog, select **Built-in String to Date Converter**. Then, for the Date format field, enter `yyyyMMdd`, as shown in Figure 3-10. Click **OK**.

LIC-DATE: Use `yyyyMMdd` here to parse the LIC-DATE item value, such as `20110908`.



Enter the format that the date is stored in the COBOL string,
For example: yyyyMMdd or dd/MM/yyyy.

Converter: Built-in String to Date Converter

From: java.lang.String

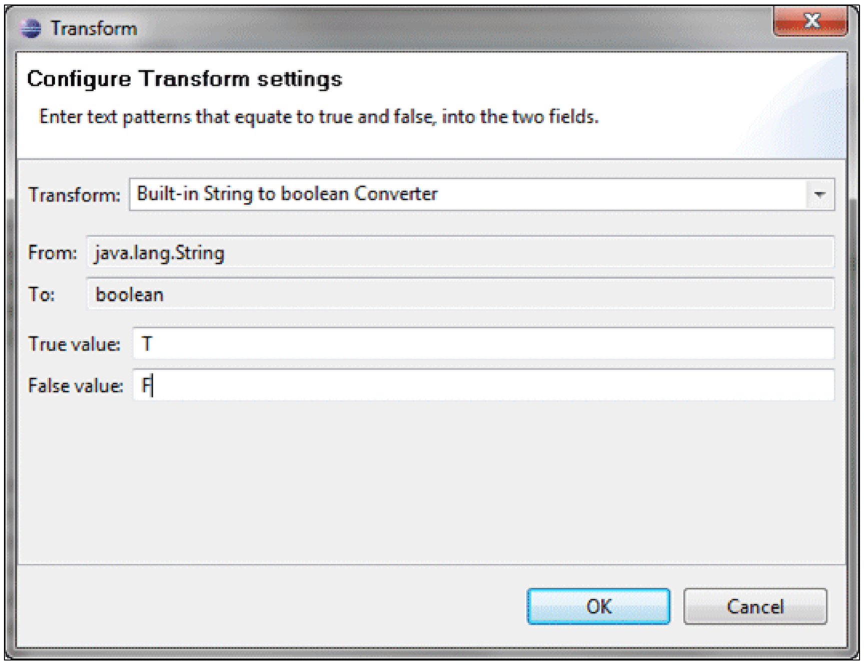
To: java.util.Date

Date format yyyyMMdd

Figure 3-10 Configuring the date converter

9. Next, change APPROVED from type String to Boolean, by using the type converter:
   a. Expand the **RESPONSE** item. Then, right-click the row that contains the **APPROVED** data item and click **Add Converter**.
   b. On the Configure Transform settings dialog, select **Built-in String to boolean Converter**.
   c. For the True value field, type `T` and for the False value field, type `F`, as shown in Figure 3-11.
   d. Click **OK**.

Values: The T and F values are COBOL values that represent True and False. You can also customize these values as Y/N, YES/NO, and so on.


Transform

Configure Transform settings

Enter text patterns that equate to true and false, into the two fields.

Transform: Built-in String to boolean Converter

From: java.lang.String

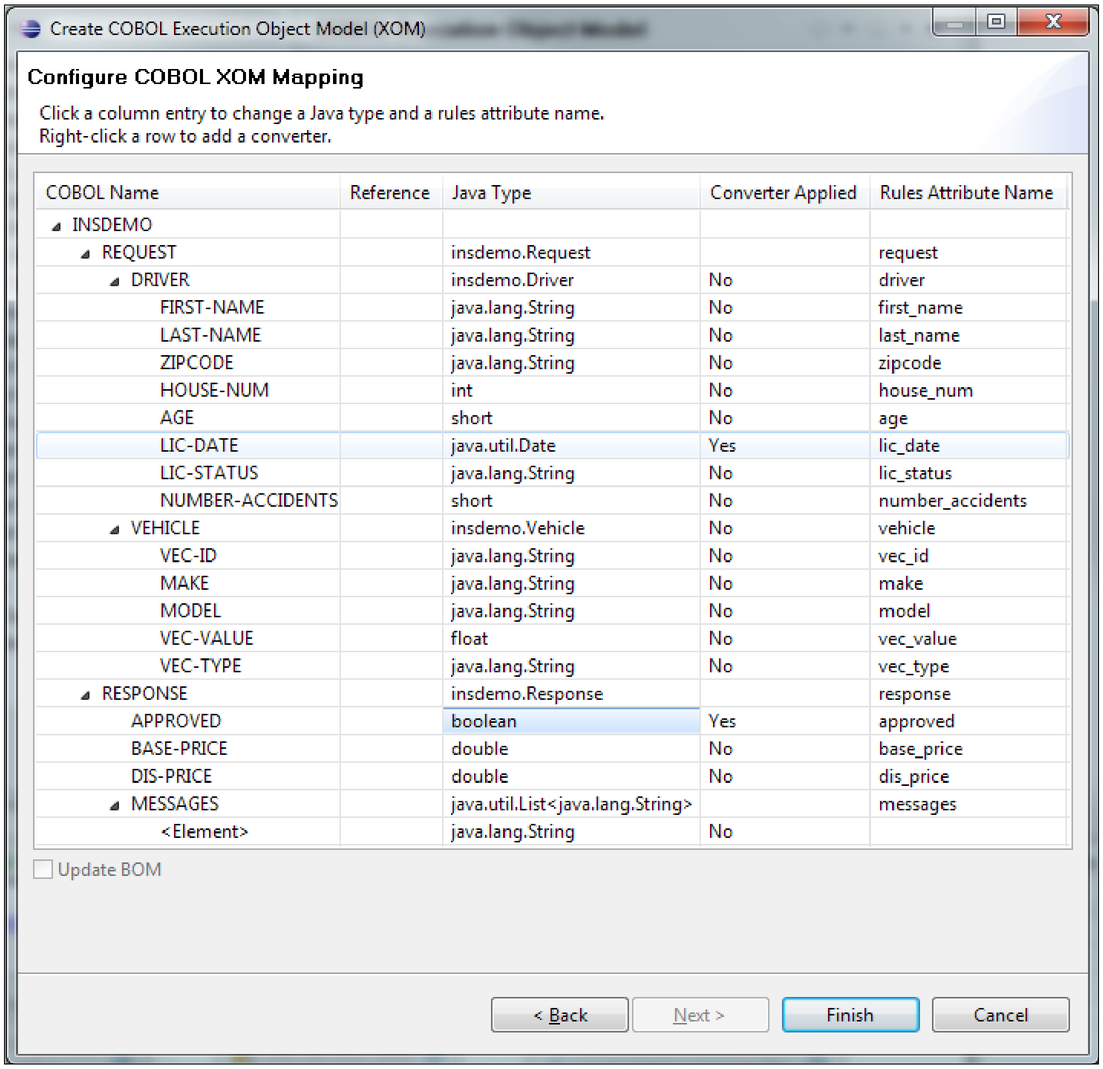
To: boolean

True value: T

False value: F

Figure 3-11 Configuring the Boolean converter

10.Click **Finish** to create the COBOL XOM (Figure 3-12).



Configure COBOL XOM Mapping

Click a column entry to change a Java type and a rules attribute name.
Right-click a row to add a converter.

COBOL Name	Reference	Java Type	Converter Applied	Rules Attribute Name
◢ INSDEMO				
◢ REQUEST		insdemo.Request		request
◢ DRIVER		insdemo.Driver	No	driver
FIRST-NAME		java.lang.String	No	first_name
LAST-NAME		java.lang.String	No	last_name
ZIPCODE		java.lang.String	No	zipcode
HOUSE-NUM		int	No	house_num
AGE		short	No	age
LIC-DATE		java.util.Date	Yes	lic_date
LIC-STATUS		java.lang.String	No	lic_status
NUMBER-ACCIDENTS		short	No	number_accidents
◢ VEHICLE		insdemo.Vehicle	No	vehicle
VEC-ID		java.lang.String	No	vec_id
MAKE		java.lang.String	No	make
MODEL		java.lang.String	No	model
VEC-VALUE		float	No	vec_value
VEC-TYPE		java.lang.String	No	vec_type
◢ RESPONSE		insdemo.Response		response
APPROVED		boolean	Yes	approved
BASE-PRICE		double	No	base_price
DIS-PRICE		double	No	dis_price
◢ MESSAGES		java.util.List<java.lang.String>		messages
<Element>		java.lang.String	No	

☐ Update BOM

Figure 3-12 Finishing the data type configuration

11. Click **OK** to close the Properties window (Figure 3-13).

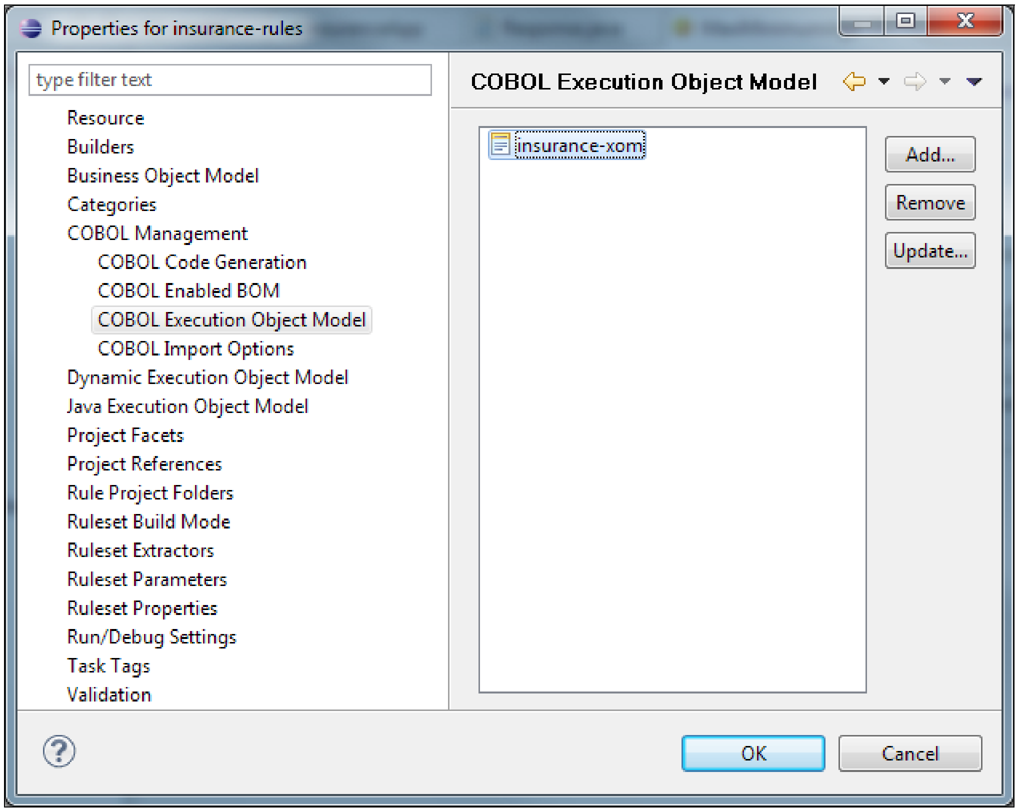


Figure 3-13 Generating the COBOL XOM

12. The following artifacts are created, as shown in Figure 3-14:

   – Java XOM project: `insurance-xom`
   – COBOL XOM:
      • Configuration file: `CobolXomConfig.xml`
      • Marshaller: `insurance-xom-xmarshaller.jar`

Important: The `insurance-xom` project and COBOL XOM files are generated artifacts. Do *not* change these artifacts manually.

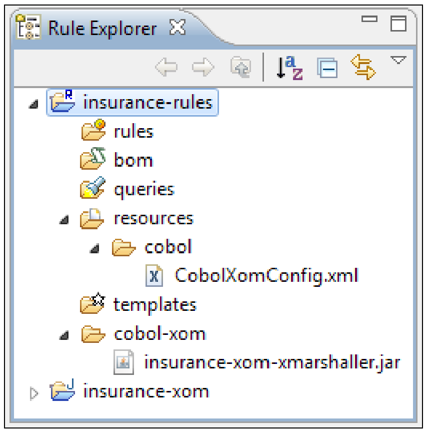


Figure 3-14 COBOL XOM artifacts

3.2.4 Creating a business object model from the Java XOM

The *business object model (BOM)* is a business layer that is used to author business rules. This section describes how to create a BOM in Rule Designer that is based on the Java XOM that you created in 3.2.3, “Creating COBOL XOM from a COBOL copybook” on page 37.

Follow these steps to create a BOM from the COBOL XOM:

1. In the Design part of the Rule Project Map tab, click **Create BOM**, as shown in Figure 3-15.

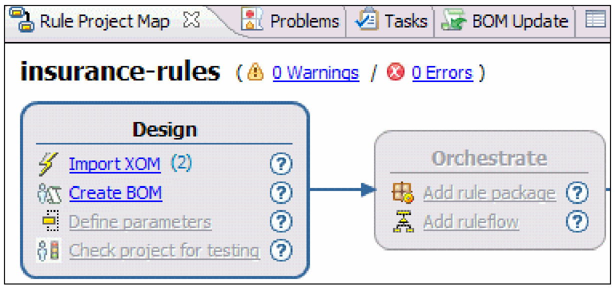
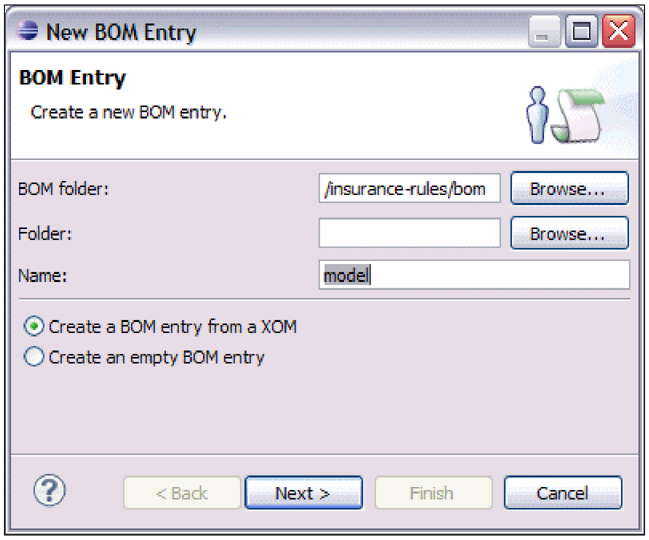


Figure 3-15 Select Create BOM from the Rule Project Map

2. In the New BOM Entry dialog, in the Name field, accept the default name for the BOM entry. In this scenario, the default name is `model`. Ensure that the **Create a BOM entry from a XOM** option is selected (Figure 3-16) and click **Next**.



BOM Entry

Create a new BOM entry.

BOM folder:	/insurance-rules/bom
Folder:	
Name:	model

[✓ Create a BOM entry from a XOM]
Create an empty BOM entry

Figure 3-16 Creating a BOM entry from a Java XOM

3. On the BOM Entry dialog, in the Choose a XOM entry field, click **Browse XOM**. On the Browse XOM dialog, select **insurance-xom**, as shown in Figure 3-17, and click **OK**.

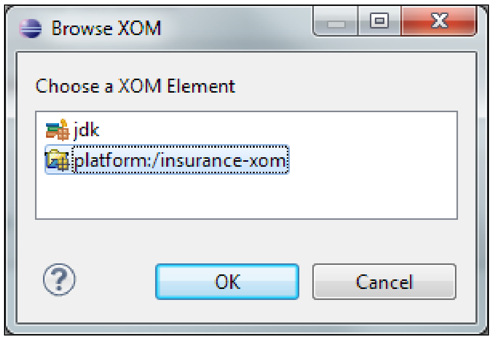


Figure 3-17 Selecting a generated Java XOM

4. In the Select classes field of the BOM Entry dialog, select the **XOM** package. When you select the package, you automatically select all the classes that it contains, as shown in Figure 3-18. Click **Finish**.

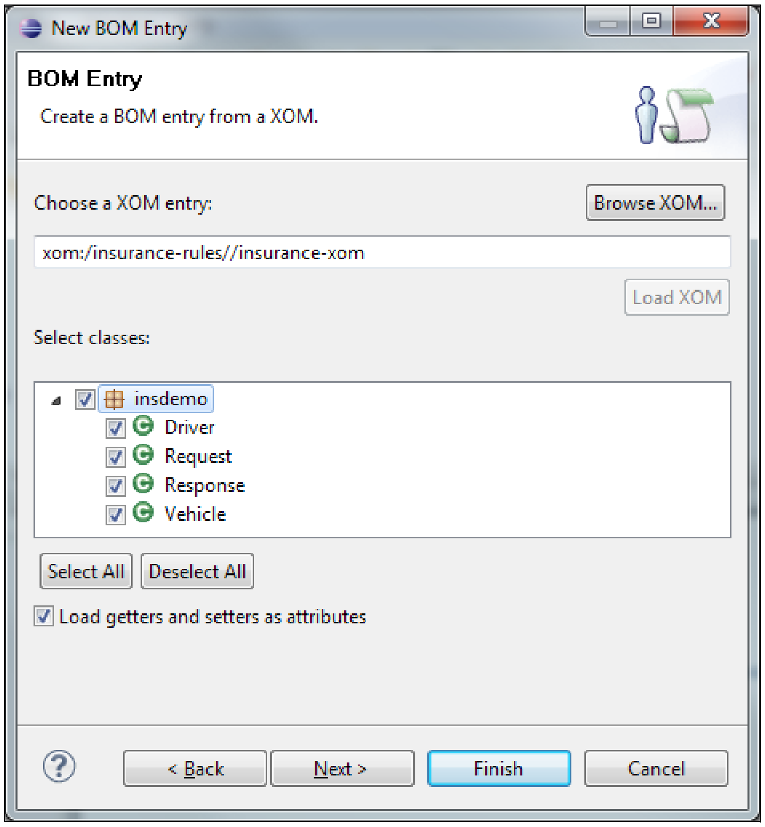


Figure 3-18 Selecting a XOM package to import all classes

5. In the Rule Explorer view, the `bom` folder contains a new BOM entry model, as shown in Figure 3-19.

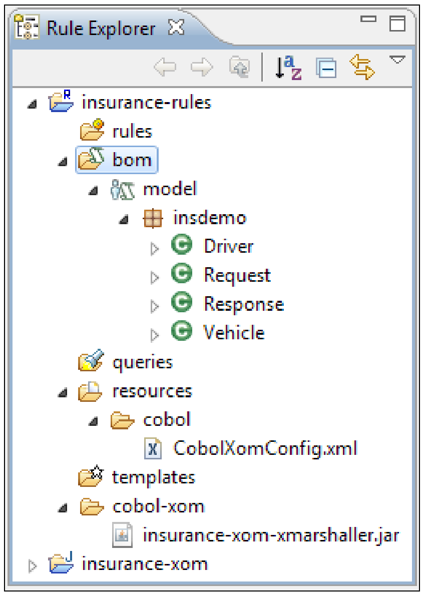
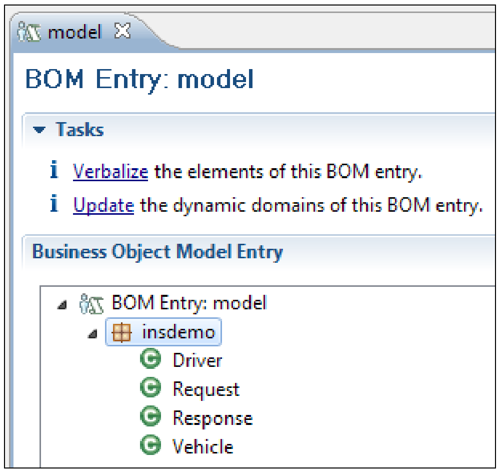


Figure 3-19 Viewing the generated BOM

6. View the generated BOM and its verbalization:
   a. In the Rule Explorer view, double-click **bom** → **model** to open the BOM editor.
   b. In the BOM Editor, expand the insdemo package to view the generated BOM, as shown in Figure 3-20.
   c. Double-click the **Driver** class to view the default class verbalization.



Verbalize the elements of this BOM entry.

Update the dynamic domains of this BOM entry.

Figure 3-20 Viewing the generated BOM Entry model in the BOM editor

d. The resulting Class Verbalization section (of the Class Driver window) is shown (Figure 3-21).

Language: The default verbalization is in English. If you are working in a localized version of Rule Designer, you can verbalize the BOM classes in the language of your locale.

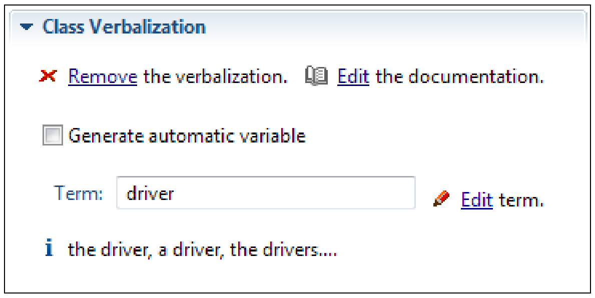

Class Verbalization
Remove the verbalization.
Edit the documentation.
Generate automatic variable
Term:
driver
Edit term.
the driver, a driver, the drivers....

Figure 3-21 Viewing the default verbalization

3.2.5 Declaring ruleset parameters

Ruleset parameters provide the means to exchange data between a COBOL application and the rule application. You define ruleset parameters by name, type, and direction.

In this example, you decide on the status of an insurance request and response, so that you create ruleset parameters for the Request and Response classes. You use the IN direction for the request parameter. The value of the request parameter is provided as input from the COBOL client application on execution. The direction for the response parameter must be IN_OUT. The value of the request parameter is set by the IN value passed by the client and then updated by the engine on the way OUT. The updated value is returned to the client.

Important: You *cannot* use the OUT parameter direction with zRES, because COBOL programs do not support memory allocation dynamically.

Follow these steps to declare ruleset parameters:

1. In the Design part of the Rule Project Map tab, click **Define parameters**, as shown in Figure 3-22.

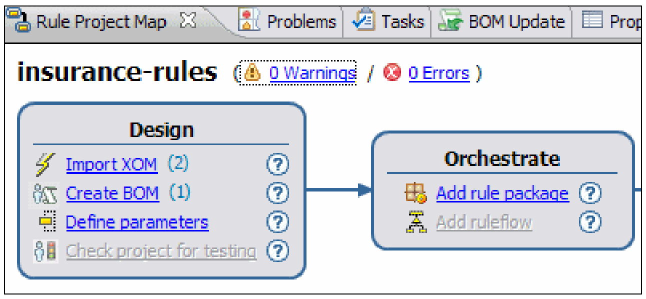
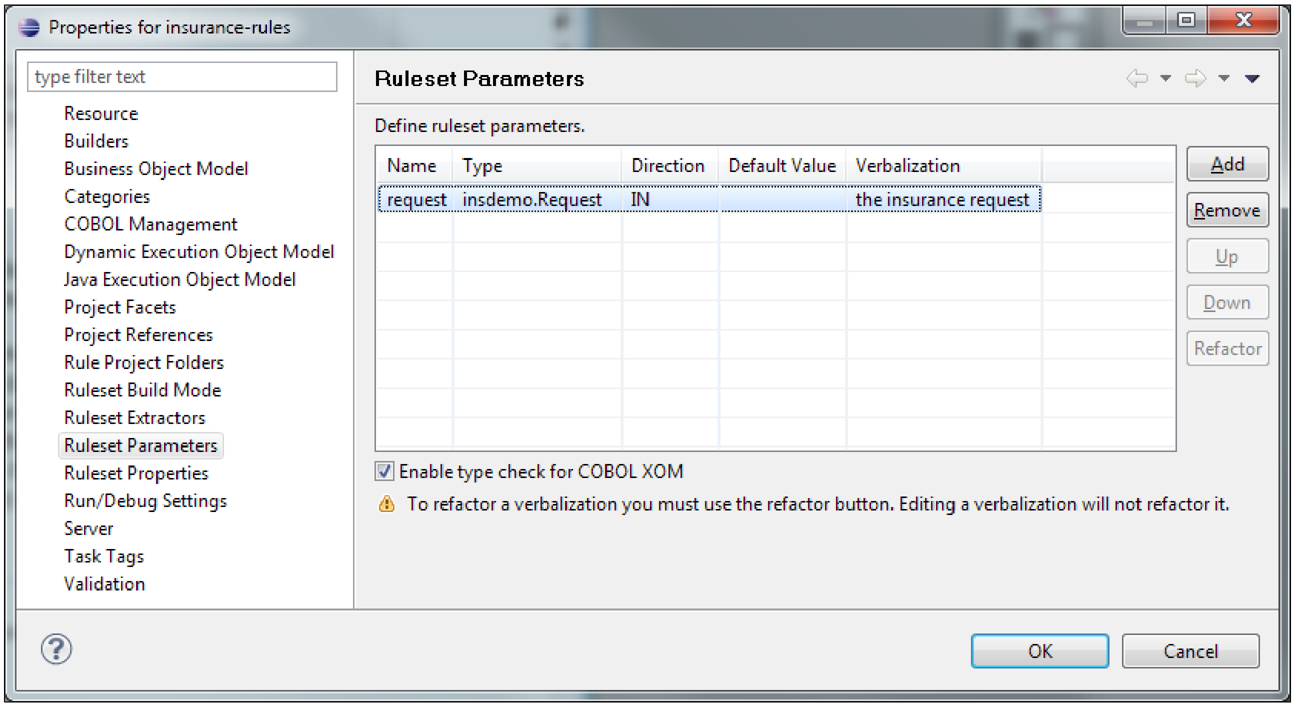


Figure 3-22 Selecting Define parameters option

2. In the Ruleset Parameters dialog, select **Enable type check for COBOL XOM**.
3. To define a request parameter, click **Add**. Then, change the following default values, as shown in Figure 3-23:
   – In the Name column, type `request`.
   – In the Type column, click the ellipsis (**...**) on the right of the cell, and select **Request**. The `xom.Request` entry is entered in the cell automatically.
   – In the Direction column, select the **IN** direction.
   – In the Verbalization column, type `the insurance request`.



Ruleset Parameters

Define ruleset parameters.

Name	Type	Direction	Default Value	Verbalization
request	insdemo.Request	IN		the insurance request

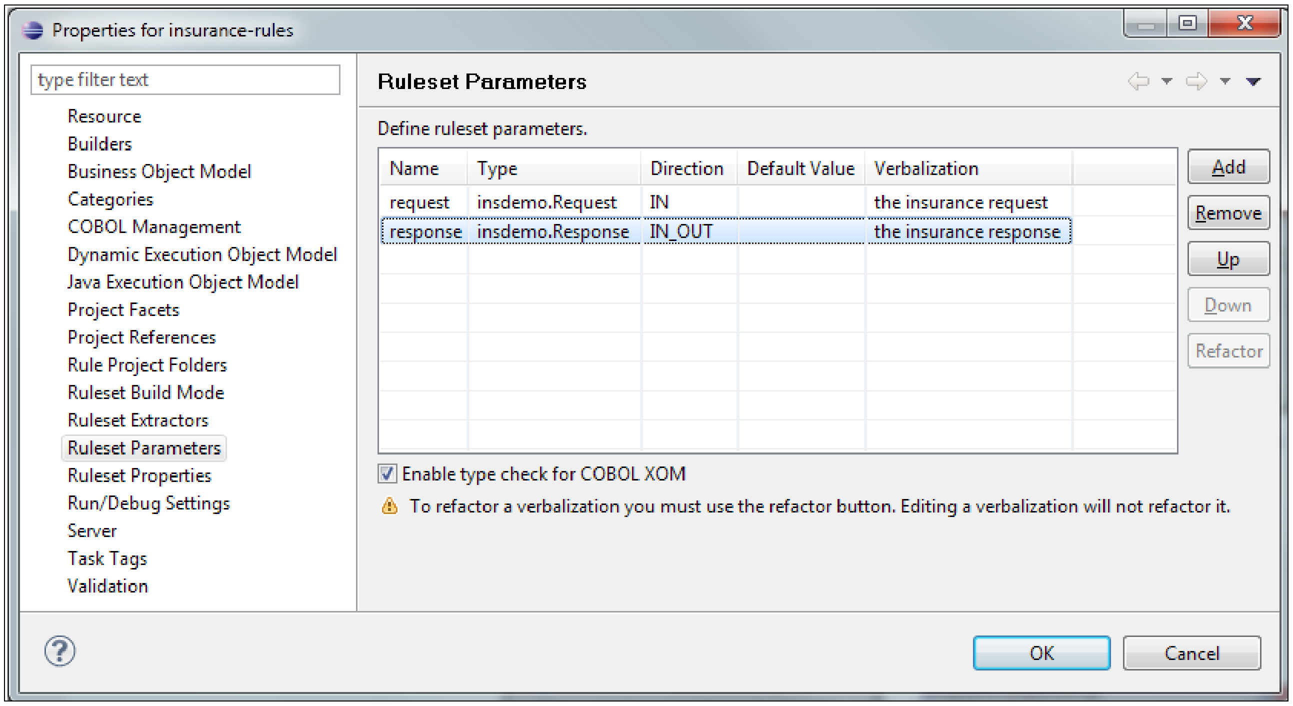
☑ Enable type check for COBOL XOM

To refactor a verbalization you must use the refactor button. Editing a verbalization will not refactor it.

Figure 3-23 Adding the ruleset parameter for request

4. To define the response parameter, click **Add**. Then, change the following default values, as shown in Figure 3-24:
   – In the Name column, type `response`.
   – In the Type column, select **Response**. The `xom.Response` entry is added to the cell automatically.
   – In the Direction column, select the **IN_OUT** direction.
   – In the Verbalization column, type `the insurance response`.

   Click **OK**.



Define ruleset parameters.

Name	Type	Direction	Default Value	Verbalization
request	insdemo.Request	IN		the insurance request
response	insdemo.Response	IN_OUT		the insurance response

☑ Enable type check for COBOL XOM

To refactor a verbalization you must use the refactor button. Editing a verbalization will not refactor it.

Figure 3-24 Add the ruleset parameter for response

3.2.6 Adding BOM methods and mapping them to the XOM

You use methods to specify conditions and actions in your rules. You create methods in the Rule Designer. When you add methods to the BOM, you use BOM to XOM mapping in the BOM Editor to implement the method.

Important: You *cannot* map the BOM method to a Java XOM method, because you must not change the XOM.

This section describes how to add the following BOM methods:

- addMessage: Defines what is needed to pass information from the rules.
- reject: Identifies whether the insurance request was rejected.

Adding the addMessage method

To add the addMessage method, follow these steps:

1. In the Outline view, expand the model package, and double-click the **Response** class, as shown in Figure 3-25.

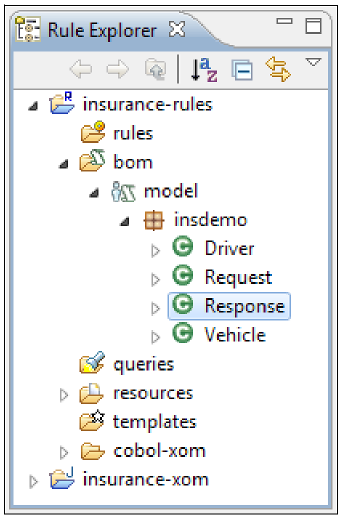
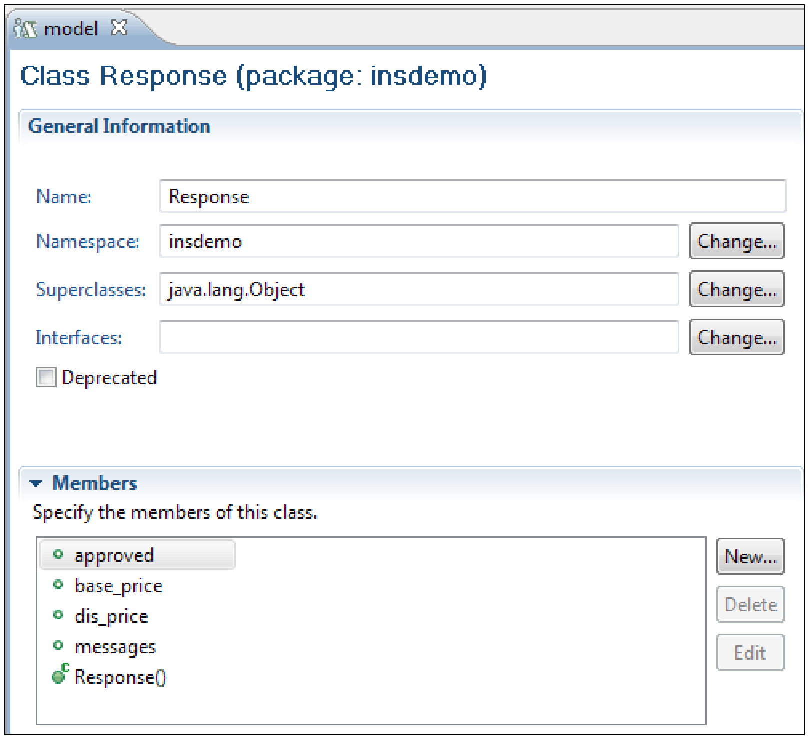


Figure 3-25 Selecting the Response class

2. On the Class Response page of the BOM editor, to the right of the Members section, click **New**, as shown in Figure 3-26.


model

Class Response (package: insdemo)

General Information

Name: Response

Namespace: insdemo

Superclasses: java.lang.Object

Interfaces:

Deprecated

Members

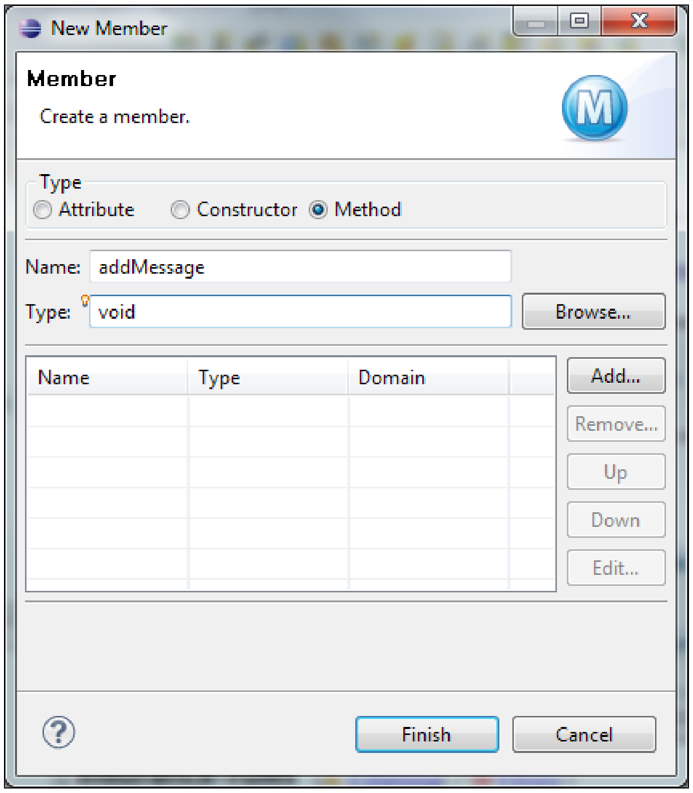
Specify the members of this class.

approved
base_price
dis_price
messages
Response()

Figure 3-26 Creating a new member

3. In the New Member dialog (Figure 3-27), enter the following information:
   – For the Type, select **Method**.
   – For the Name, enter `addMessage`.
   – For the Type, enter `void`.

   Click **Add**.


New Member
Member
Create a member.
Type: ( ) Attribute ( ) Constructor (●) Method
Name: addMessage
Type: void

Name	Type	Domain

Figure 3-27 Creating a new method for addMessage

4. In the Method Argument dialog (Figure 3-28), enter the following information:
   – For the Name, enter `msg`.
   – For the Type, enter `java.lang.String`.

   Click **OK**, and then click **Finish** on the New Member dialog.

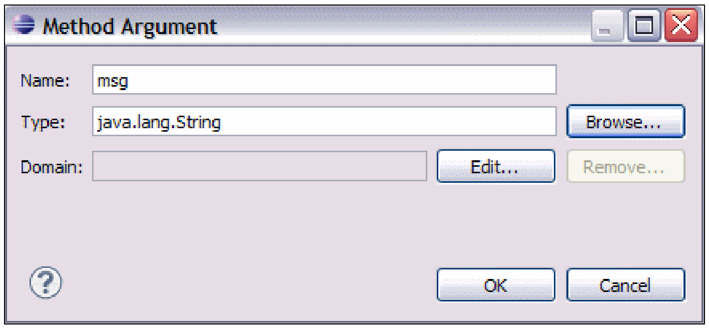
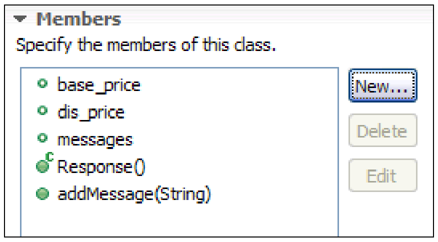

Method Argument
Name: msg
Type: java.lang.String
Domain:

Figure 3-28 Adding the method argument

5. On the Class page of the BOM editor, the Members list now includes the addMessage(String) method, as shown in Figure 3-29. Double-click the **addMessage** method.

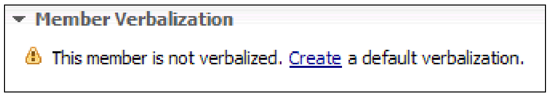

Members

Specify the members of this class.

- base_price
- dis_price
- messages
- Response()
- addMessage(String)

Figure 3-29 addMessage method created

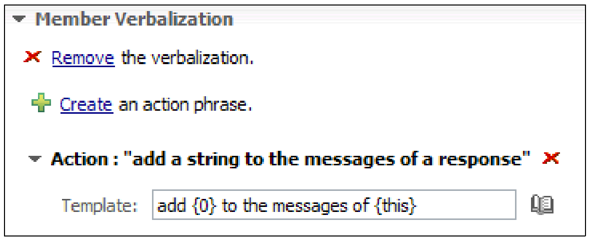
6. In the Member Verbalization section of the BOM editor (Figure 3-30), click **Create** to view the default verbalization.


Member Verbalization

This member is not verbalized. Create a default verbalization.

Figure 3-30 Creating verbalization

7. The default verbalization of the addMessage class is now displayed. Keep the default verbalization of **add {0} to the messages of {this}**, as shown in Figure 3-31.


Member Verbalization

Remove the verbalization.

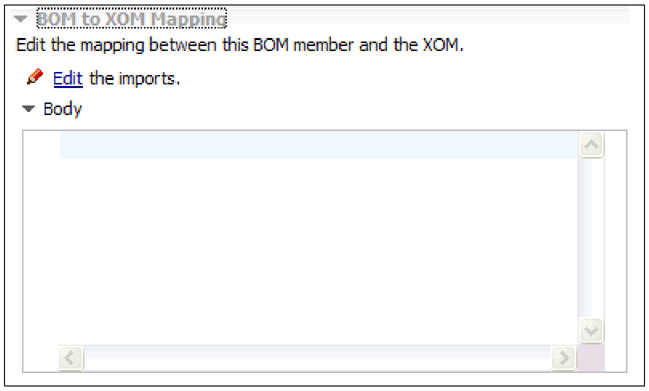
Create an action phrase.

Action : "add a string to the messages of a response"

Template: add {0} to the messages of {this}

Figure 3-31 Keeping the default verbalization

8. Scroll down to the BOM to XOM Mapping section of the BOM editor and expand it to activate the BOM to XOM Mapping editor, as shown in Figure 3-32.


BOM to XOM Mapping

Edit the mapping between this BOM member and the XOM.

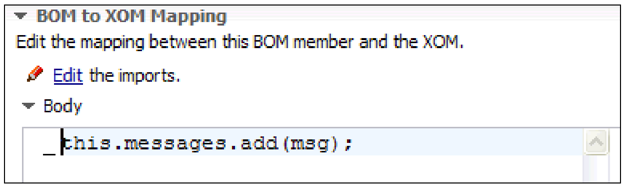
Edit the imports.

Body

Figure 3-32 Activating the BOM to XOM Mapping editor

9. Enter the following Java code (Figure 3-33):

```
this.messages.add(msg) ;
```



```
this.messages.add(msg);
```

Figure 3-33 Adding method implementation

10. Save your work.

Adding the reject method

To add the reject method, follow these steps:

1. Double-click the **Response** class in the Rule Explorer. Under the Members section, click **New**, as shown in Figure 3-34.

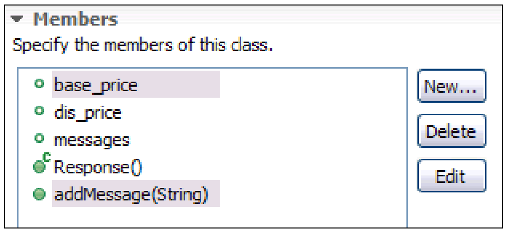
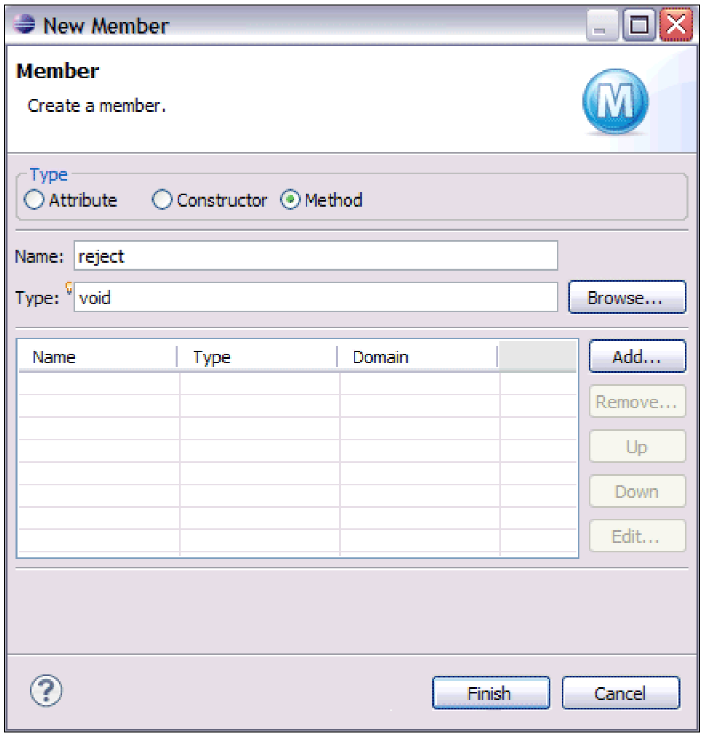


Figure 3-34 Creating a new reject method

2. In the New Member dialog (Figure 3-35), enter the following information:
   - For the Type, select **Method**.
   - For the Name, enter `reject`.
   - For the Type, enter `void`.

   Click **Finish**.



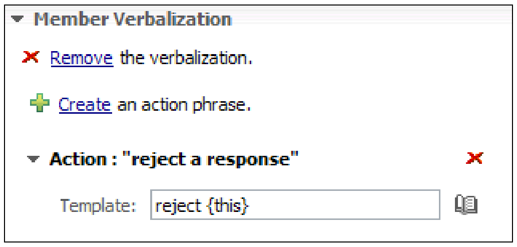
Type: Attribute Constructor [✓ Method]

Name: reject

Type: void

Figure 3-35 Defining the method argument

3. Create the default verbalization for the reject method. Double-click the `reject()` method. Then, click **Create** and accept the default verbalization of `reject {this}`, as shown in Figure 3-36.



Action : "reject a response"

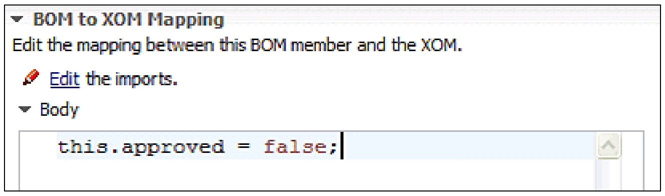
Template: reject {this}

Figure 3-36 Defining verbalization for the reject method

4. Scroll down to the BOM to XOM Mapping section of the BOM editor and expand it to activate the BOM to XOM Mapping editor. Enter the following Java code, as shown in Figure 3-37.

   Type `this.approved = false;`

   Save your work.



BOM to XOM Mapping

Edit the mapping between this BOM member and the XOM.

Edit the imports.

Body

```
this.approved = false;
```

Figure 3-37 Implementing the reject method

The Rule Explorer now shows that these members are present in their classes, as shown in Figure 3-38.

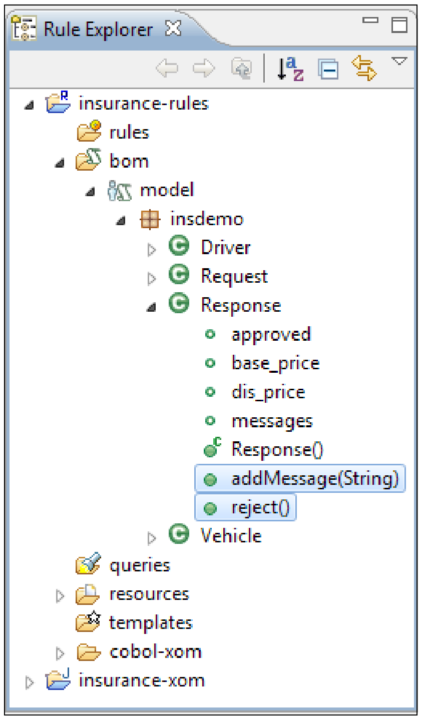


Figure 3-38 Viewing the new BOM methods

3.2.7 Creating the ruleflow

Before writing the rules, you orchestrate how the rules execute. You control the order in which rules are executed by using *ruleflows*. When defining the flow of execution, you organize rules into packages that contain related rules. This section explains how to create a package that relates to the validation rules.

Follow these steps to create a ruleflow:

1. In Rule Designer, in the Orchestrate part of the Rule Project Map, click **Add rule package**, (Figure 3-39).

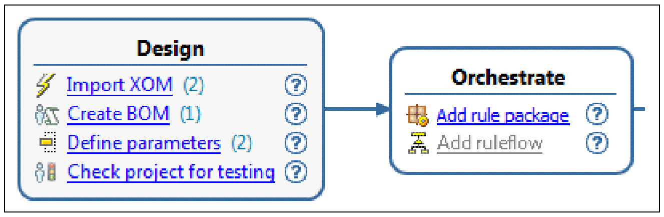
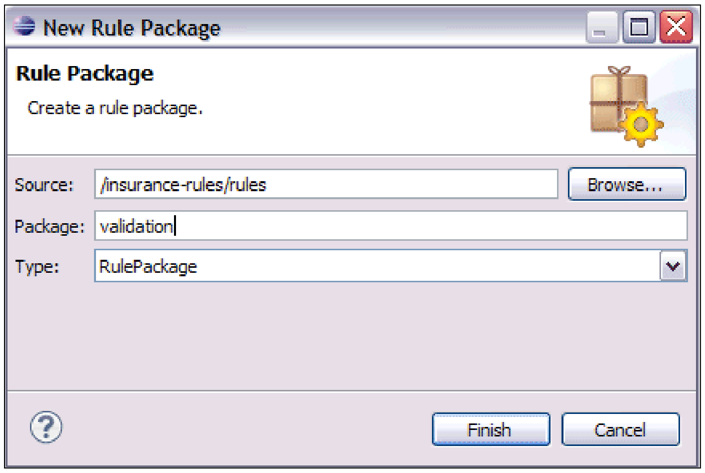


Figure 3-39 Adding a new rule package

2. In the New Rule Package dialog, in the Package field, enter `validation` and click **Finish**, (Figure 3-40).



New Rule Package

Rule Package

Create a rule package.

Source: /insurance-rules/rules

Package: validation

Type: RulePackage

Figure 3-40 Entering the validation package name

3. To create the ruleflow, in the Orchestrate part of the Rule Project Map, click **Add ruleflow**, (Figure 3-41).

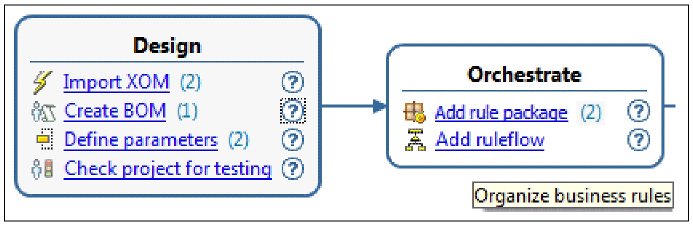
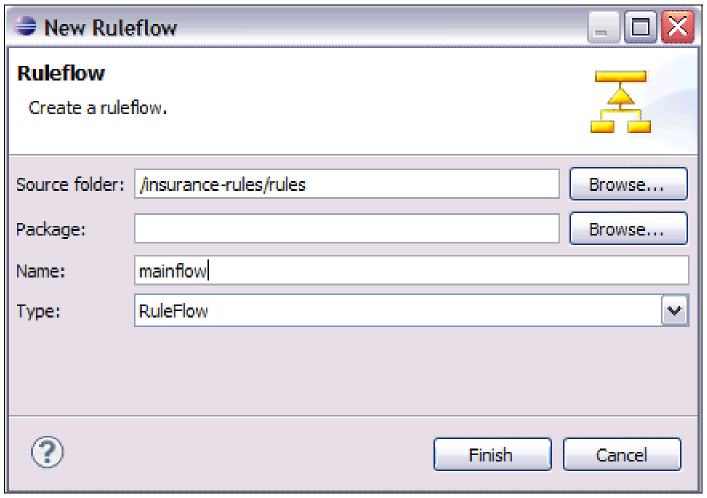


Figure 3-41 Adding a new ruleflow

4. In the New Ruleflow dialog, in the Name field, enter `mainflow` and click **Finish** (Figure 3-42).


New Ruleflow

Ruleflow

Create a ruleflow.

Source folder: /insurance-rules/rules

Package:

Name: mainflow

Type: RuleFlow

Figure 3-42 Entering the ruleflow name

5. To create a ruleflow by using the Ruleflow diagram that is shown in Figure 3-43 on page 60 and Figure 3-44 on page 60, complete the following steps:

   a. Add the start node, which is the starting point of the ruleflow. Click the start node icon ( ) in the ruleflow diagram toolbar and drop it in the ruleflow diagram.

   b. Add the end node, which is the endpoint of the ruleflow. Click the end node icon ( ) in the ruleflow diagram toolbar and drop it in the ruleflow diagram.

c. Add the task for the validation rule package, which is the rule task of the ruleflow. Click the **validation** rule package in the Rule Explorer view and drag it into the ruleflow diagram as shown in Figure 3-43.

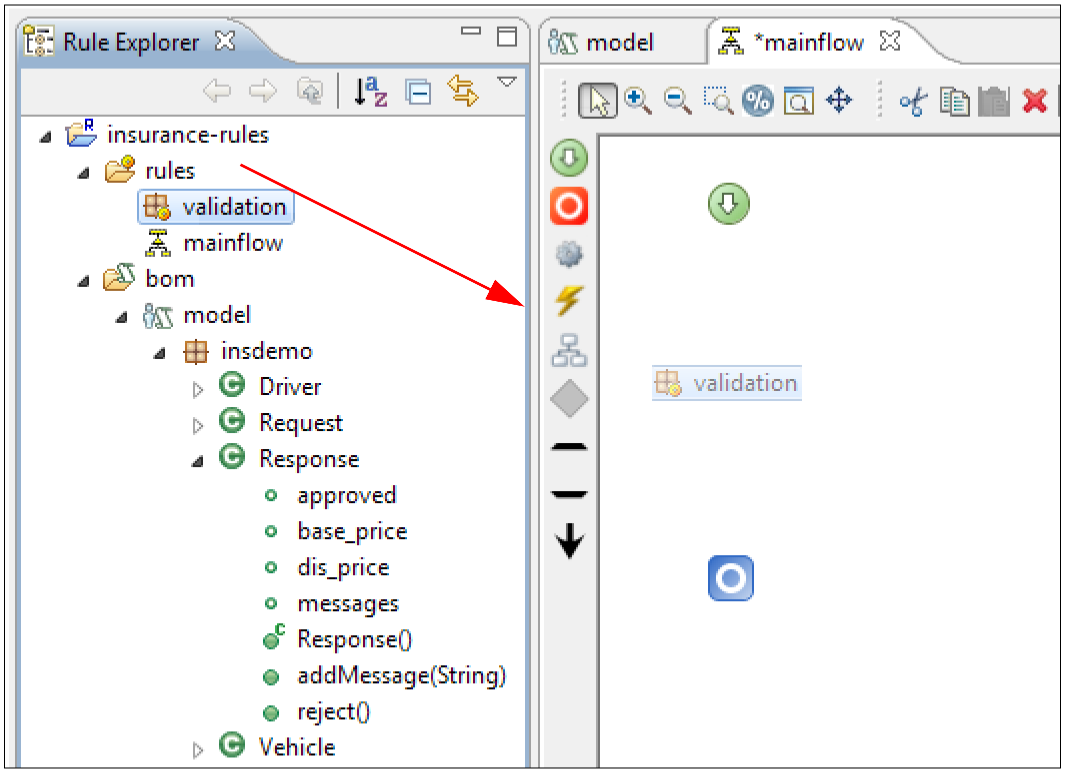


Figure 3-43 Dragging the validation package to the canvas

d. Connect the elements:
   i. Click the arrow icon (↓) to start connection mode.
   ii. Click the start node icon (⇩) and then click the **validation** task box.
   iii. Click the **validation** task box again, and finally, click the end node icon (○).

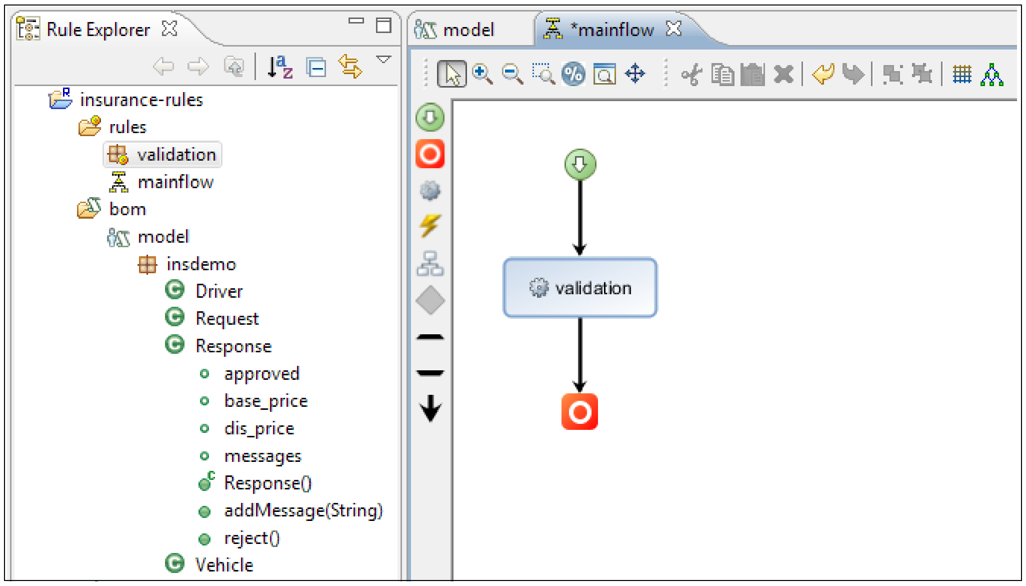


Figure 3-44 Designing the ruleflow

6. Next, you can optionally refine the diagram by clicking the button. Figure 3-45 shows the diagram.

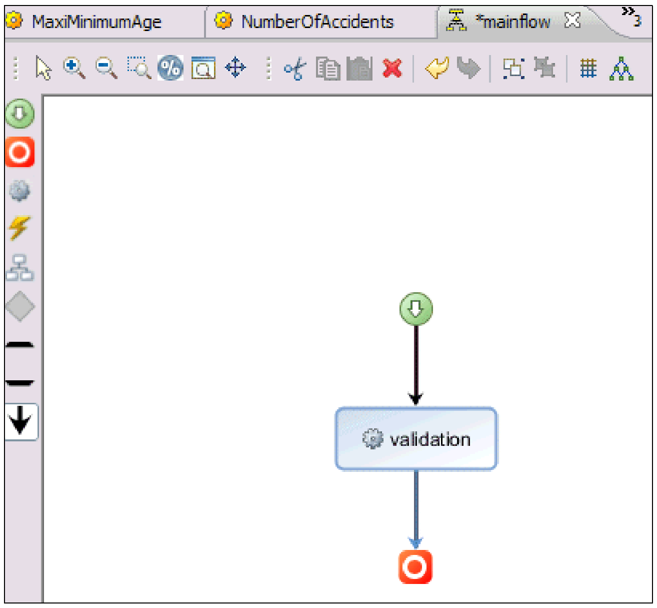


Figure 3-45 Refining the ruleflow

7. Save your work.

3.2.8 Authoring rules

This section explains how to write action rules and put them into the relevant package. You can create the following rules in Rule Designer for the validation packages:

- MaxiMinimumAge rule
- NumberOfAccidents rule

To create the action rules, follow these steps:

1. Create the MaxiMinimumAge rule:

   a. In the rules project, right-click **validation** and then click **New** → **Action Rule**, as indicated in Figure 3-46.

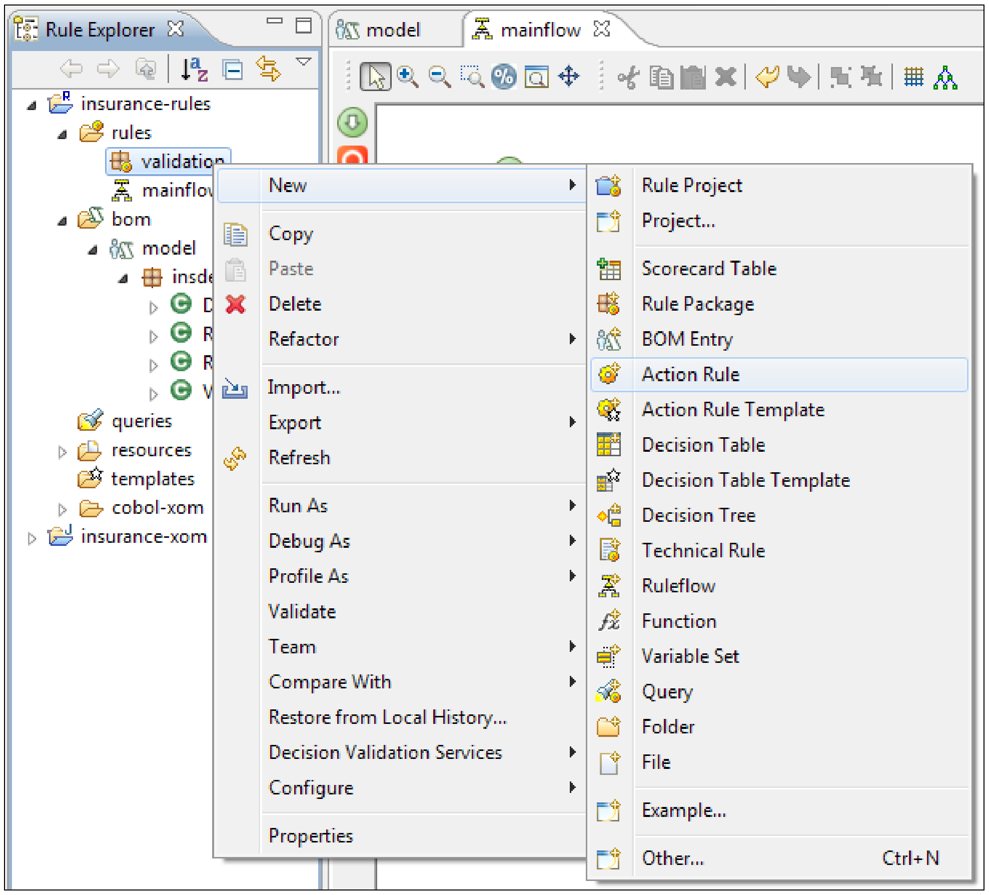
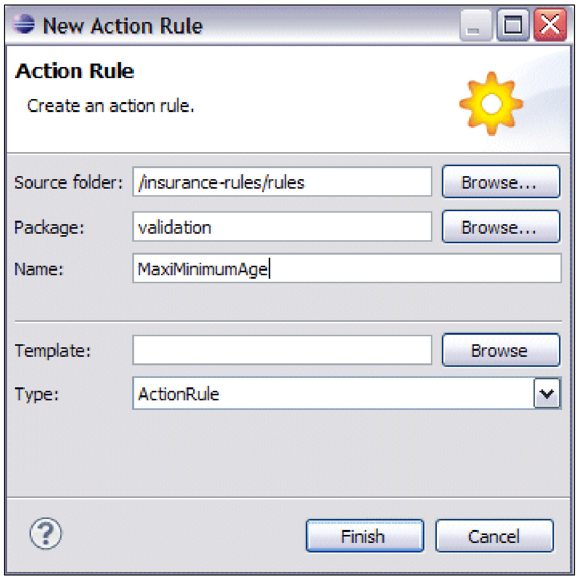


Figure 3-46 Creating an action rule

b. In the New Action Rule dialog, enter `MaxiMinimumAge` in the Name field, as shown in Figure 3-47. Click **Finish**.



Source folder: /insurance-rules/rules
Package: validation
Name: MaxiMinimumAge
Template:
Type: ActionRule

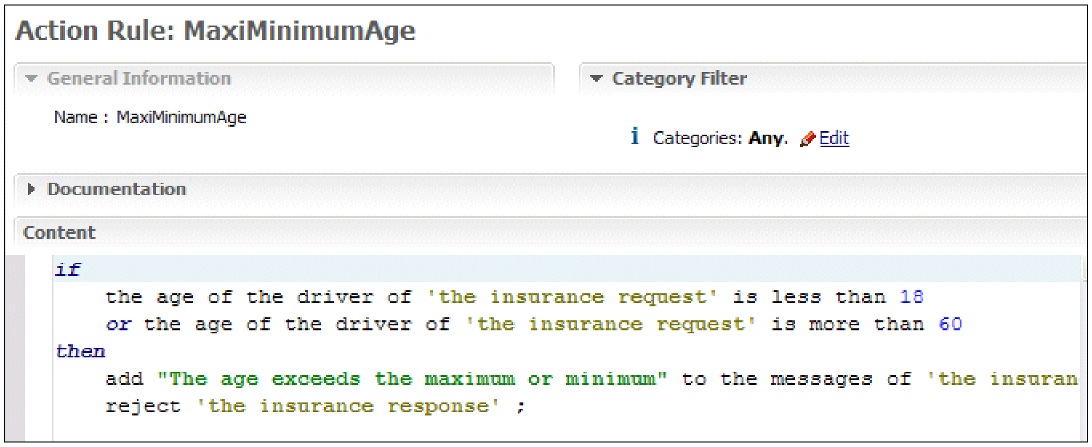
Figure 3-47 Entering the rule name

c. The new action rule, MaxiMinimumAge, is displayed in the Rule Explorer view and the Intellirule Editor opens. Enter the validation with the MaxiMinimumAge rule, as shown in Example 3-2.

Example 3-2 Validation with the MaxiMinimumAge rule

```
if
    the age of the driver of 'the insurance request' is less than 18
    or the age of the driver of 'the insurance request' is more than 60
then
    add "The age exceeds the maximum or minimum" to the messages of 'the
insurance response' ;
    reject 'the insurance response' ;
```

Figure 3-48 shows the generated action rule.



Name : MaxiMinimumAge

Categories: **Any**.

```
if
    the age of the driver of 'the insurance request' is less than 18
    or the age of the driver of 'the insurance request' is more than 60
then
    add "The age exceeds the maximum or minimum" to the messages of 'the insuran
    reject 'the insurance response' ;
```

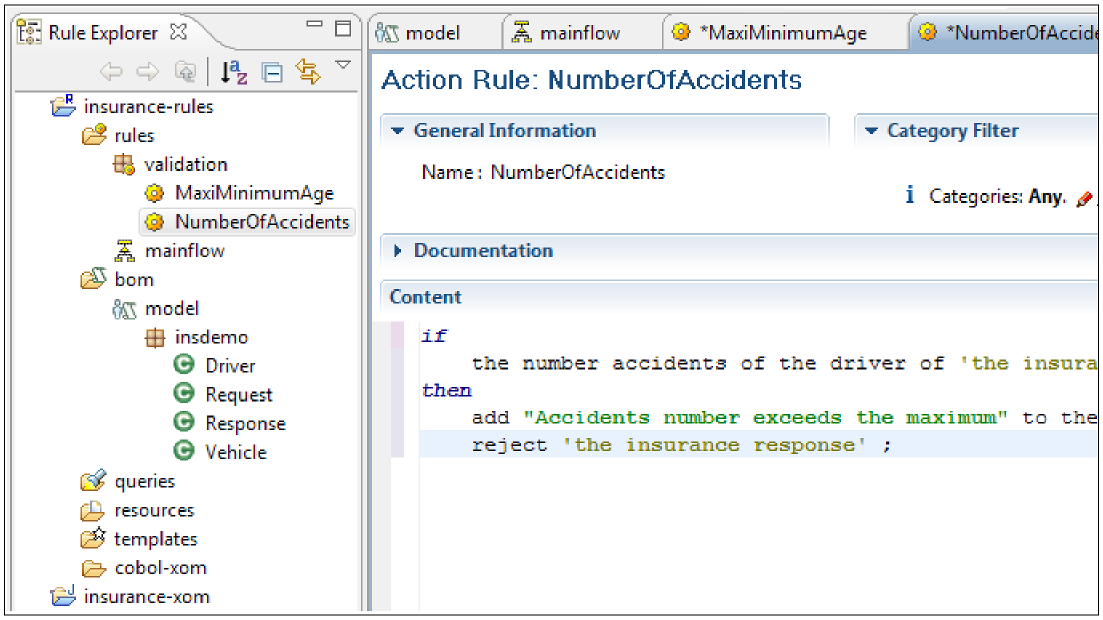
Figure 3-48 Viewing the MaxiMinimumAge rule

2. Create a second action rule by repeating step 1 on page 62 with a name of `NumberOfAccidents`, as shown in Example 3-3.

Example 3-3 Validation with the NumberOfAccidents rule

```
if
    the number accidents of the driver of 'the insurance request' is more than
3
then
    add "Accidents number exceeds the maximum" to the messages of 'the
insurance response' ;
    reject 'the insurance response' ;
```

Figure 3-49 shows the generated action rule.



Rule Explorer | model | mainflow | *MaxiMinimumAge | *NumberOfAccid

Action Rule: NumberOfAccidents

General Information

Name : NumberOfAccidents

Category Filter

Categories: Any.

Documentation

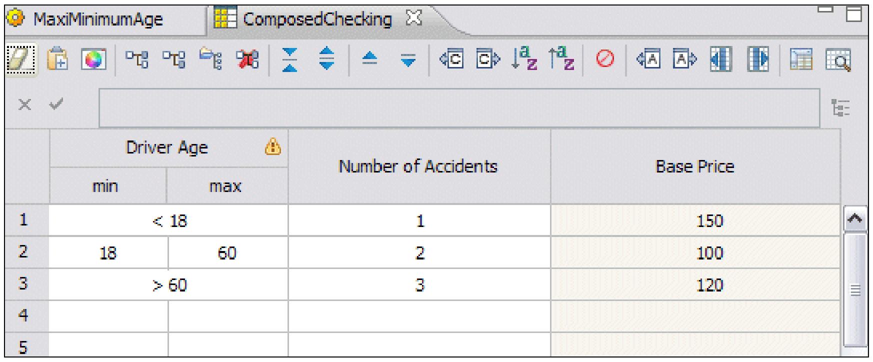
Content

```
if
    the number accidents of the driver of 'the insura
then
    add "Accidents number exceeds the maximum" to the
    reject 'the insurance response' ;
```

Figure 3-49 View the generated rules

3. Save your work.

You can also use a Decision Table or Decision Tree to write decision rules, as shown in the example in Figure 3-50.



MaxiMinimumAge | ComposedChecking

	Driver Age		Number of Accidents	Base Price
	min	max		
1	< 18		1	150
2	18	60	2	100
3	> 60		3	120
4				
5				

Figure 3-50 Decision table example

3.2.9 Preparing the rule execution

This section shows you how to deploy rules to the zRule Execution Server for z/OS (zRES) and how to view the deployed ruleset on the zRule Execution Server web console.

Step 1: Creating a RuleApp project

First, you must create a RuleApp project to contain the rulesets that you want to execute. To create a RuleApp project, follow these steps:

1. In Rule Designer, in the Deploy and Integrate section of the Rule Project Map, click **Create RuleApp project** (Figure 3-51).

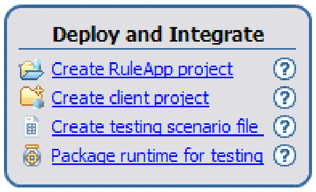
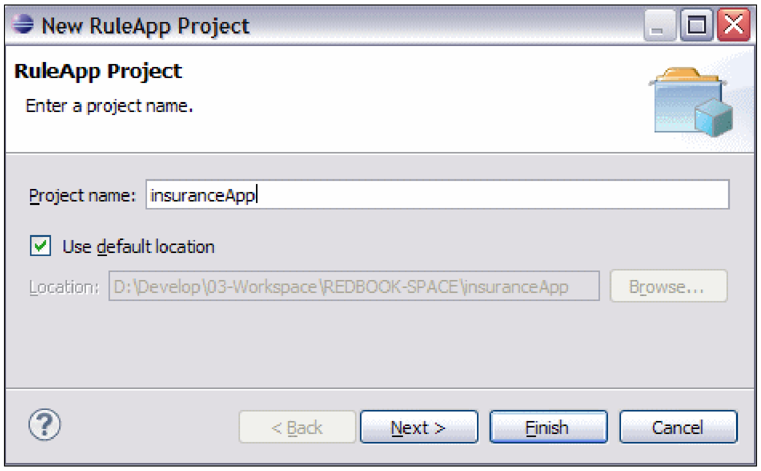


Figure 3-51 Creating a RuleApp project

2. In the New RuleApp Project dialog, enter `insuranceApp` in the Project name field as the name for your RuleApp project. Ensure that **Use default location** is selected (Figure 3-52). Click **Next**.



RuleApp Project

Enter a project name.

Project name: insuranceApp

☑ Use default location

Location: D:\Develop\03-Workspace\REDBOOK-SPACE\insuranceApp

Figure 3-52 Entering the RuleApp name

3. The rule project is listed in the Rule Projects tab, as shown in Figure 3-53. Click **Finish**.

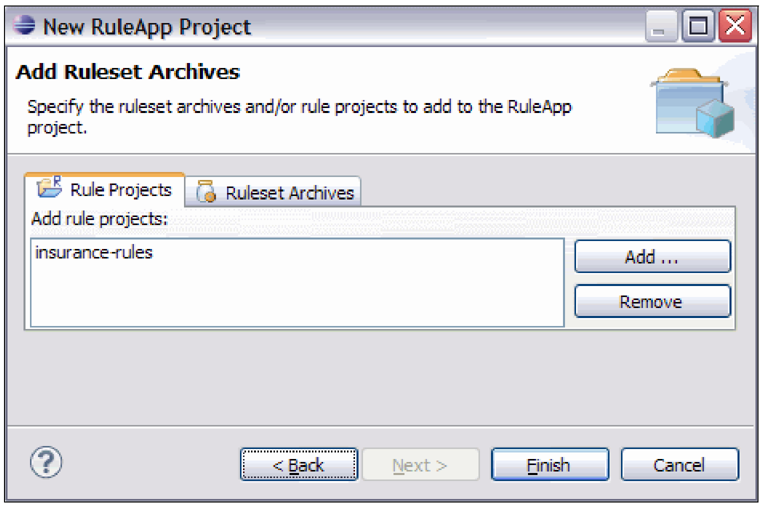


Figure 3-53 Viewing the Rule Projects tab

4. The RuleApp project is created and displayed in the Rule Explorer view, as shown in Figure 3-54.

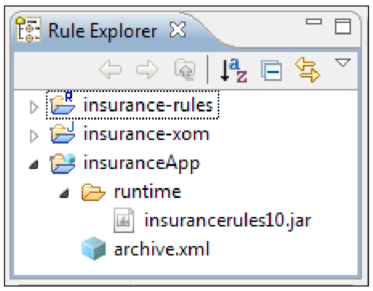


Figure 3-54 Viewing the insuranceApp project

Step 2: Deploying the RuleApp to the zRule Execution Server for z/OS

To be able to execute the ruleset with zRule Execution Server for z/OS (zRES), you must deploy the following artifacts to zRule Execution Server for z/OS:

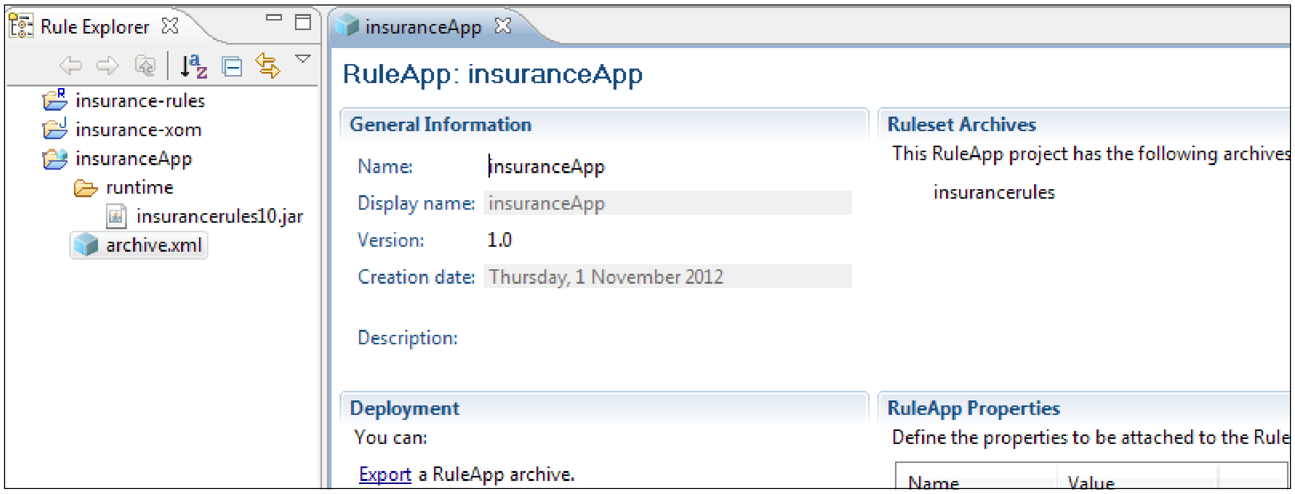
- RuleApps containing the business rules within rulesets
- A JAR resource or library that contains Java classes that are used by the rules

Important: The RuleApp must be deployed to zRule Execution Server for z/OS. Ensure that you start zRule Execution Server for z/OS successfully before you continue this step. For information, see the *Deployment of RuleApps and XOMs* topic in the IBM Operational Decision Manager Version 8.0.1 Information Center:

http://pic.dhe.ibm.com:/infocenter/dmanager/v8r0m1/topic/com.ibm.wodm.zos.dserver.zres/topics/con_zres_deploy_ruleapps_xoms.html

To deploy the XOM, marshaller, and RuleApp, follow these steps:

1. Within the insuranceApp RuleApp project, double-click the `archive.xml` file to open the RuleApp editor (Figure 3-55).



RuleApp: insuranceApp

General Information

Name: insuranceApp
Display name: insuranceApp
Version: 1.0
Creation date: Thursday, 1 November 2012

Description:

Ruleset Archives

This RuleApp project has the following archives

insurancerules

Deployment

You can:

Export a RuleApp archive.

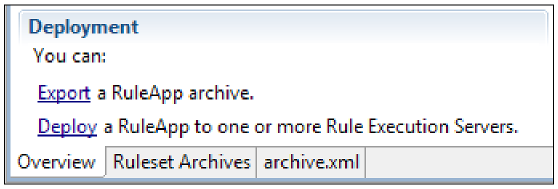
RuleApp Properties

Define the properties to be attached to the Rule

Name	Value

Figure 3-55 RuleApp editor for insuranceApp

2. In the Deployment pane, click **Deploy** to deploy the RuleApp to the Rule Execution Server (Figure 3-56).



Deployment

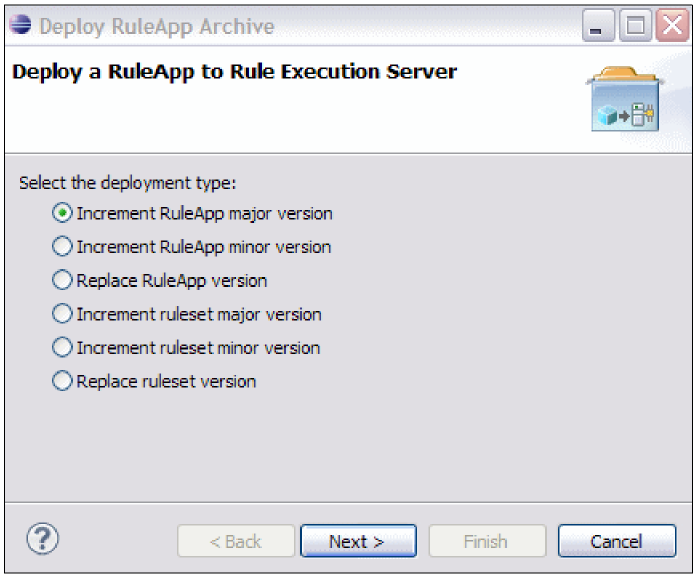
You can:

Export a RuleApp archive.

Deploy a RuleApp to one or more Rule Execution Servers.

Figure 3-56 Deploying the RuleApp

3. In the Deploy RuleApp Archive dialog, select the default option **Increment RuleApp major version** for deployment type (Figure 3-57) and click **Next**.



Deploy RuleApp Archive

Deploy a RuleApp to Rule Execution Server

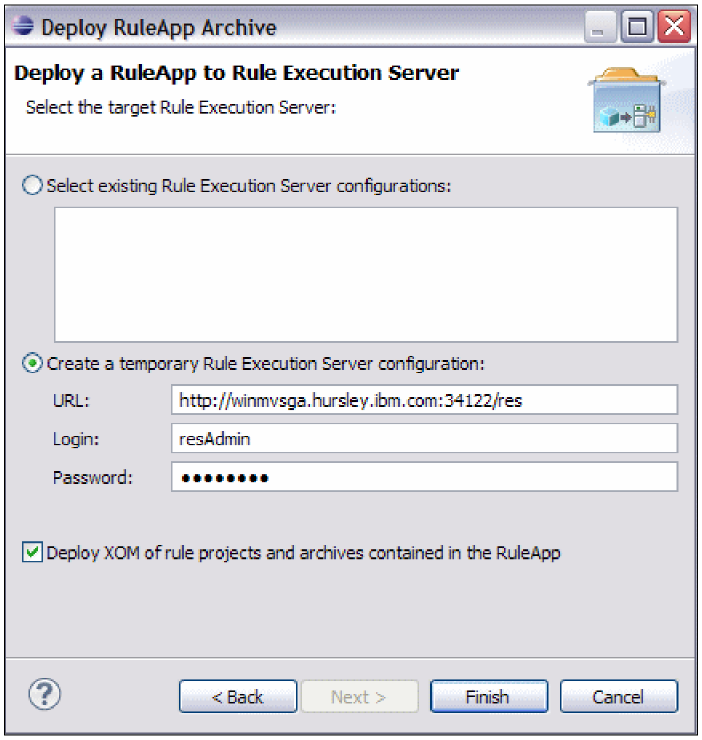
Select the deployment type:

- [✓ Increment RuleApp major version]
- Increment RuleApp minor version
- Replace RuleApp version
- Increment ruleset major version
- Increment ruleset minor version
- Replace ruleset version

Figure 3-57 Selecting the deployment type

4. Select **Create a temporary Rule Execution Server configuration** and enter the following details, as shown in Figure 3-58:
   - URL: `http://<your.server.address>:<PORT>/res`
   - Login: `resAdmin`
   - Password: `resAdmin`

   Select **Deploy XOM of rule projects and archives contained in the RuleApp** and click **Finish**.


Deploy RuleApp Archive
Deploy a RuleApp to Rule Execution Server
Select the target Rule Execution Server:
Select existing Rule Execution Server configurations:
< Back
Next >
Finish
Cancel

Create a temporary Rule Execution Server configuration:

URL: http://winmvsga.hursley.ibm.com:34122/res

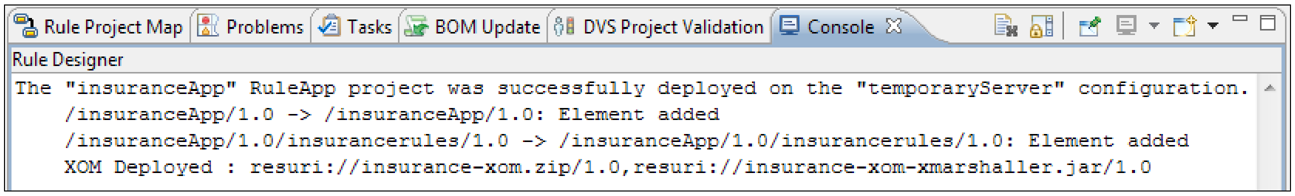
Login: resAdmin

Password: ●●●●●●●●

Deploy XOM of rule projects and archives contained in the RuleApp

Figure 3-58 Configuring the RuleApp deployment

In the Console tab, you can see the confirmation that the project has been deployed, as shown in Figure 3-59. The artifacts are now deployed to the zRule Execution Server for z/OS server.



```
The "insuranceApp" RuleApp project was successfully deployed on the "temporaryServer" configuration.
    /insuranceApp/1.0 -> /insuranceApp/1.0: Element added
    /insuranceApp/1.0/insurancerules/1.0 -> /insuranceApp/1.0/insurancerules/1.0: Element added
    XOM Deployed : resuri://insurance-xom.zip/1.0,resuri://insurance-xom-xmarshaller.jar/1.0
```

Figure 3-59 Deploying the RuleApp confirmation

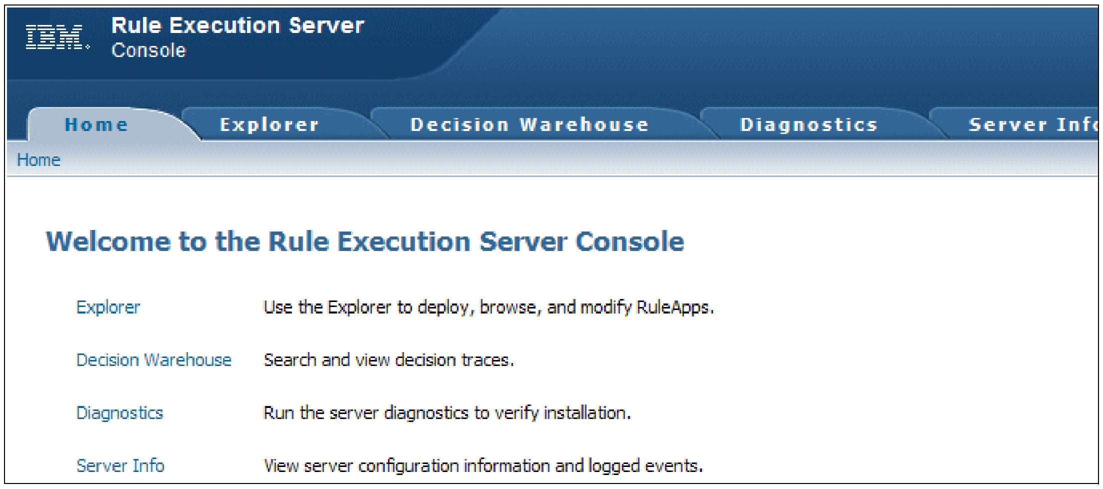
Step 3: Viewing deployed rule artifacts in the Rule Execution Server console

You can log in to the zRule Execution Server console and use the Navigator pane to view the deployed RuleApp and XOM. To view your deployed artifacts, follow these steps:

1. In a web browser, open the web console for zRule Execution Server for z/OS by using the following URL:

   `http://<your.server.address>:<PORT>/res`

2. At the login prompt for the Rule Execution Server console, enter the following login details:
   – Login: `resAdmin`
   – Password: `resAdmin`
3. On the Rule Execution Server, click **Explorer** (Figure 3-60).

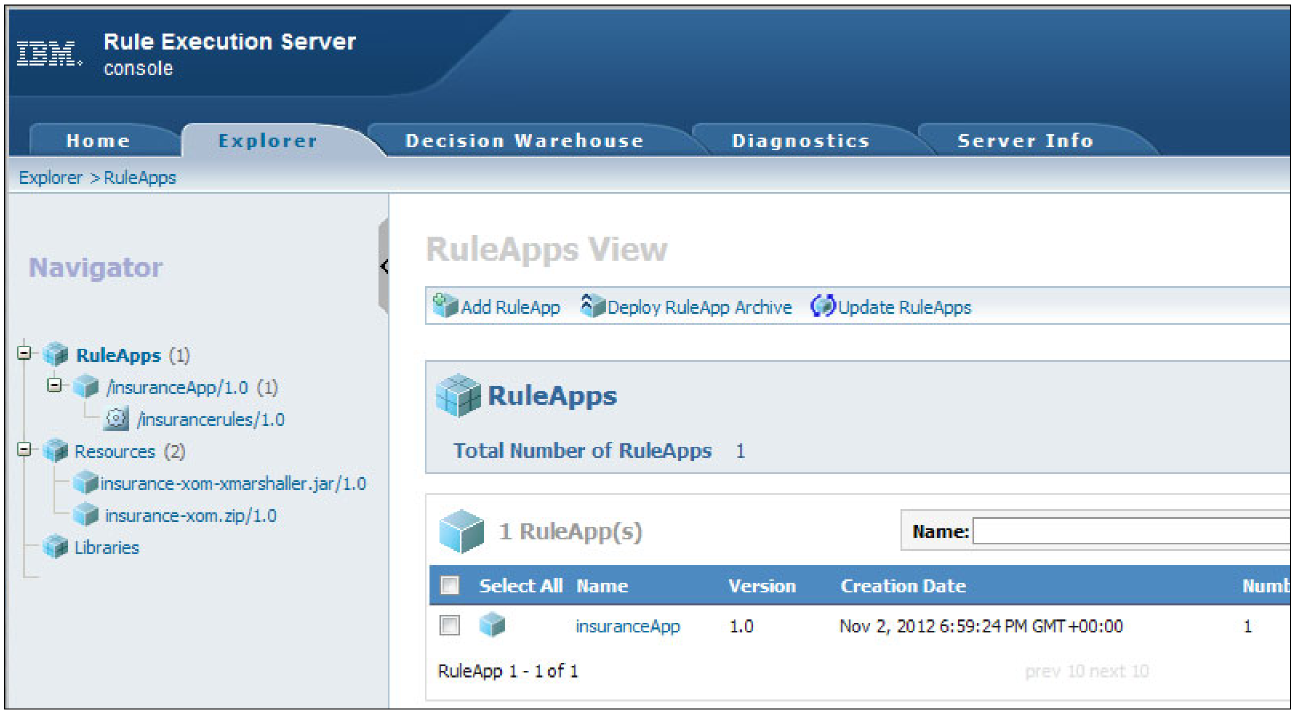


Welcome to the Rule Execution Server Console

Explorer	Use the Explorer to deploy, browse, and modify RuleApps.
Decision Warehouse	Search and view decision traces.
Diagnostics	Run the server diagnostics to verify installation.
Server Info	View server configuration information and logged events.

Figure 3-60 Exploring the rule project

4. In the Navigator pane, click **RuleApps** to view the deployed RuleApp (Figure 3-61).



Total Number of RuleApps 1

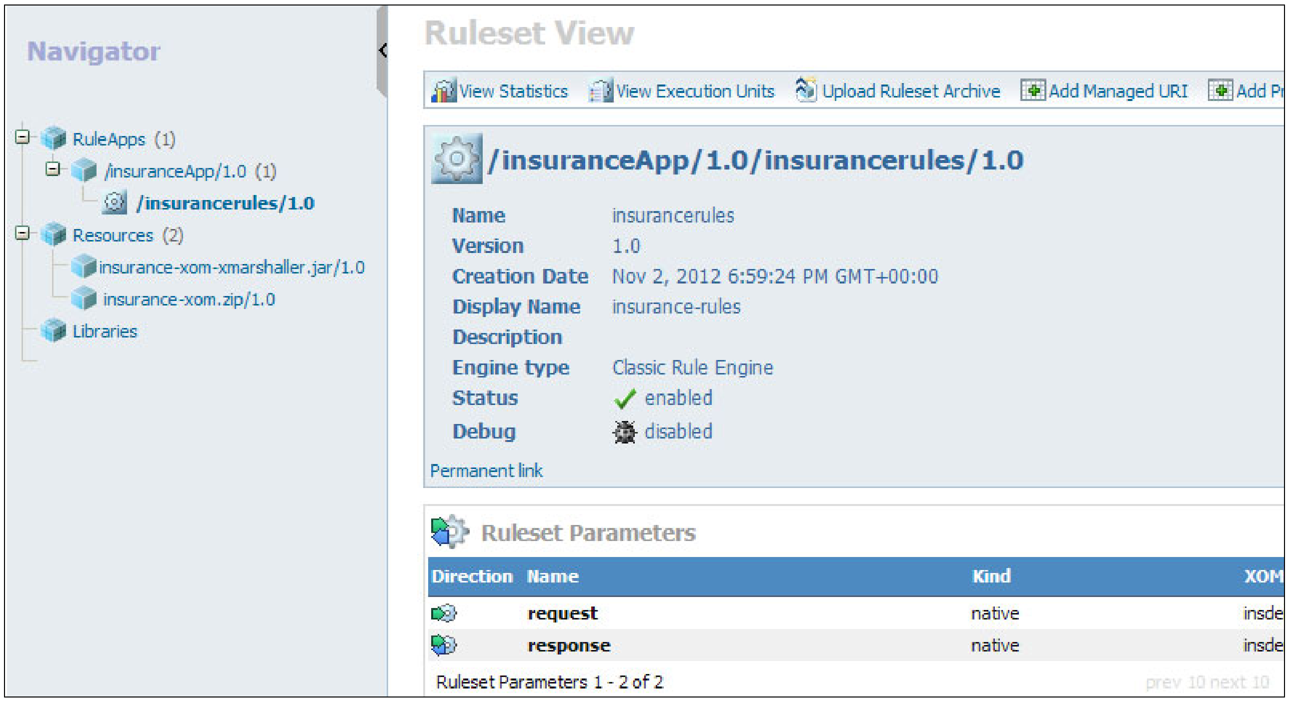
1 RuleApp(s)

Select All	Name	Version	Creation Date	Numb
☐	insuranceApp	1.0	Nov 2, 2012 6:59:24 PM GMT+00:00	1

RuleApp 1 - 1 of 1 prev 10 next 10

Figure 3-61 Viewing the deployed ruleset

5. With the RulesApps tree fully expanded, click the **/insurancerules/1.0** ruleset to see the Ruleset View (Figure 3-62).



/insuranceApp/1.0/insurancerules/1.0

Name	insurancerules
Version	1.0
Creation Date	Nov 2, 2012 6:59:24 PM GMT+00:00
Display Name	insurance-rules
Description	
Engine type	Classic Rule Engine
Status	✓ enabled
Debug	disabled

Permanent link

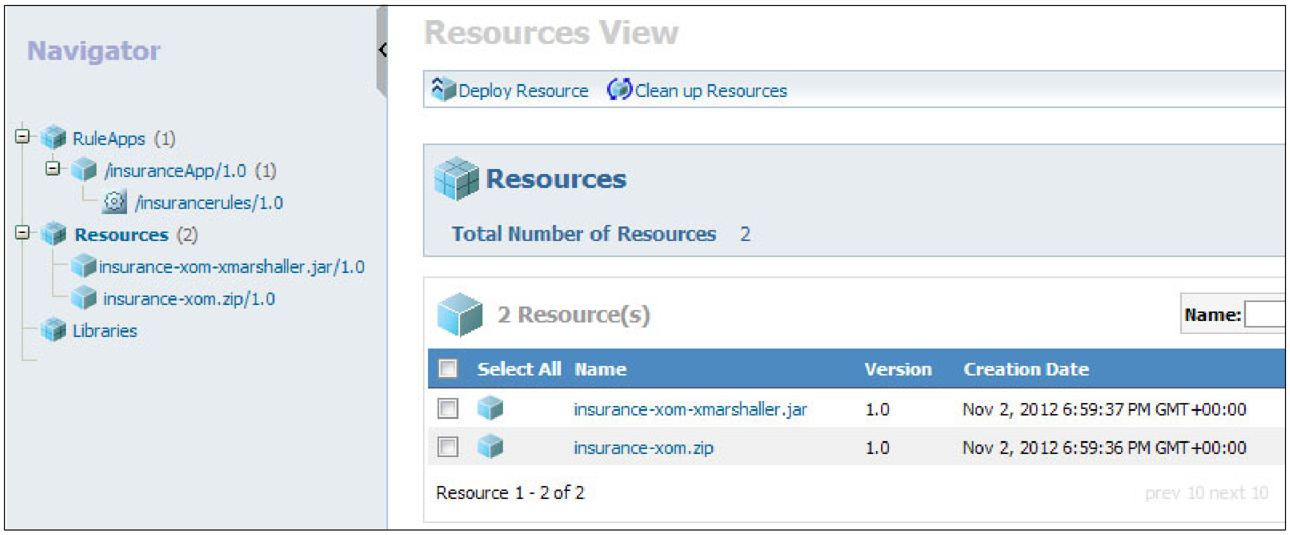
Ruleset Parameters

Direction	Name	Kind	XOM
	request	native	insde
	response	native	insde

Ruleset Parameters 1 - 2 of 2 prev 10 next 10

Figure 3-62 Viewing the deployed ruleset view

6. In the Navigator pane, click **Resources** to view the deployed XOM and the marshaller file (Figure 3-63).



Resources

Total Number of Resources 2

2 Resource(s) Name:

Select All	Name	Version	Creation Date
☐	insurance-xom-xmarshaller.jar	1.0	Nov 2, 2012 6:59:37 PM GMT+00:00
☐	insurance-xom.zip	1.0	Nov 2, 2012 6:59:36 PM GMT+00:00

Resource 1 - 2 of 2 prev 10 next 10

Figure 3-63 Viewing deployed Java XOM and marshaller XOM

3.2.10 Building a COBOL application for rule execution

To execute the rules, you call the ruleset from the COBOL application. You can use the zRule COBOL stub API to invoke the rule execution in a running instance of zRule Execution Server for z/OS.

To build the COBOL application, follow these steps:

1. Include the required copybooks for the zRule COBOL stub API:

   ```
   01 WS-REASON-CODES.
   COPY HBRC.
   COPY HBRWS.
   ```

2. Specify the ruleset path to initialize the values that are passed to zRule Execution Server for z/OS:

   ```
   * ruleset path from the zRules Execution Server
     MOVE "/insuranceApp/insurancerules"  TO HBRA-CONN-RULEAPP-PATH
   ```

3. Configure the ruleset parameter:

   – Set the name of the parameter:

     ```
     MOVE 'request' TO HBRA-RA-PARAMETER-NAME(1)
     ```

   – Set the length of the parameter:

     ```
     MOVE LENGTH OF REQUEST TO HBRA-RA-DATA-LENGTH(1)
     ```

   – Set the address of the parameter:

     ```
     SET HBRA-RA-DATA-ADDRESS(1) TO ADDRESS OF REQUEST
     ```

4. Connect to zRule Execution Server for z/OS:

   ```
   CALL 'HBRCONN' USING HBRA-CONN-AREA.
   ```

5. Execute the ruleset:

   ```
   CALL 'HBRCONN' USING HBRA-CONN-AREA.
   ```

6. Disconnect from zRule Execution Server for z/OS:

   ```
   CALL 'HBRDISC' USING HBRA-CONN-AREA.
   ```

COBOL application sample

Example 3-4 includes a sample COBOL application that you can use to call the rules that you designed in the insurance-rules project.

Additional resources: You can find the `INSMAIN.cbl` application sample in the additional information that is included in this book in the `code/Chapter3/CopybookBased` directory. See Appendix C, "Additional material" on page 335.

Example 3-4 COBOL application sample to call the rules on zRule Execution Server for z/OS

```
IDENTIFICATION DIVISION.
       PROGRAM-ID. "INSMAIN".
       ENVIRONMENT DIVISION.
       DATA DIVISION.
       WORKING-STORAGE SECTION.
           COPY INSDEMO.
       01 WS-REASON-CODES.
           COPY HBRC.
           COPY HBRWS.
```

```
 01 WS-MESSAGE-IDX      PIC 9(2).
 01 WS-MAX-TABLE-LEN    PIC 9(18).

 PROCEDURE DIVISION.
* Init ruleset parameter data
     MOVE 'John'            TO FIRST-NAME
     MOVE 'Smith'           TO LAST-NAME
     MOVE 'XA123456'        TO ZIPCODE
     MOVE 123456            TO HOUSE-NUM
     MOVE 17                TO AGE
     MOVE '20110908'        TO LIC-DATE
     MOVE 'F'               TO LIC-STATUS
     MOVE 4                 TO NUMBER-ACCIDENTS
     MOVE 'F'               TO APPROVED
     MOVE 100               TO BASE-PRICE
     MOVE 0                 TO MSG-COUNT
* Move ruleset parameters to table HBRA-RA-PARMETERS
     MOVE ZERO              TO HBRA-CONN-RETURN-CODES
     MOVE LOW-VALUES        TO HBRA-RA-PARMETERS
     MOVE "/insuranceApp/insurancerules"
                            TO HBRA-CONN-RULEAPP-PATH
* Parameter Borrower
     MOVE LOW-VALUES        TO HBRA-RA-PARMETERS.
     MOVE 'request'         TO HBRA-RA-PARAMETER-NAME(1)
     MOVE LENGTH OF REQUEST  TO HBRA-RA-DATA-LENGTH(1)
     SET HBRA-RA-DATA-ADDRESS(1)
                            TO ADDRESS OF REQUEST
* Parameter Loan
     MOVE 'response'        TO HBRA-RA-PARAMETER-NAME(2)
     MOVE LENGTH OF RESPONSE TO HBRA-RA-DATA-LENGTH(2)
* For ODO Table, the length represents the max length.
     COMPUTE WS-MAX-TABLE-LEN = LENGTH OF Messages * 100
     ADD WS-MAX-TABLE-LEN    TO HBRA-RA-DATA-LENGTH(2)
     SET HBRA-RA-DATA-ADDRESS(2)
                            TO ADDRESS OF RESPONSE
* Get connection to rule execution server
     CALL 'HBRCONN' USING HBRA-CONN-AREA.
     IF HBRA-CONN-COMPLETION-CODE NOT EQUAL HBR-CC-OK
         DISPLAY "connect zRules failed"
         DISPLAY "CC code " HBRA-CONN-COMPLETION-CODE
         DISPLAY "RC code " HBRA-CONN-REASON-CODE
         DISPLAY "Message " HBRA-RESPONSE-MESSAGE
     ELSE
         DISPLAY 'connect zRules successful'
     END-IF
* Invoke rule execution server
     CALL 'HBRRULE' USING HBRA-CONN-AREA
     IF HBRA-CONN-COMPLETION-CODE NOT EQUAL HBR-CC-OK
         DISPLAY "invoke zRules failed"
         DISPLAY "CC code " HBRA-CONN-COMPLETION-CODE
         DISPLAY "RC code " HBRA-CONN-REASON-CODE
         DISPLAY "Message " HBRA-RESPONSE-MESSAGE
     ELSE
         DISPLAY 'invoke zRules successful'
     END-IF
     * Get disconnect to rule execution server
           CALL 'HBRDISC' USING HBRA-CONN-AREA
     * Display result
           DISPLAY         "********** EXECUTION RESULT *********"
           DISPLAY         "DRIVER NAME: " FIRST-NAME
           DISPLAY         "RESPONSE APPROVED: " APPROVED
           IF approved = "F"
              DISPLAY      "Reject messages:"
              PERFORM VARYING WS-MESSAGE-IDX FROM 1 BY 1
                              UNTIL WS-MESSAGE-IDX > MSG-COUNT
                 DISPLAY messages (WS-MESSAGE-IDX)
              END-PERFORM
           END-IF
           DISPLAY         "*************************************"
     STOP RUN.
```

Rule execution

You can compile and run the COBOL application on z/OS. In the COBOL application sample that is shown in Example 3-4 on page 71, you hardcoded the following input values:

- AGE: `17`
- NUMBER-ACCIDENTS: `4`

Example 3-5 shows the results after the rule execution.

Example 3-5 Rule execution result

```
********** EXECUTION RESULT *********
DRIVER NAME: John
RESPONSE APPROVED: F
Reject messages:
Accidents number exceeds the maximum
The age exceeds the maximum or minimum
*************************************
```

3.3 Getting started from an existing rule project

This section provides guidance about how to share business rules from an existing Java-based rule project to a COBOL application on z/OS.

3.3.1 Scenario overview

In this scenario, an insurance company has an existing business rule application to perform user validation for an insurance application. The rule projects, which the company currently uses, contain a BOM that is based on a Java XOM. The company deploys the rules to the Rule Execution Server in a distributed environment.

The company now wants to share the Java rule projects with COBOL applications that run on z/OS and to manage the changes that are made to these rules. To share rules with COBOL applications, the company must add the necessary COBOL structures to the BOM and then generate a COBOL copybook. With these structures in the rule project, the company can then deploy the rules application to zRule Execution Server for z/OS so that the COBOL application can call the rulesets and execute the rules.

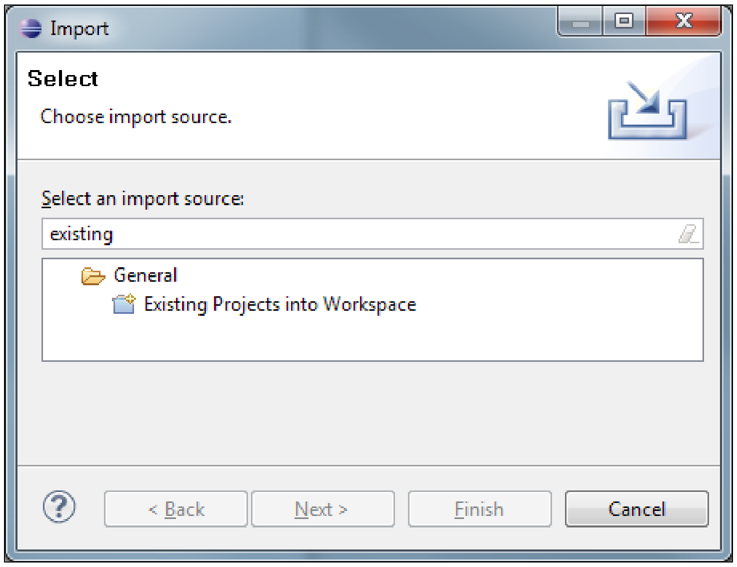
This section provides an existing rule project, `sharinginsurance-rules`, that has a BOM that is generated from a Java XOM (`sharinginsurance-xom`). It also uses a RuleApp project (`sharinginsuranceApp`) that is used for rule deployment to the runtime environment.

You can import the existing rule project from the source code that is delivered with this book. See Appendix C, “Additional material” on page 335 for details.

To import the example rule project, follow these steps:

1. From the Rule Explorer view, right-click and then select **Import** from the menu.
2. In the Import dialog, select **Existing Projects into Workspace**, as shown in Figure 3-64. Click **Next**.

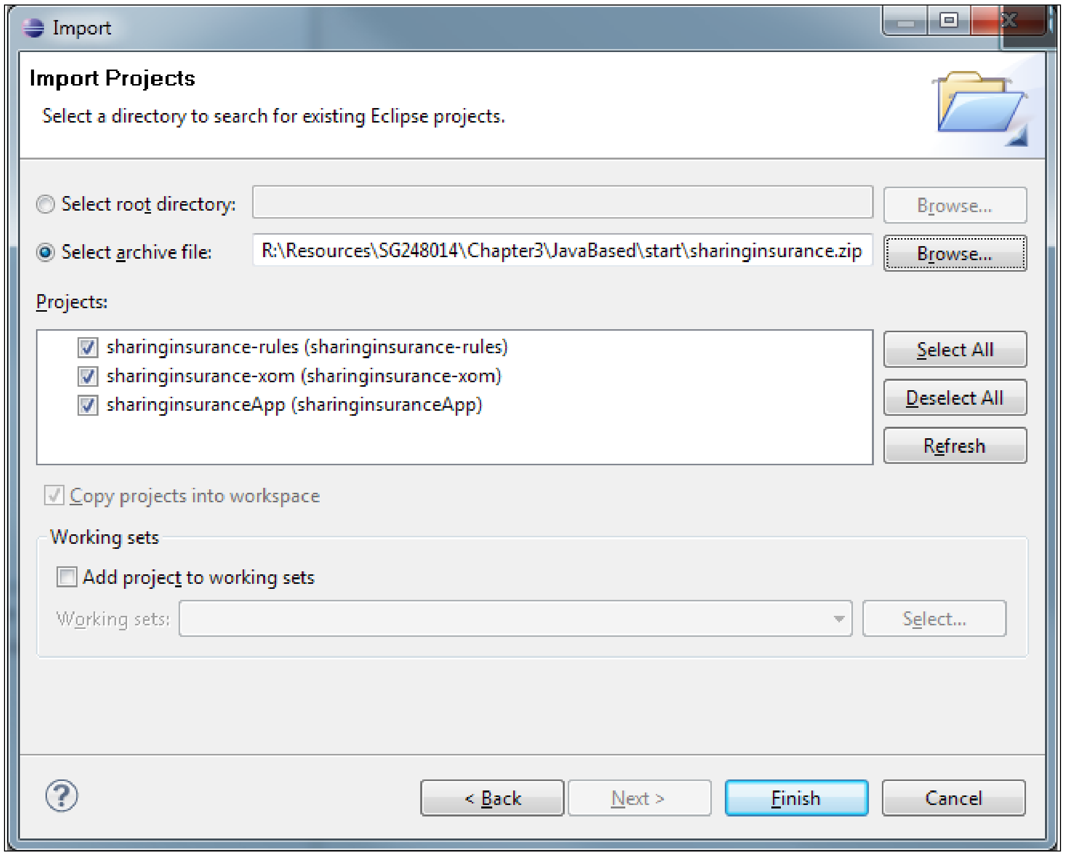
Tip: You can quickly reduce the list of import sources by typing part of the name of the import source in the Select an import source field.



Select an import source: existing

Figure 3-64 Importing existing projects

3. In the Import Projects dialog, select the **Select Archive File** option and browse to the `sharinginsurance.zip` file. Select all three projects, as shown in Figure 3-65, and click **Finish**.


Import

Import Projects

Select a directory to search for existing Eclipse projects.

( ) Select root directory:

(●) Select archive file: R:\Resources\SG248014\Chapter3\JavaBased\start\sharinginsurance.zip

Projects:

- ☑ sharinginsurance-rules (sharinginsurance-rules)
- ☑ sharinginsurance-xom (sharinginsurance-xom)
- ☑ sharinginsuranceApp (sharinginsuranceApp)

☑ Copy projects into workspace

Working sets

☐ Add project to working sets

Working sets:

Figure 3-65 Importing the insurance projects

4. Figure 3-66 shows the existing rule project structure in Rule Designer.

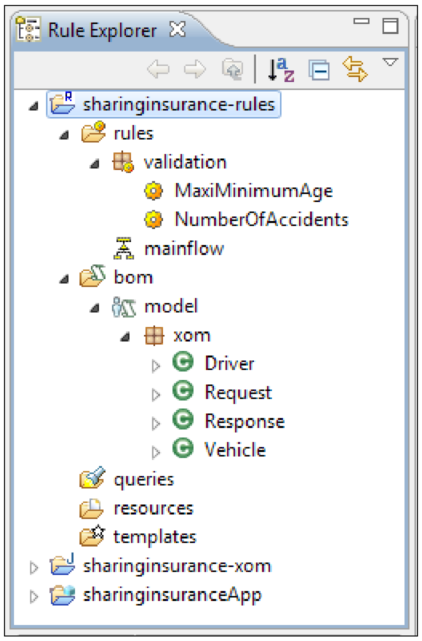


Figure 3-66 Existing rule project structure

3.3.2 Generating a copybook from the BOM

You use the COBOL Enabled BOM feature of Rule Designer to generate a copybook from the BOM in the existing rule project.

Use the following configuration for the BOM:

- Specify each Java class type that you want to use as top-level data items in the copybook.
- Enter a name for the runtime marshaller project and package, which are created during the copybook generation.

To configure the BOM for copybook generation, follow these steps:

1. In the Rule Explorer, right-click the **sharinginsurance-rules** rule project, and select **Properties** → **COBOL Management** → **COBOL Enabled BOM** (Figure 3-67). Click **Add**.

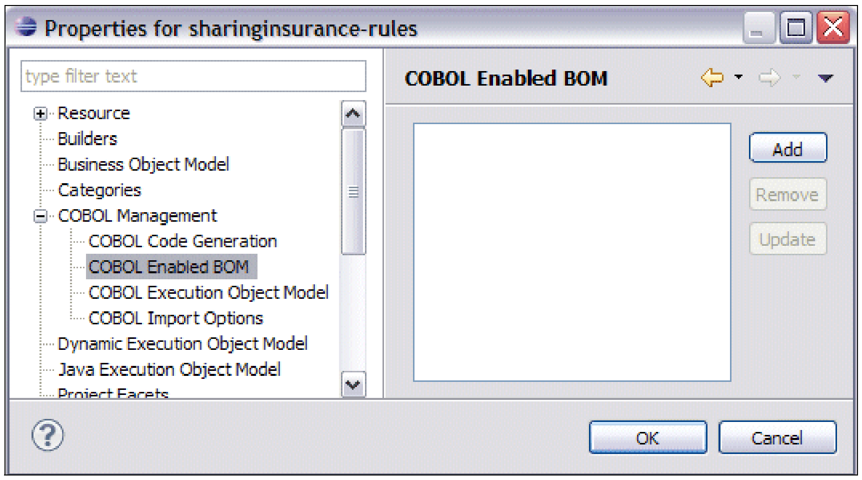


Figure 3-67 Navigating to the COBOL enabled BOM

2. In the Select BOM entry dialog, select **model**, and click **OK** (Figure 3-68).

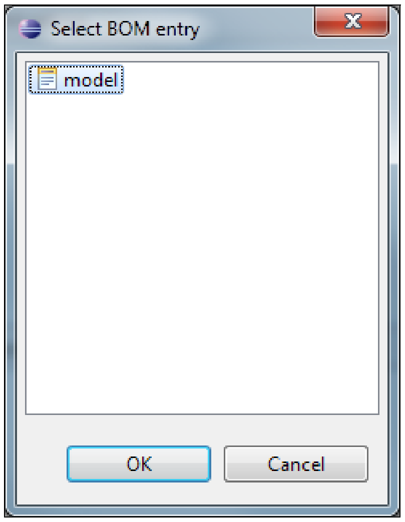
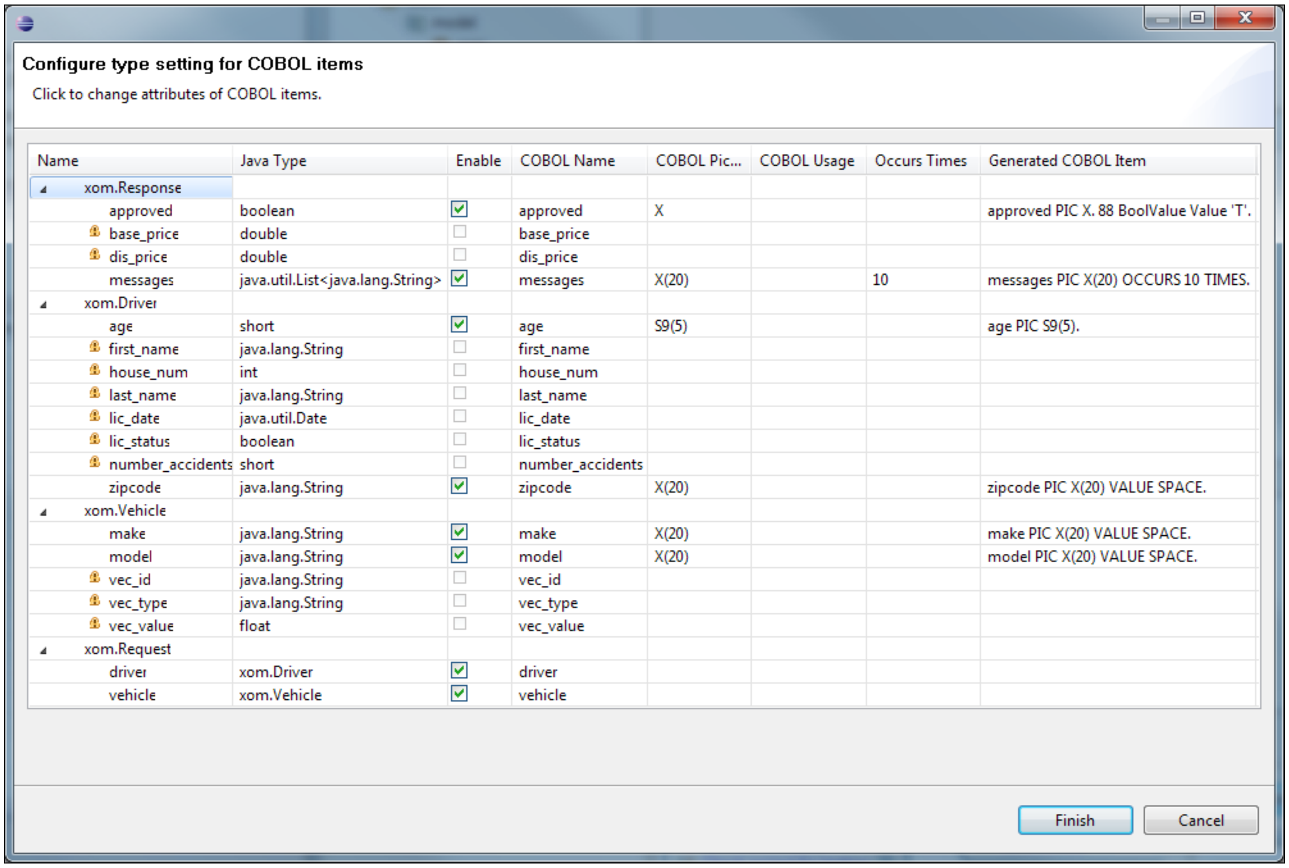


Figure 3-68 Selecting the BOM model

3. In the Resource Configuration section, accept the default names for the runtime Marshaller Project and Marshaller Package. Click **Next**.

A table shows the proposed mapping between the Java structures in the BOM and the COBOL structures, as shown in Figure 3-69.

Important: Several yellow warning triangles show on certain lines. They are present because the Java attribute name is not a valid COBOL name. Those fields have not been enabled.



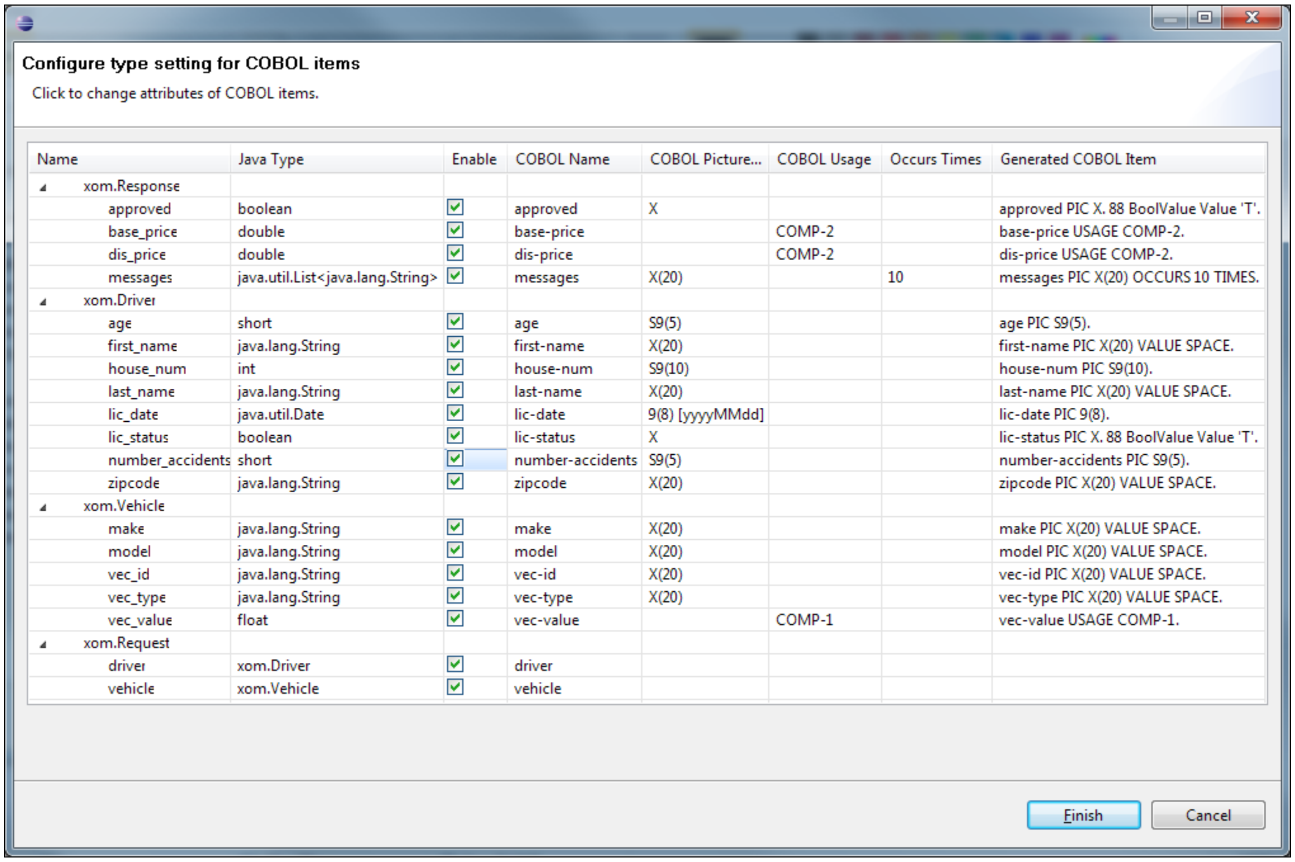
Configure type setting for COBOL items

Click to change attributes of COBOL items.

Name	Java Type	Enable	COBOL Name	COBOL Pic...	COBOL Usage	Occurs Times	Generated COBOL Item
xom.Response							
approved	boolean	✓	approved	X			approved PIC X. 88 BoolValue Value 'T'.
base_price	double	☐	base_price				
dis_price	double	☐	dis_price				
messages	java.util.List<java.lang.String>	✓	messages	X(20)		10	messages PIC X(20) OCCURS 10 TIMES.
xom.Driver							
age	short	✓	age	S9(5)			age PIC S9(5).
first_name	java.lang.String	☐	first_name				
house_num	int	☐	house_num				
last_name	java.lang.String	☐	last_name				
lic_date	java.util.Date	☐	lic_date				
lic_status	boolean	☐	lic_status				
number_accidents	short	☐	number_accidents				
zipcode	java.lang.String	✓	zipcode	X(20)			zipcode PIC X(20) VALUE SPACE.
xom.Vehicle							
make	java.lang.String	✓	make	X(20)			make PIC X(20) VALUE SPACE.
model	java.lang.String	✓	model	X(20)			model PIC X(20) VALUE SPACE.
vec_id	java.lang.String	☐	vec_id				
vec_type	java.lang.String	☐	vec_type				
vec_value	float	☐	vec_value				
xom.Request							
driver	xom.Driver	✓	driver				
vehicle	xom.Vehicle	✓	vehicle				

Figure 3-69 Configuring the BOM to COBOL type mapping

4. For each of the fields with a warning, amend the COBOL name to make it suitable, for example, by converting each underscore (_) character to a dash (-) character (Figure 3-70).



Configure type setting for COBOL items

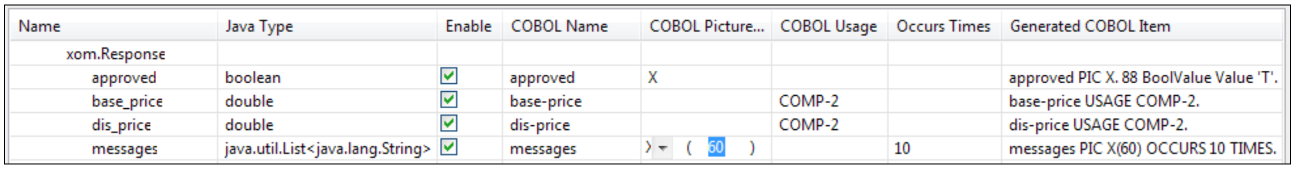
Click to change attributes of COBOL items.

Name	Java Type	Enable	COBOL Name	COBOL Picture...	COBOL Usage	Occurs Times	Generated COBOL Item
xom.Response							
approved	boolean	✓	approved	X			approved PIC X. 88 BoolValue Value 'T'.
base_price	double	✓	base-price		COMP-2		base-price USAGE COMP-2.
dis_price	double	✓	dis-price		COMP-2		dis-price USAGE COMP-2.
messages	java.util.List<java.lang.String>	✓	messages	X(20)		10	messages PIC X(20) OCCURS 10 TIMES.
xom.Driver							
age	short	✓	age	S9(5)			age PIC S9(5).
first_name	java.lang.String	✓	first-name	X(20)			first-name PIC X(20) VALUE SPACE.
house_num	int	✓	house-num	S9(10)			house-num PIC S9(10).
last_name	java.lang.String	✓	last-name	X(20)			last-name PIC X(20) VALUE SPACE.
lic_date	java.util.Date	✓	lic-date	9(8) [yyyyMMdd]			lic-date PIC 9(8).
lic_status	boolean	✓	lic-status	X			lic-status PIC X. 88 BoolValue Value 'T'.
number_accidents	short	✓	number-accidents	S9(5)			number-accidents PIC S9(5).
zipcode	java.lang.String	✓	zipcode	X(20)			zipcode PIC X(20) VALUE SPACE.
xom.Vehicle							
make	java.lang.String	✓	make	X(20)			make PIC X(20) VALUE SPACE.
model	java.lang.String	✓	model	X(20)			model PIC X(20) VALUE SPACE.
vec_id	java.lang.String	✓	vec-id	X(20)			vec-id PIC X(20) VALUE SPACE.
vec_type	java.lang.String	✓	vec-type	X(20)			vec-type PIC X(20) VALUE SPACE.
vec_value	float	✓	vec-value		COMP-1		vec-value USAGE COMP-1.
xom.Request							
driver	xom.Driver	✓	driver				
vehicle	xom.Vehicle	✓	vehicle				

Figure 3-70 Changing the COBOL names

5. Click the **COBOL Picture** field for the messages item of the xom.Response class and change the default length for messages from X(20) to X(60), as shown in Figure 3-71. Click **Finish**.

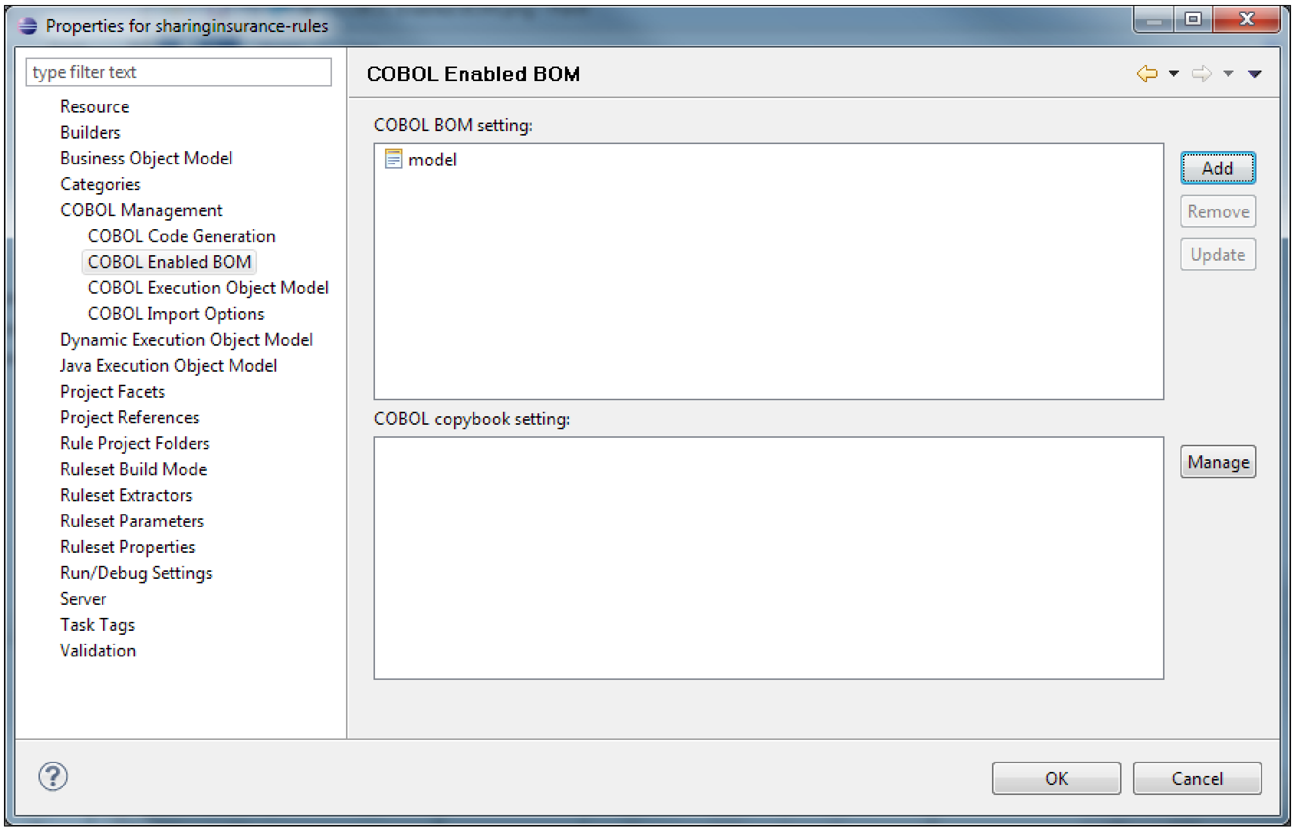
Important: The default mapping from Java String to COBOL Picture length is 20. You adjust this value per rule project. In this scenario, the sharinginsurance-rules project uses a COBOL Picture length mapping value of 60, as required by the real reject message in rules.



Name	Java Type	Enable	COBOL Name	COBOL Picture...	COBOL Usage	Occurs Times	Generated COBOL Item
xom.Response							
approved	boolean	✓	approved	X			approved PIC X. 88 BoolValue Value 'T'.
base_price	double	✓	base-price		COMP-2		base-price USAGE COMP-2.
dis_price	double	✓	dis-price		COMP-2		dis-price USAGE COMP-2.
messages	java.util.List<java.lang.String>	✓	messages	X ▾ (60)		10	messages PIC X(60) OCCURS 10 TIMES.

Figure 3-71 Changing the default mapping of the message item of the xom.Response class

6. The BOM model is now listed as a COBOL enabled BOM, as shown in Figure 3-72. Click **Manage**.



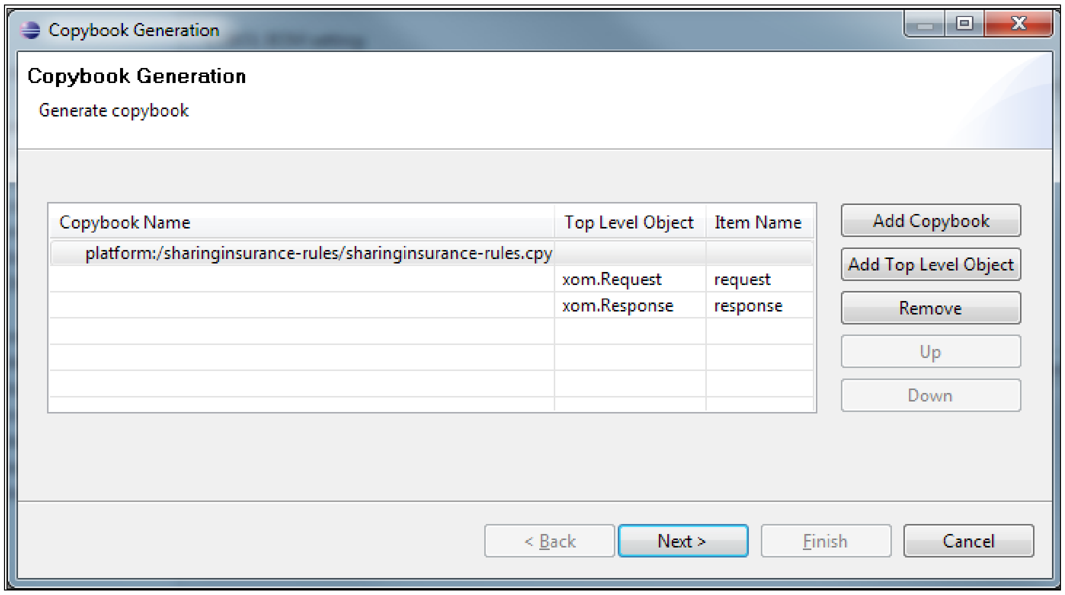
COBOL BOM setting:

model

COBOL copybook setting:

Figure 3-72 COBOL enabled BOM

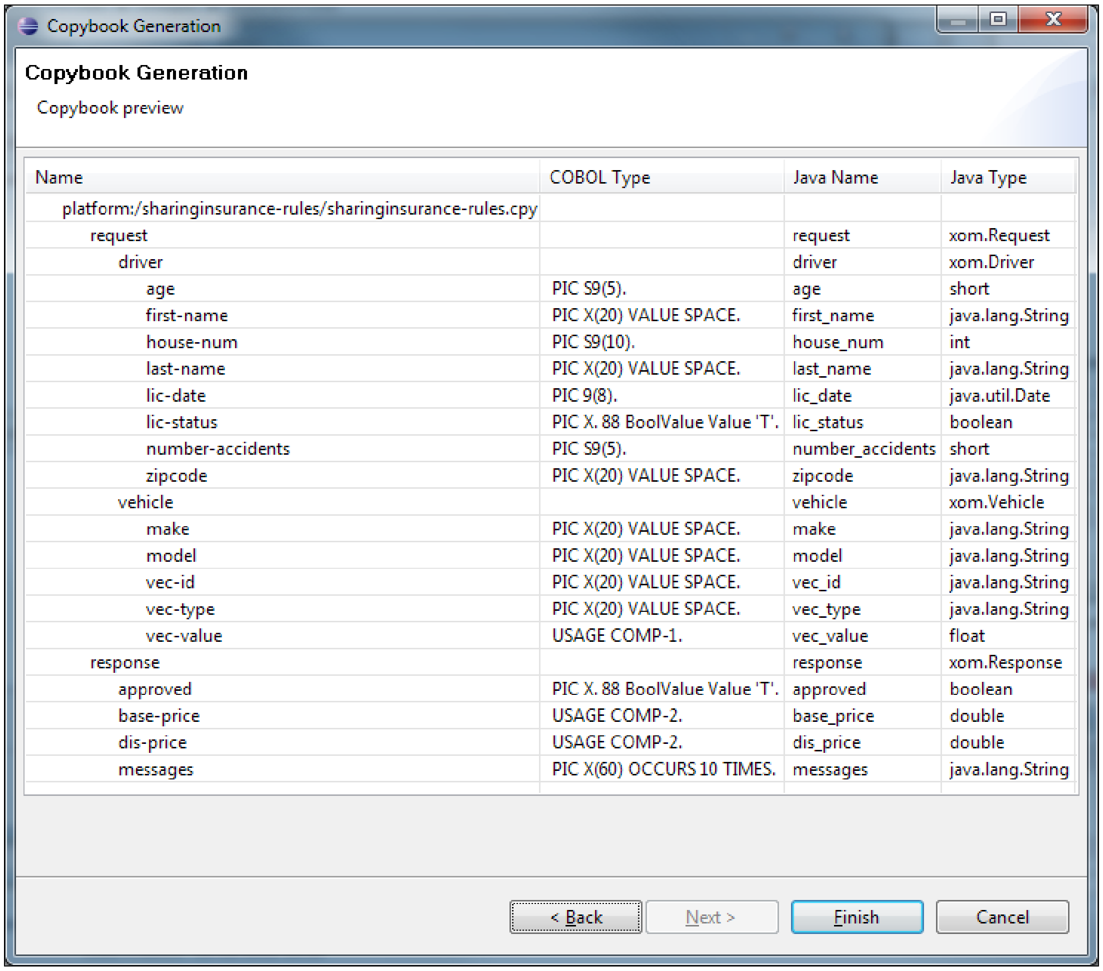
7. In the Copybook Generation dialog (Figure 3-73), review the information and click **Next**.



Copybook Name	Top Level Object	Item Name
platform:/sharinginsurance-rules/sharinginsurance-rules.cpy		
	xom.Request	request
	xom.Response	response

Figure 3-73 Copybook generation information

8. You see the Copybook Generation preview dialog, as shown in Figure 3-74. Review the information and click **Finish**.



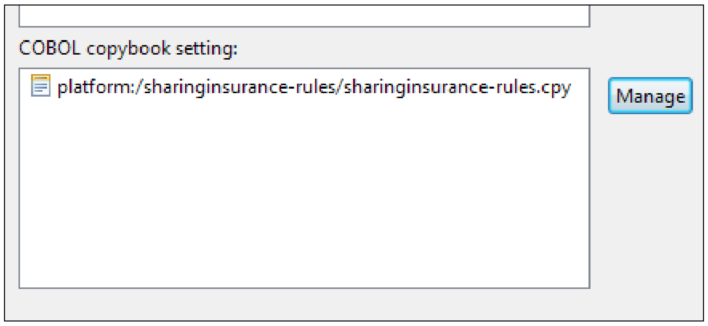
Copybook Generation

Copybook preview

Name	COBOL Type	Java Name	Java Type
platform:/sharinginsurance-rules/sharinginsurance-rules.cpy			
request		request	xom.Request
driver		driver	xom.Driver
age	PIC S9(5).	age	short
first-name	PIC X(20) VALUE SPACE.	first_name	java.lang.String
house-num	PIC S9(10).	house_num	int
last-name	PIC X(20) VALUE SPACE.	last_name	java.lang.String
lic-date	PIC 9(8).	lic_date	java.util.Date
lic-status	PIC X. 88 BoolValue Value 'T'.	lic_status	boolean
number-accidents	PIC S9(5).	number_accidents	short
zipcode	PIC X(20) VALUE SPACE.	zipcode	java.lang.String
vehicle		vehicle	xom.Vehicle
make	PIC X(20) VALUE SPACE.	make	java.lang.String
model	PIC X(20) VALUE SPACE.	model	java.lang.String
vec-id	PIC X(20) VALUE SPACE.	vec_id	java.lang.String
vec-type	PIC X(20) VALUE SPACE.	vec_type	java.lang.String
vec-value	USAGE COMP-1.	vec_value	float
response		response	xom.Response
approved	PIC X. 88 BoolValue Value 'T'.	approved	boolean
base-price	USAGE COMP-2.	base_price	double
dis-price	USAGE COMP-2.	dis_price	double
messages	PIC X(60) OCCURS 10 TIMES.	messages	java.lang.String

Figure 3-74 Copybook generation preview

9. Returning to the COBOL Enabled BOM dialog, you see that a new copybook has been created in the COBOL copybook setting section (Figure 3-75). Click **OK**.



COBOL copybook setting:

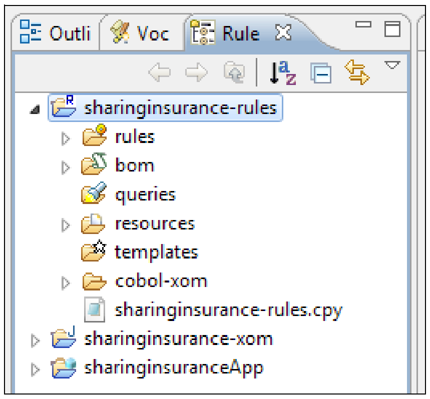
platform:/sharinginsurance-rules/sharinginsurance-rules.cpy

Figure 3-75 COBOL copybook settings

You can view the generated copybook and marshaller project, as shown in Figure 3-76:

- `INSSHAR.cpy` generated copybook
- `model-marshaller` generated marshaller project

Important: Do not change the generated copybook. When a change occurs to the BOM of the rule project, use the COBOL enabled BOM feature to update the copybook.



```
sharinginsurance-rules
    rules
    bom
    queries
    resources
    templates
    cobol-xom
    sharinginsurance-rules.cpy
sharinginsurance-xom
sharinginsuranceApp
```

Figure 3-76 Generated copybook

3.3.3 Deploying rule artifacts to zRule Execution Server for z/OS

To execute a ruleset with zRule Execution Server for z/OS, you must deploy the rule project and the Java XOM to a zRule Execution Server for z/OS. The deployment process is the same as the process that is described in 3.2, “Getting started from a COBOL copybook” on page 35.

Important: Ensure that you start a zRule Execution Server for z/OS successfully before you attempt to deploy rules.

To deploy rule artifacts to zRule Execution Server for z/OS, follow these steps:

1. Deploy the sharinginsuranceApp by using one of the following options:
   – Opening the project and double-clicking the **archive.xml** file to open the RuleApp editor. Then, in the Deployment pane of the RuleApp editor, click **Deploy**.
   – Using the menu options by right-clicking the project and selecting **Validate** → **Deploy**, as shown in Figure 3-77.

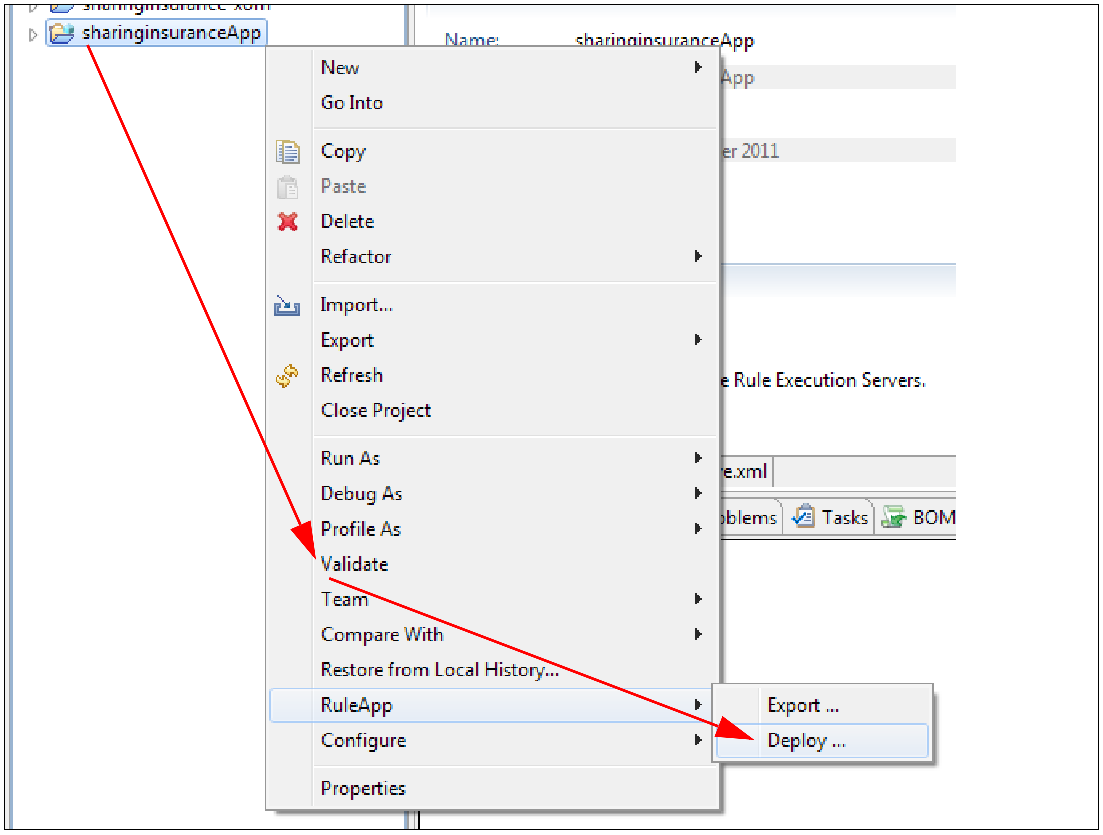


Figure 3-77 Deploying the RuleApp

2. For the deployment type, accept the **Increment RuleApp major version** default option, and click **Next**.
3. Select **Create a temporary Rule Execution Server configuration**, and enter the following details:
   – URL: `http://<your.server.address>:<PORT>/res`
   – Login: `resAdmin`
   – Password: `resAdmin`

   Click **Finish**.

Your artifacts are deployed to zRule Execution Server for z/OS. You can now build a COBOL application to invoke the rule execution.

3.3.4 Building a COBOL application for rule execution

You deployed the rule artifacts to zRule Execution Server for z/OS. You can now use the generated copybook to build a COBOL application for rule execution on z/OS.

Additional resources: You can find the examples that are used in this section in the additional information that is included in this book. For details, see Appendix C, "Additional material" on page 335.

Generated copybook example

First, you must know the structure of the generated copybook, as shown in Example 3-6.

Example 3-6 Generated copybook INSSHAR.cpy

```
01 request.
           02 driver.
               03 age pic S9(5).
               03 first-name pic X(20) value SPACE.
               03 house-num pic S9(10).
               03 last-name pic X(20) value SPACE.
               03 lic-date pic 9(8).
               03 lic-status pic X.
                   88 BoolValue value 'T'.
               03 number-accidents pic S9(5).
               03 zipcode pic X(20) value SPACE.
           02 vehicle.
               03 make pic X(20) value SPACE.
               03 model pic X(20) value SPACE.
               03 vec-id pic X(20) value SPACE.
               03 vec-type pic X(20) value SPACE.
               03 vec-value usage COMP-1.
       01 response.
           02 approved pic X.
               88 BoolValue value 'T'.
           02 base-price usage COMP-2.
           02 dis-price usage COMP-2.
           02 messages-Num pic 9(9).
           02 messages pic X(60) value SPACE Occurs 10 Times.
```

COBOL application example

Now, you can build a COBOL application according to the generated copybook, as shown in Example 3-7.

Example 3-7 COBOL application INSSHAR.cbl

```
IDENTIFICATION DIVISION.
       PROGRAM-ID. "INSSHAR".
       ENVIRONMENT DIVISION.
       DATA DIVISION.
       WORKING-STORAGE SECTION.
      * include the generated copybook
           COPY INSSHAR.
       01 WS-REASON-CODES.
           COPY HBRC.
           COPY HBRWS.
       PROCEDURE DIVISION.
      * Init ruleset parameter data
           MOVE 'John'             TO FIRST-NAME
           MOVE 17                 TO AGE
```

```
          MOVE 4                  TO NUMBER-ACCIDENTS
       ......
      * Move ruleset path to table HBRA-RA-PARMETERS
      ......
          MOVE "/sharinginsuranceApp/sharinginsurancerules"
                                  TO HBRA-CONN-RULEAPP-PATH
      * move ruleset parameter for request and response
          MOVE 'request'          TO HBRA-RA-PARAMETER-NAME(1)
      ......
          MOVE 'response'         TO HBRA-RA-PARAMETER-NAME(2)
        ......
      * Get connection to rule execution server
          CALL 'HBRCONN' USING HBRA-CONN-AREA.
      ......
      * Invoke rule execution server
          CALL 'HBRRULE' USING HBRA-CONN-AREA
      ......
      * Get disconnect to rule execution server
          CALL 'HBRDISC' USING HBRA-CONN-AREA
      ......
      * Display result
          DISPLAY        "RESPONSE APPROVED: " APPROVED
      ......
     STOP RUN.
```

Rule execution result

You can compile and run the COBOL application on z/OS. The COBOL application example in Example 3-7 on page 84 produces the results that are shown in Example 3-8.

Example 3-8 Results of compiling and running the COBOL application

```
********** EXECUTION RESULT *********
DRIVER NAME: John
RESPONSE APPROVED: F
Reject messages:
Accidents number exceeds the maximum
The age exceeds the maximum or minimum
**************************************
```

4

Managing business decisions through the full lifecycle

This chapter looks at the lifecycle of a decision and considerations when deploying to a z/OS environment.

The following topics are covered in this chapter:

© Copyright IBM Corp. 2013. All rights reserved.

4.1 What is the lifecycle of rule artifacts in decisions

Rule artifacts within Operational Decision Manager follow a general lifecycle that can be tailored by the particular setup of your system. Rules can pass through one or more of the available tools and through one or more repositories or databases before being deployed to the production server. Your lifecycle can contain development, testing, production, and maintenance areas, that are maintained by developers and business users as appropriate.

A typical lifecycle includes rulesets and flows that are being developed by the development department and saved in a central repository. Specific rules can then be accessed and maintained by business users without the necessity of returning the rules to development to be implemented in the software. From this point, rule artifacts are tested and then can be deployed to the main production system. Maintenance can be carried out either by the business users (for rules modifications) or by development (for more substantial changes). A diagram of this process is shown in Figure 4-1.

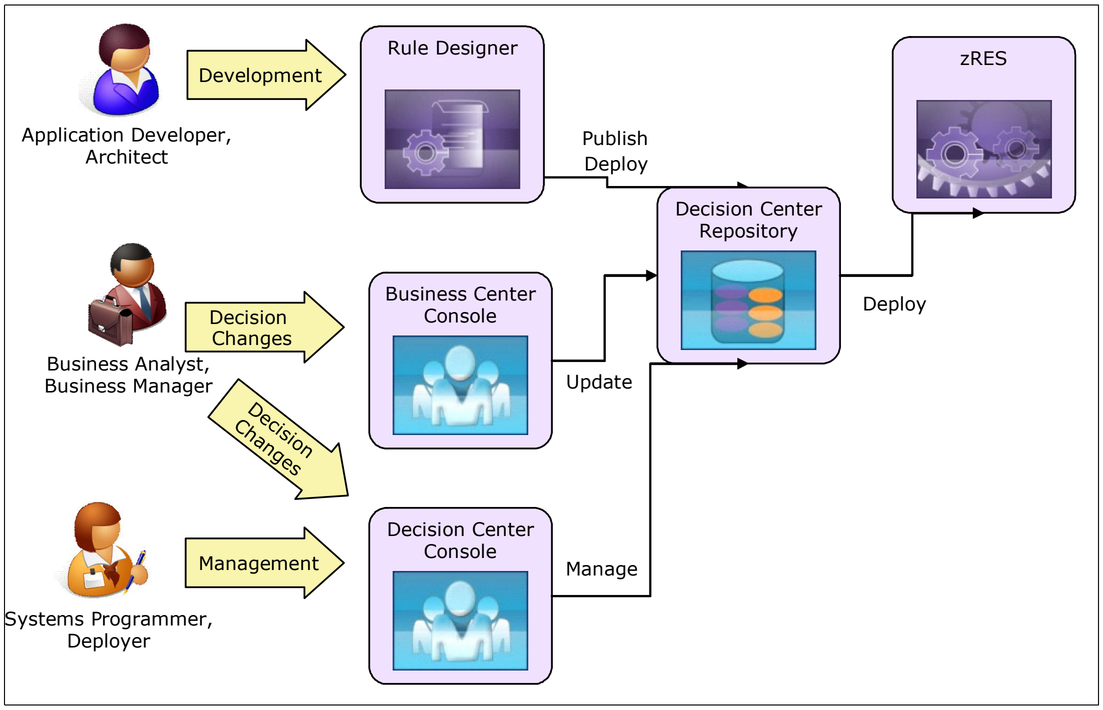


Figure 4-1 Users and management of rules

One of the key reasons for extracting decisions into a decision management system is that it allows you to separate the lifecycle of the decisions from the lifecycle of the application that invokes the decision. This separation is important particularly on System z where the application deployment cycles are often long, and the business wants to be able to change the business behavior of the application more quickly. By separating the decision lifecycle from the application lifecycle, you can gain agility in your business applications without sacrificing the reliability of the core logic of your business applications.

4.2 Working with rules through the lifecycle

The decision lifecycle starts with the initial location of the rules within the business applications. After the rules are identified, the data that is associated with the decisions can be identified and a copybook can be created that contains the required information.

The business application must be refactored to remove the current implementation of the business decision and replace it with calls out to the Rule Execution Server for z/OS (zRES). The application might have decision logic and business logic interspersed throughout it. To optimize the application, refactor the application so that it makes the fewest calls possible to decisions, as shown in Figure 4-2.

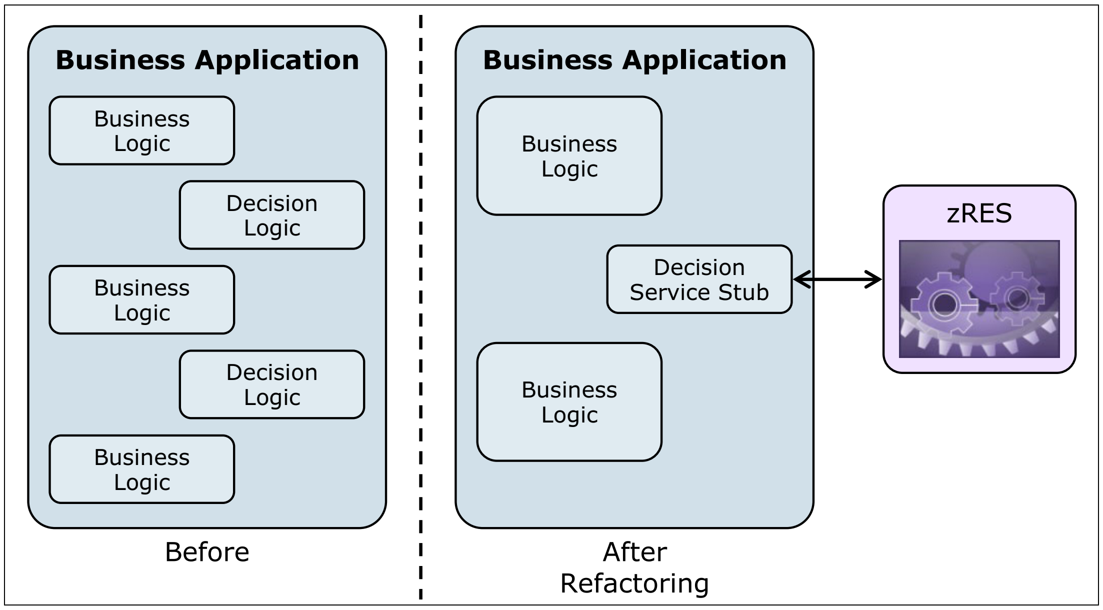


Figure 4-2 Refactoring the business application

After the copybook containing the required data for the decision is created or identified, the business rules for the initial version of the business decision can be authored. The initial creation of the business object model (BOM) and verbalization is done in Rule Designer. This action is generally considered an IT project, because it requires knowledge of the COBOL data structure.

After the BOM is created, the IT department typically creates the initial version of the business rules for the decision based on the rules that were previously embedded in the application. After the first version of the business rules is created, the decision can be deployed to a zRES environment and tested with the application.

With the first version of the decision now deployed, you must decide how to manage the ongoing lifecycle of the decision. There are multiple approaches to this problem. In certain development shops, the management of the business decisions remains a purely IT-based process, using the Rule Designer environment for managing and maintaining the decisions. Now that the business rules from the decision are authored in a far more accessible language, it is often advantageous to allow the business team to interact directly with the authored rules.

Business users can view and modify rules using Decision Center or Business Center as shown in Figure 4-3. Testing can still be carried out in the Decision Center without needing to involve the IT department.

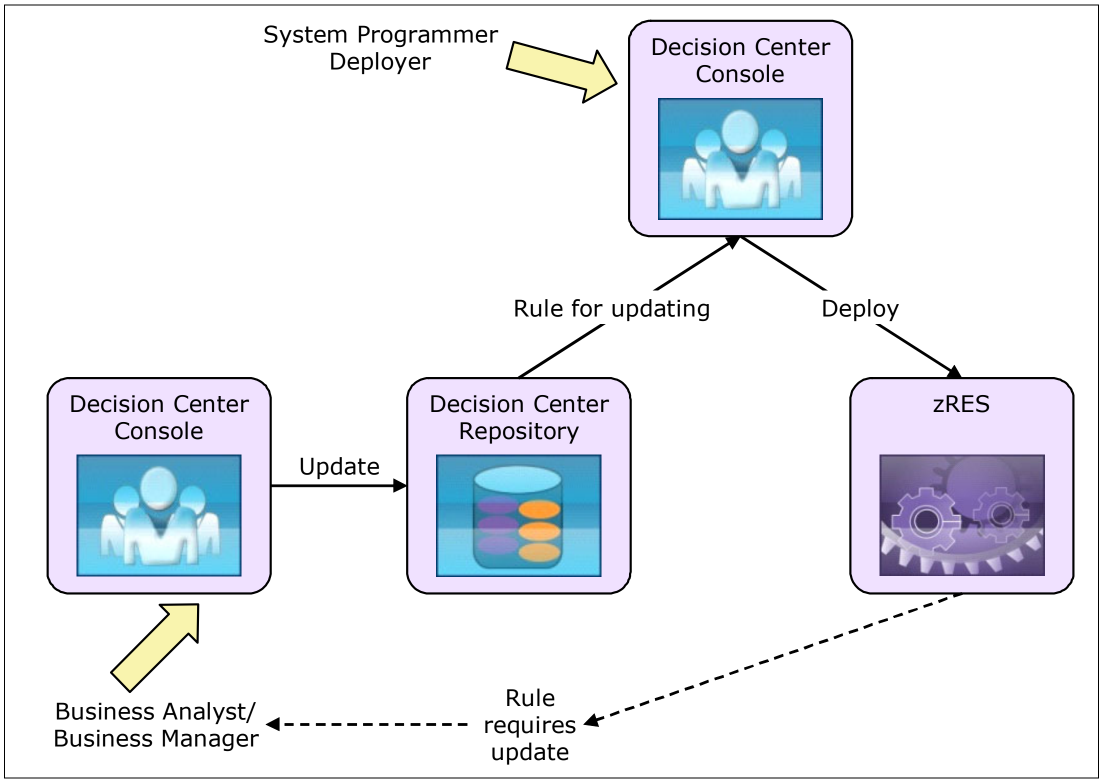


Figure 4-3 Movement of a rule through the lifecycle

After the initial deployment of the rules, the application developer can also publish the rules to the Decision Center repository. Other roles can become involved in the decision lifecycle.

The Decision Center console provides two distinct functions:

- The ability to change the rules of decisions that are published to the repository
- The ability to manage versions and deploy decisions from the repository to the rule runtime environments

An important part of the decision lifecycle is the ability to test the changes to the decision before deploying it to the final server. You can use Decision Center to define scenarios or test cases using, for example, a spreadsheet format to define the input and expected output. Decision Center can then take this data, deploy the ruleset to a configured zRES, and execute the decisions based on the supplied data. Decision Center can report situations where expected results are not returned. This function gives you the ability for the decision lifecycle to happen completely in isolation from the application lifecycle. Changes to the decision behavior can be tested in isolation from the application before they are deployed into the production system. For information about decision testing, see Chapter 6, “Decision testing and simulation” on page 117.

Also, the Business console is available. It allows searching, viewing, and updating rules but it does not give users any other access. The Business console also allows users to view the modifications of a rule on a timeline. It does not currently offer the testing facilities that are available to the Decision Center.

4.2.1 Managing artifacts

There are a number of artifacts in the decision lifecycle that require managing, as shown in Figure 4-4:

- Ruleset
- Java execution object module (XOM)
- RuleApp

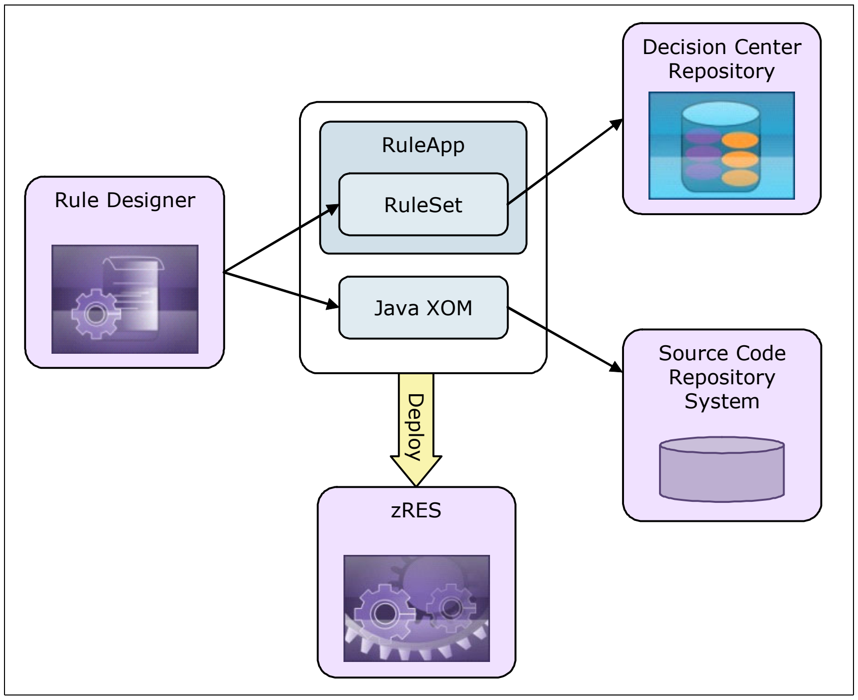


Figure 4-4 Artifacts within the decision lifecycle

Ruleset

The ruleset contains a number of elements:

- The business object model (BOM)
- The verbalization of that model
- The authored rules from the decision
- The ruleflow that guides the execution of the decision
- The declaration of the required parameters for this decision

The ruleset is the primary artifact that requires management. It contains the rules themselves and the BOM on which they are based. The *ruleset* is the artifact that is published to Decision Center. You can access the Decision Center to change the rules within a decision.

A ruleset can inherit from another ruleset. In this case, the decision contains the BOM, verbalization, and rules from both rulesets. This approach is a useful way to reuse rules that are shared across multiple decisions and manage the changes to the shared rules in a single project.

The name of a ruleset is significant. It forms part of the ruleset path that is used by the client to identify the decision that it wants to invoke on the server.

Any changes to the rules within a decision require the ruleset to be redeployed to make the new decision behavior available. A preferred practice is to increment the minor version of the decision when redeploying behavioral changes in the decision.

Java XOM

The *Java execution object module (XOM)* is standard Java code that is the Java representation of the imported COBOL copybook. zRES uses this Java XOM at run time for mapping the COBOL data structure before rules can be executed. The Java XOM can be deployed to the zRES server directly from Rule Designer or by using scripts on the server instance.

Because the code is standard Java code, it must be managed and maintained by using a source code management system. Rule Designer is based on Eclipse. Many of the standard source code management systems have plug-ins that allow you to manage and handle versioning for the Eclipse Java project directly from the Rule Designer environment. If the starting point for the project is an imported copybook, this code is generated by Rule Designer and must not be edited. If possible, mark this code as read only within the source code management system.

Only deploy the Java XOM if there are changes to the underlying data structures that define the interface to this decision. This situation occurs when a change is made to the COBOL copybook that was imported to create the BOM. A change of this nature also requires a corresponding change in the COBOL client applications to use the new copybook structure. For this reason, changes to the COBOL copybook are considered an IT project and happen less frequently than changes to the decision behavior in the rules. A preferred practice is to increment the major version number of the decision when making these interface changes.

RuleApp

The *RuleApp* is the deployment container for one or a number of related rulesets. RuleApps are created for deployment either within Rule Designer or Decision Center. They are a compressed (`.zip`) file that contains the required artifacts to execute the decision.

A RuleApp can be deployed directly to a server from either Rule Designer or Decision Center. Or, a RuleApp can be exported as a JAR file that can be managed externally to the Operational Decision Manager tool and deployed to a server by using scripts. This approach can be useful when defining the process of performing decision updates where there is no access from Rule Designer or Decision Center to the production zRES for z/OS.

The name of a RuleApp is significant. The name forms part of the ruleset path that is used by the client to identify the decision that it wants to invoke on the server.

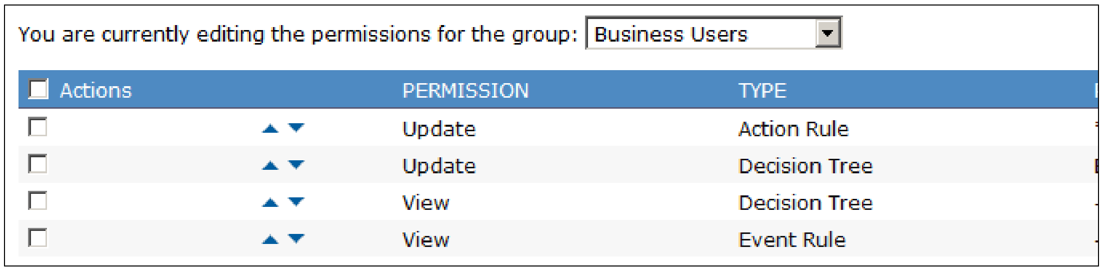
4.2.2 What roles are involved in the decision lifecycle

There are three roles that are involved in the lifecycle of a decision. The names vary from company to company, but there are normally people who can be attributed to one or more of the following roles:

- Application/decision developer
- Systems administrator/programmer
- Business team member

The major interaction among the team members occurs in the Decision Center environment. Here, a developer synchronizes the RuleApps on which the developer is working in Rule Designer. The systems administrator goes to Rule Designer to version and deploy decisions. The business team accesses Rule Designer to view or change decisions.

The Decision Center environment provides role-based access authorities to allow the systems administrator to give people the correct authority to access the rulesets for which they are responsible. The granularity of access authority is delivered at the ruleset level. But, with a little customizing, the granularity of management can be changed to match whatever is required by the organization. Figure 4-5 shows how a set of permissions might appear for a particular group of users.


You are currently editing the permissions for the group: Business Users

Actions	PERMISSION	TYPE
☐ ▲▼	Update	Action Rule
☐ ▲▼	Update	Decision Tree
☐ ▲▼	View	Decision Tree
☐ ▲▼	View	Event Rule

Figure 4-5 Example permission settings for a business user

Rule developer

The rule developer is generally an IT-based person. In most organizations, the rule developer is also part of the application development team that is responsible for the application that is being modernized by having its decisions extracted. In a larger organization, a dedicated team with the skill to externalize and develop decisions might exist, outside of the application development team.

The rule developer is responsible for creating the initial version of the BOM and the verbalization of that model. The rule developer normally writes the first draft or pass at the rules within the decision based on what currently exists within the application code. The rule developer's primary tool is Rule Designer. The rule developer is responsible for publishing the ruleset to Decision Center. This person also ensures that the Java XOM code is maintained in a source code management system.

Systems administrator

The systems administrator is responsible for the production zRES servers and their actions. In the rule lifecycle, the systems administrator generally is responsible for versioning and deploying new versions of a decision into the zRES. The systems administrator is also responsible for maintaining the security model within the Decision Center environment to ensure that users have access to only the rulesets for which they are responsible.

Business team

The business team is ultimately responsible for the business policies, which are enforced through the business decisions. The business team provides the input to the behavior of the business decisions. Depending on the level of adoption, the business team's interaction varies.

The business team provides input to the rule development team for business policy changes to implement in the decisions. In one scenario, the business team can use Business Center to search, view, and modify specific rules from the existing system, requesting assistance from the rules development team if it is required.

In another scenario, the business team can view the rules from existing decisions in Decision Center and make recommendations to the rule development team to update specific rules that must be made to implement new business policies. In this mode, the business team has read-only access to the rulesets that contain the applicable rules.

In a third scenario, the business team changes the rules within a decision directly to implement changes that are required as business policies are updated. The business team tests the changes to the decisions by using the testing and simulation capabilities that are accessed through Decision Center. The business team then notifies the systems administrator that a new version of a decision is available and must be deployed.

4.3 Sharing decision artifacts between z/OS and a distributed environment

When considering sharing decisions between z/OS and other platforms, ensure that the correct decision artifacts are available on each platform. As part of the decision lifecycle, the required artifacts can come from separate management systems.

Figure 4-6 shows the artifacts that are involved in a decision and the artifacts that are required to be deployed to various platforms.

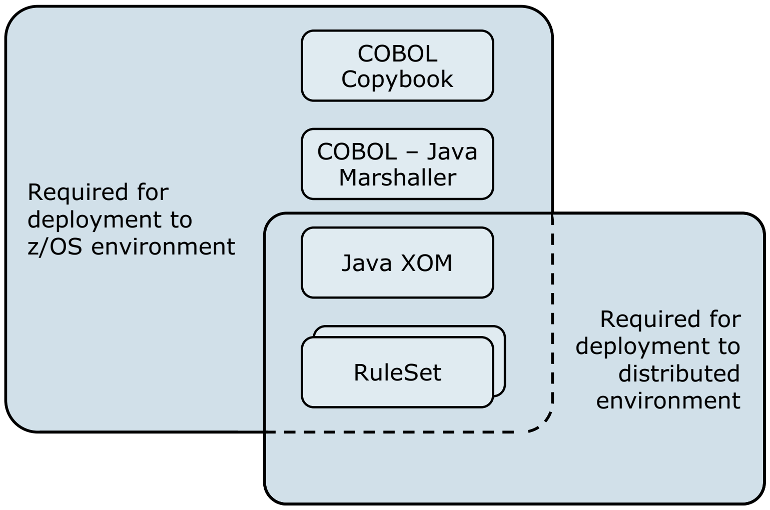

COBOL Copybook
COBOL – Java Marshaller
Required for deployment to z/OS environment
Java XOM
RuleSet
Required for deployment to distributed environment

Figure 4-6 Deployment of decision artifacts

Java marshaller: The Java marshaller that is shown in Figure 4-6 is used to convert the COBOL data to Java so that it can be accessed by the Java based ruleset. In Rules Designer and Decision Center, it is contained within the Java XOM and only appears as a separate entity when deployed to the zRES server on a z/OS platform. It is only required on the z/OS platform and must not be adjusted.

Normally, only the rulesets and the Java XOM are required on the distributed platform. Often in this configuration, the client is local to the Rule Execution Server for distributed. The Java XOM is part of the client application class path, so it is not explicitly deployed to the server as a resource. The rule session inherits the class path of the client application.

On zRES, the COBOL copybook that was either imported into, or generated by, Rule Designer is required by the client COBOL application. The COBOL copybook ensures that the data layout is exactly the layout that is expected by the marshalling code. Because the client in this case is COBOL and the connection to the server is managed by the zRES API stub, the Java XOM resource must be deployed to the server. This deployment can be done either from the Rule Designer environment or locally using supplied scripts.

In both cases, the Rule Execution Server requires that the ruleset is deployed within a RuleApp. This deployment is from either Rule Designer, Decision Center, or locally by using scripts.

When sharing the decision across multiple platforms, it is important to make sure that the decision lifecycle updates and deploys the correct parts of the decision when changes are made. If the underlying data structure definitions are not changed when updating a decision, only the rulesets within a RuleApp must be redeployed to make the new decision version available. If changes are made to the copybook that is used to create the BOM, or if the Java XOM was used to create the copybook, you need to redeploy artifacts. Redeploy artifacts on all platforms where the decision is implemented to minimize the chances of unexpected behavior or failures. For this reason, changes to the data model must be minimized after the decisions are in production and considered an IT project to implement.

4.4 Installation topologies for Decision Center and Business Center

The locations of the Decision Center repository and the Decision Center and Business Center consoles are largely independent of where the decision executes. Figure 4-7 on page 96 shows possible options for installing a Decision Center to be used with zRES. However, it is likely that more than one Decision Center console will be employed, possibly in different locations. The Business Center console can also be in one or more locations.

4.4.1 Basic topologies

Figure 4-7 shows three topologies using only one instance of the Decision Center console and the Decision Center repository.

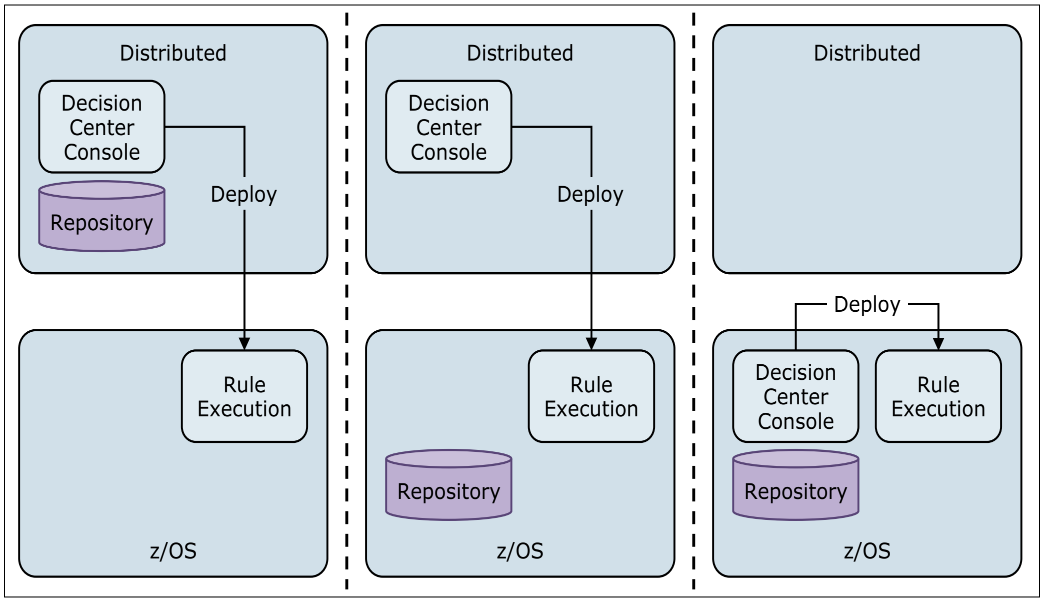


Figure 4-7 Deployment options for Decision Center repository and console

The Decision and Business Center consoles require deployment to a web container. This web container can be any one of the supported Java Platform, Enterprise Edition (Java EE) application servers for distributed or WebSphere Application Server for z/OS. The Decision and Business Center consoles also require Java Database Connectivity (JDBC) access to the repository database.

The following sections provide a brief description of the topologies that are shown in Figure 4-7.

Topology 1: Decision Center console and repository on distributed

In this topology, both the Decision Center console and repository are hosted on a distributed or Linux for System z platform. The standard deployment from the Decision Center console to Rules Execution Server for z/OS (zRES) is by HTTP-based communication. As long as access is granted to the specific ports that are configured for deployment on the zRES or other Rule Execution Server instances, it is straightforward to deploy to any zRES instance. To Decision Center, the deployment interface to zRES looks the same as other Java EE deployed Rule Execution Server instances.

All security to the console is role-based using the underlying application server to perform authentication checks. You might prefer this configuration, because it does not require all users of the Decision Center console to be defined to z/OS security.

Topology 2: Decision Center console on distributed and repository on z/OS

This topology is similar to Topology 1. However, the database is on a z/OS logical partition (LPAR). In this case, the Decision Center console must also have ports that are enabled in any firewall to allow remote client access to the database on z/OS. Generally, Decision Center is used less than the Decision Server, so the remote location of the database is not a performance problem.

You might prefer this configuration if the rule repository is required to have z/OS qualities of services (QoS) associated with it or if the rule repository is managed on the same platform as the COBOL source code repository. Generally, in this case, the security access to the remote database is delegated to an application server-level connection. That way, each Decision Center console user does not have to be defined to z/OS security.

Topology 3: Decision Center console and repository on z/OS

This topology places both the Decision Center console and repository on z/OS. The Decision Center console requires a WebSphere Application Server for z/OS instance in which to run and the repository requires an IBM DB2® instance. This topology can also be deployed to distributed Rule Execution Server instances, as well as to zRES instances.

You might prefer this configuration if all administration and deployment of a project are contained within the z/OS teams.

4.4.2 Advanced topologies

More complex topologies can also be used, with multiple Decision or Business Center consoles, in more than one location. It is also possible to use multiple repositories, for example, to separate out rules under development, rules under test, and rules currently in production. This can lead to any number of topologies combining the different resources that are in use. Figure 4-8 shows an example of this type of layout.

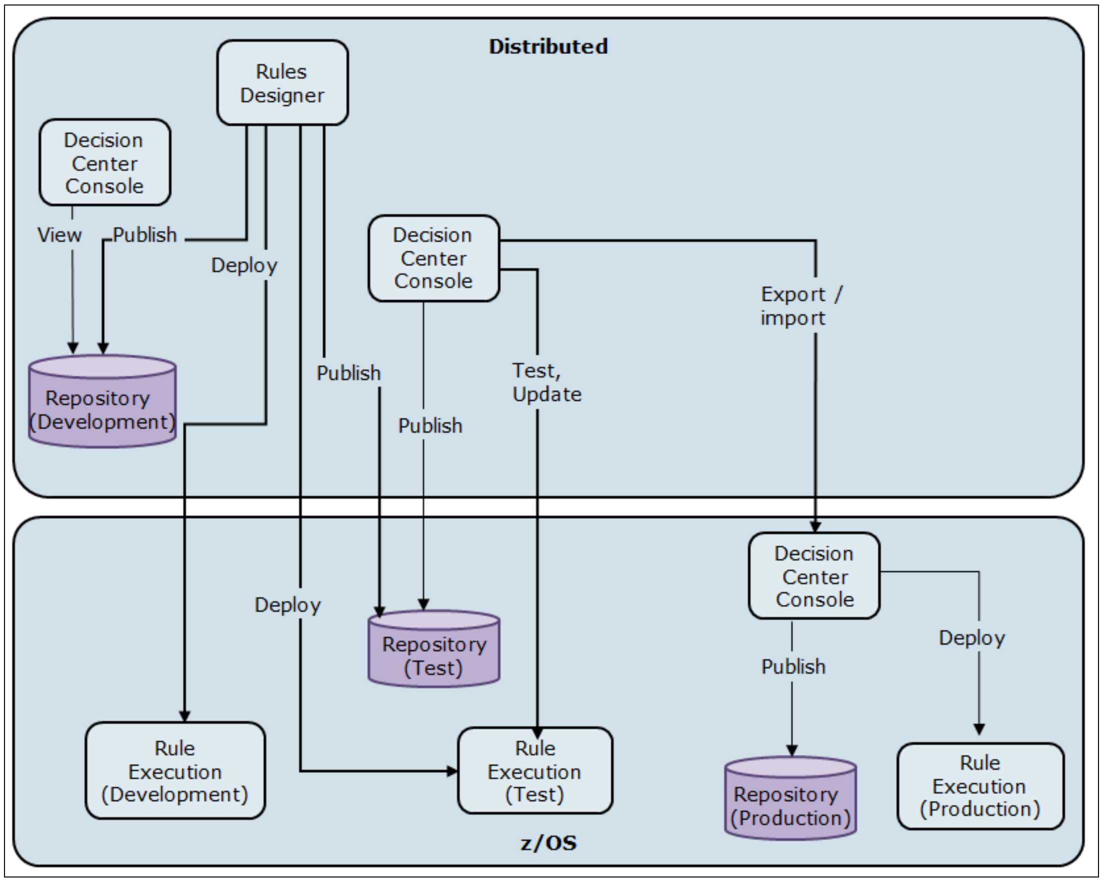


Figure 4-8 Example of an advanced layout for Decision Center consoles and repositories

In Figure 4-8 on page 97, three Decision Centers consoles are in use, and they are accessing three separate repositories. Rules under development are developed in Rules Designer by the IT department and are published to a Development repository so that they can be viewed by the business department in a Decision Center console that runs on a distributed platform. These rules can then be deployed to a zRES, which is maintained by a Decision Center console running on distributed, but using a Decision Center repository that is on z/OS. The rules can be modified, as required, and are stored in the Decision Center repository. After the rules are finalized, they are exported and then deployed directly to the production zRES. They are imported into a production repository that is stored on z/OS so that they can be viewed by a Decision Center console, which is also running on z/OS, and deployed directly to the zRES using the Representational State Transfer (REST) API.

4.5 Managing artifacts through the lifecycle

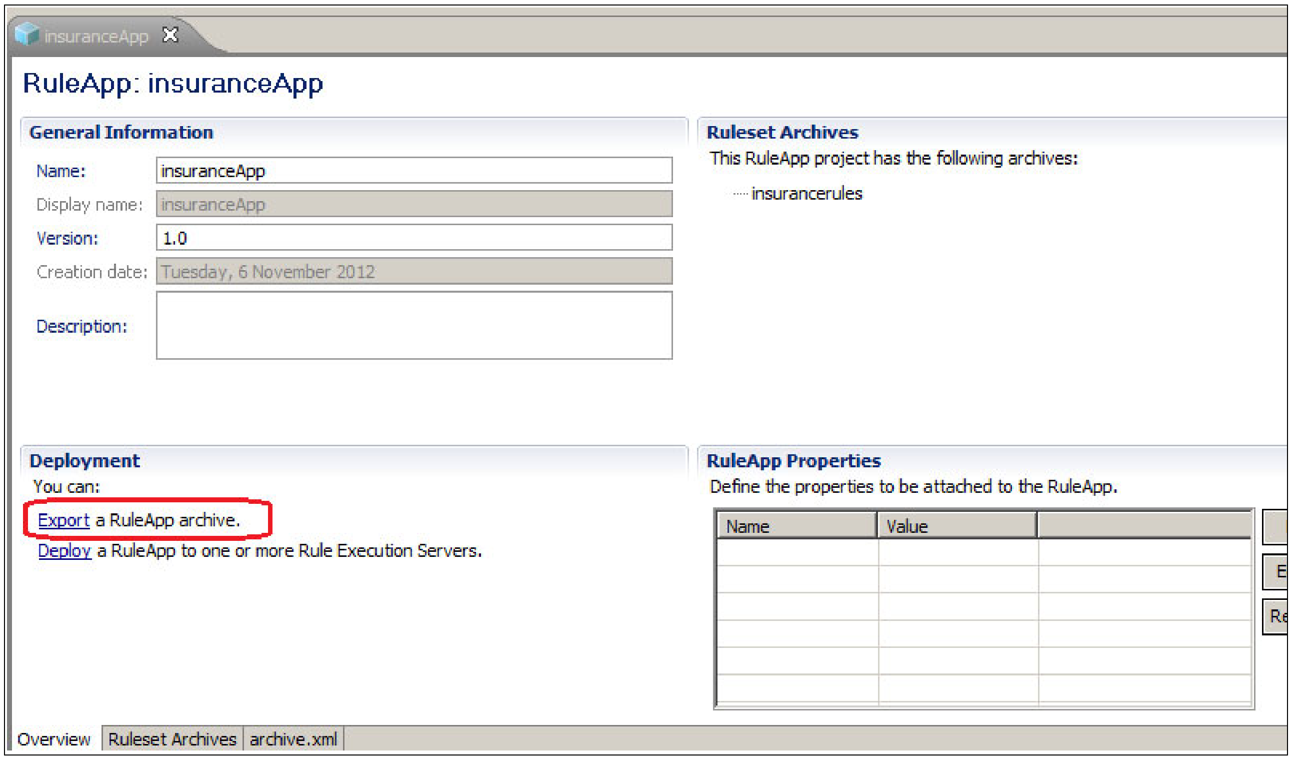
This section describes how the tools that are available in Operational Decision Manager for z/OS can be used to manage artifacts through the lifecycle.

4.5.1 Rules Designer

The Rules Designer is primarily a tool for creating rule artifacts. It is used for the creation of rules, ruleflows, decision trees, and other rules artifacts. It can be used to deploy rules to the zRES.

Because this tool is primarily aimed at an IT department rather than a system programmer, it is unlikely that it will be used to deploy a rule artifact to the production system. The main communication for the Rules Designer is with development and the test Rules Execution Servers, and with Decision Center repositories in these areas. It can, however, be used to export rules artifacts for later deployment to a rule server by using external scripts that implement the REST API.

Figure 4-9 shows the export window for a RuleApp.


insuranceApp

RuleApp: insuranceApp

General Information

Name:	insuranceApp
Display name:	insuranceApp
Version:	1.0
Creation date:	Tuesday, 6 November 2012
Description:	

Ruleset Archives

This RuleApp project has the following archives:

insurancerules

Deployment

You can:

Export a RuleApp archive.

Deploy a RuleApp to one or more Rule Execution Servers.

RuleApp Properties

Define the properties to be attached to the RuleApp.

Name	Value	

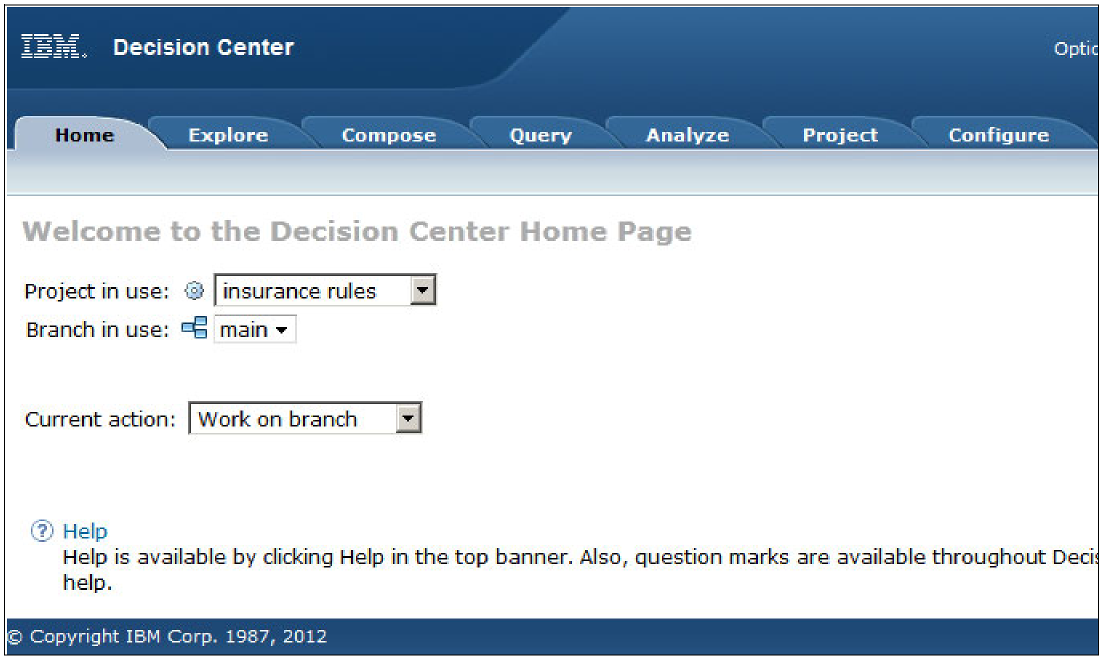
Figure 4-9 Deployment and export from Rules Designer

Anything that is deployed from the Rule Designer to a zRES takes effect immediately.

The Rule Designer can also be used to publish rules to a Decision Center repository by first connecting to the appropriate Decision Center and then synchronizing with it. This allows for publishing rules to the main root or to one of the branches. Roots and branches are described in “Versioning” on page 100.

4.5.2 Decision Center

The Decision Center, as shown in Figure 4-10, can be configured to support a variety of users, allowing it to be a central tool in managing artifacts currently in use. Anything deployed from the Decision Center will be effective immediately on the zRES to which it has been deployed.



Welcome to the Decision Center Home Page

Project in use: insurance rules

Branch in use: main

Current action: Work on branch

Help

Help is available by clicking Help in the top banner. Also, question marks are available throughout Deci
help.

© Copyright IBM Corp. 1987, 2012

Figure 4-10 Decision Center on startup

When configured for business users, it might allow the modification of specific rules and events and viewing of more complex scenarios. Deploying might not be appropriate in these circumstances. Rules can be updated or modified. Testing a specific rule can also be done.

When used by a developer, rule artifacts can be created and modified, and then deployed to a zRES. Although, more often, the Rule Designer is used for the main development and Decision Center is used as a maintenance tool.

When used by a system programmer, it is likely that the majority of functions are enabled. Although in some cases, it might be useful to disable the editing capabilities on artifacts that have been developed by the other teams. The system programmer's primary use of the Decision Center is to deploy the rule artifacts to the appropriate Rule Execution Servers. The Decision Center might also be used for exporting so that rules can be deployed to the production server by using the REST APIs, as described in 4.5.4, "REST API" on page 101.

Important: The Decision Center cannot be used to deploy rules directly to the zRES.

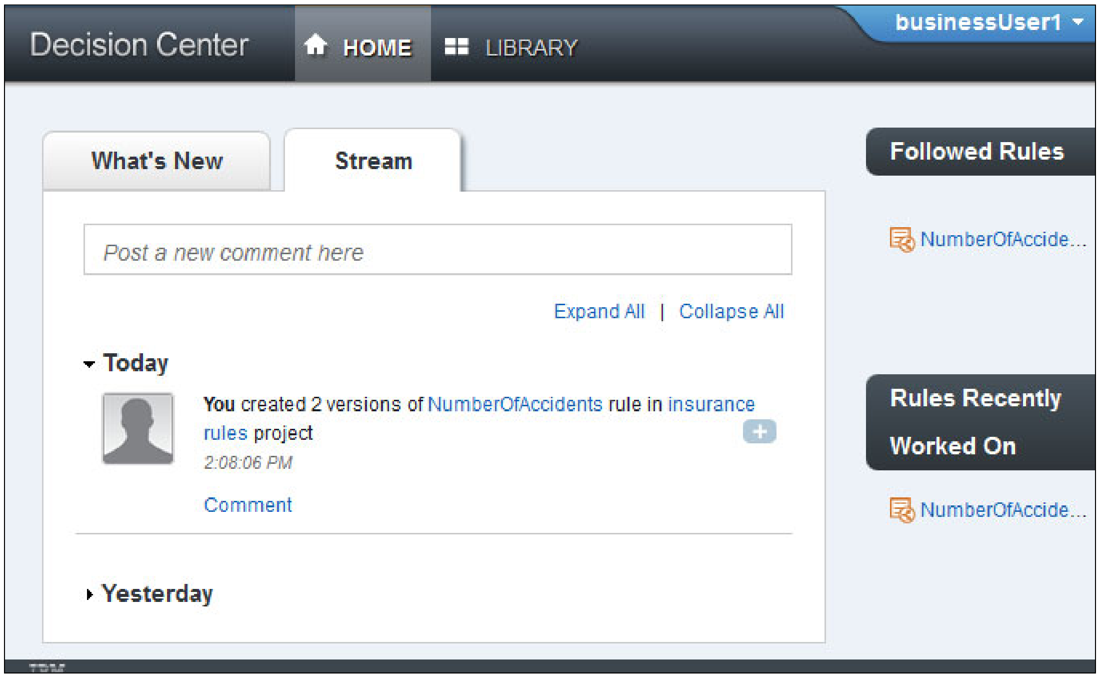
Versioning

When a new version of an element is modified, the Decision Center creates an archived version of that element. Therefore, the history of a particular element can be tracked by reviewing those archived versions.

4.5.3 Business Center

The Business Center, as shown in Figure 4-11, is primarily a tool for use by business users, and it has little capability in the lifecycle. It allows for the searching, viewing, and editing of rules. It also allows timelines to be used to see the history of a rule. However, it has no testing, deploying, or publishing capabilities, and its editing capabilities are limited.

The editing of rules in the Business Center causes an archived version of the rules to be created as described in “Versioning” on page 100.



You created 2 versions of NumberOfAccidents rule in insurance rules project
2:08:06 PM

Figure 4-11 Business Center

4.5.4 REST API

Operational Decision Manager allows for the deployment of a rule’s artifacts by using the REST API services. Therefore, the deployment of a rule artifact can be scripted to ensure that it is deployed in the same way each time. It also means that rule artifacts can be published without using an HTTP connection between the tool and zRES. This is a very useful method when deploying to a production environment, because it preserves security on the production system.

As shown in Figure 4-12 on page 102, to use this method, first the rule artifact is exported, and then, it is deployed to the zRES server by using scripts by using the REST API. The archive file can be copied to the appropriate location, which means that HTTP connections are not involved.

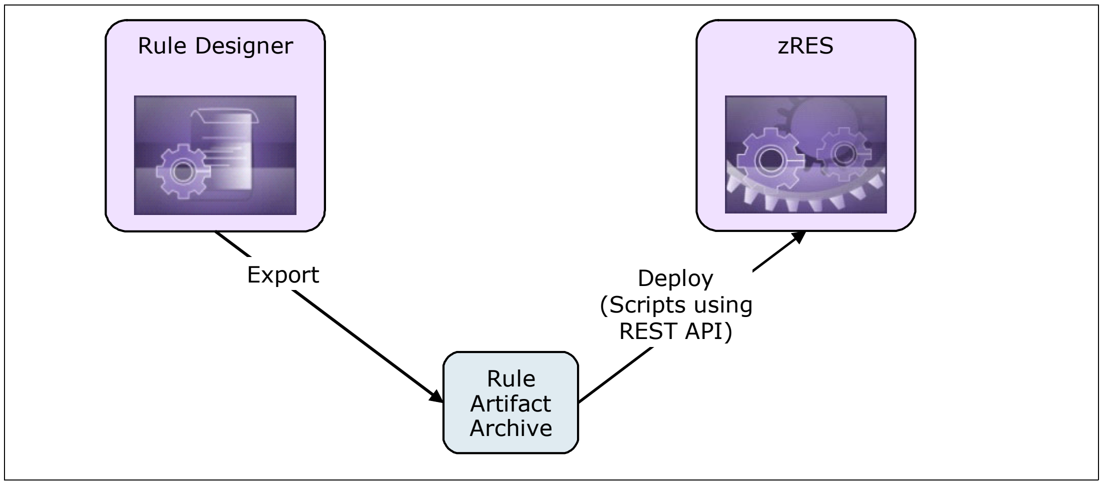


Figure 4-12 Deploying a rule artefact by using REST API services without using HTTP

4.5.5 ANT scripts

Using the ANT scripts method of rules deployment affects only the database and does not inform running applications (such as the zRES) that changes have occurred. Therefore, any changes that use this method need the zRES to be restarted for the changes to take effect.

Tip: It is a preferred practice that REST APIs are used rather than the ANT scripts method.

4.6 Usage of defined rules

After a rule artifact is defined to the Decision Server, it is available for use by calling applications. If the artifacts are deployed by a method other than the ANT scripts, no restart of the Decision Server is required. This is a preferable method when the zRES is a production one.

The rule can then be modified separately to the application by using the methodologies described in this chapter. The application using the rule does not require modification for an update to be made to that rule. By using a deployment method that requires no restart, there does not need to be any interruption of service for the rules to be updated.

5

Invoking the rules server from COBOL clients

This chapter describes the design of the decision interface. It describes the mapping between the COBOL data structures passed into the decision run time and the Java structures that are used to execute the rules. This chapter explains how you can then customize this mapping.

This chapter also describes starting a new project for deployment from a COBOL copybook.

The following topics are covered in this chapter:

© Copyright IBM Corp. 2013. All rights reserved.

5.1 Designing the decision interface

The starting point for any successful decision project is to design the decision interface correctly. Often, there is an existing copybook that is used by the application that is being enabled to access Operational Decision Manager. The natural starting point seems to be to reuse this data structure as the interface to the decision. Although this approach might seem to be the easiest, there are benefits to be gained by giving the interface more consideration.

When designing the decision interface, consider why you are externalizing business rules into an external decision server. One of the main reasons might be that the business decisions change on a shorter lifecycle than the applications that invoke them. Therefore, you might want to isolate the application from changes to the business rules in the decision. However, you also might want to isolate the business decision from maintenance changes that occur in the application, as well.

Changes in the application can alter the copybook that the application uses. Application changes result in the need to import the copybook again to update the business object model (BOM) that is used in the rules. These changes can potentially disrupt rules that are already authored.

Also, the data in the copybook might not be suitable for use with rule authoring. Application copybooks in COBOL are often a mix between a representation of the business data and its specific entries. For example, often COBOL FILLER statements are put in to align data, or fields are created to hold return codes or other diagnostic information. These fields have no business relevance to the decision and are confusing if they are included in the BOM.

It is important to ensure that the data passed across to the business decision contains all the required information to successfully make the decision. When thinking about the data, consider information that might not be used in the business rules that is embedded in the application today. This information might be useful to develop a better business decision after it is externalized. It is far easier to pass more data across when designing the decision interface from the outset than to re-engineer the interface at a later date or to add code into the decision to retrieve external data.

It is a preferred practice to design the interface to a business decision in the same way that you design a service interface. Consider the following information:

- The data that is used in the decision today
- The data that is easily available to the application
- The data that is required today, and potentially in the future, to maintain the decision after it is externalized

Create a copybook to hold that information specifically for the decision interface that can be versioned for that purpose. The additional cost of a few COBOL MOVE statements is outweighed by the flexibility that this approach provides in isolating changes in both the decision and the calling application.

5.2 Coding the COBOL client application

To access the Rules Execution Server for z/OS (zRES), the COBOL application must use the supplied client API. This API consists of the following API calls:

HBRCONN	To connect to the server
HBRRULE	To execute a decision
HBRDISC	To disconnect from the server

Each API call takes the HBRWS copybook-defined structure as a parameter. These calls are shown in 3.2.10, “Building a COBOL application for rule execution” on page 71.

5.2.1 HBRWS header structure

The HBRWS header structure is required for all zRule Execution Server for z/OS (zRES) client API calls. It is in the `<INSTALL-ROOT>.SHBRCOBS` data set. It must be included in any client programs that call the zRule Execution Server for z/OS.

Example 5-1 shows the layout of the HBRWS header structure.

Example 5-1 Layout of the HBRWS header structure

```
01 HBRA-CONN-AREA.
    10 HBRA-CONN-EYE                PIC X(4)   VALUE 'HBRC'.
    10 HBRA-CONN-LENTH              PIC S9(8)  COMP  VALUE +3536.
    10 HBRA-CONN-VERSION            PIC S9(8)  COMP  VALUE +2.
    10 HBRA-CONN-RETURN-CODES.
       15 HBRA-CONN-COMPLETION-CODE PIC S9(8)  COMP  VALUE -1.
       15 HBRA-CONN-REASON-CODE     PIC S9(8)  COMP  VALUE -1.
    10 HBRA-CONN-FLAGS              PIC S9(8)  COMP  VALUE +1.
    10 HBRA-CONN-INSTANCE.
       15 HBRA-CONN-PRODCODE        PIC X(4)   VALUE SPACES.
       15 HBRA-CONN-INSTCODE        PIC X(12)  VALUE SPACES.
       15 HBRA-CONN-SSID            PIC X(4)   VALUE SPACES.
       15 HBRA-CONN-GRPID           PIC X(4)   VALUE SPACES.
    10 HBRA-RESERVED01              PIC S9(8)  COMP  VALUE 0.
    10 HBRA-RESERVED02              PIC S9(8)  COMP  VALUE 0.
    10 HBRA-RESERVED03              PIC S9(8)  COMP  VALUE 0.
    10 HBRA-CONN-RULE-CCSID         PIC S9(8)  COMP  VALUE 0.
    10 HBRA-CONN-RULEAPP-PATH       PIC X(256) VALUE SPACES.
    10 HBRA-RESPONSE-AREA                     VALUE SPACES.
       15 HBRA-RESPONSE-MESSAGE     PIC X(1024).
    10 HBRA-RA-INIT                           VALUE LOW-VALUES.
       15 HBRA-RESERVED04           PIC X(1792).
    10 HBRA-RA-PARMETERS
       REDEFINES HBRA-RA-INIT.
       15 HBRA-RA-PARMS  OCCURS 32.
          20 HBRA-RA-PARAMETER-NAME     PIC X(48).
          20 HBRA-RA-DATA-ADDRESS       USAGE POINTER.
          20 HBRA-RA-DATA-LENGTH        PIC 9(8) BINARY.
    10 HBRA-RESERVED.
      15 HBRA-RESERVED05            PIC X(12).
      15 HBRA-RESERVED06            PIC X(64).
      15 HBRA-RESERVED07            PIC X(64).
      15 HBRA-RESERVED08            PIC X(128).
      15 HBRA-RESERVED09            PIC X(132).
```

The entire HBRA-CONN-AREA must be passed on each of the three API calls to zRule Execution Server for z/OS. The following sections describe the important elements of this structure.

HBRA-CONN-RETURN-CODES

The HBRA-CONN-RETURN-CODES element provides response and reason codes to the requested API call. If these responses do not have a zero value on their return from an API, the user documentation provides more details about the error that occurred.

HBRA-CONN-RULEAPP-PATH

The HBRA-CONN-RULEAPP-PATH element is important for the data structure. This value is used by zRule Execution Server for z/OS to identify which decision to execute on the specific data. After a decision is deployed to zRule Execution Server for z/OS, the value for the RULEAPP-PATH can be identified by logging on to the administrator console and viewing the deployed decision.

The RULEAPP-PATH uses this structure:

/<RULEAPP-NAME>/<RULEAPP-VER>/<RULESET-NAME>/<RULESET-VER>

The *RULE-APP-VER* and *RULESET-VER* are generally required only if the client wants to execute a particular version of a decision that is deployed to zRule Execution Server for z/OS. In most cases, this will be the latest version of the decision. In this case, the path is simplified to this structure:

/<RULEAPP-NAME>/<RULESET-NAME>

For example, the Miniloan example that is supplied with Operational Decision Manager uses this structure:

```
/zRulesMiniLoanDemoRuleApp/zRulesMiniLoanDemo
```

HBRA-RESPONSE-AREA

If any text messages or warnings are returned by the Java portion of the server, they are returned in the HBRA-RESPONSE-AREA data area to help you diagnose any problems. A preferred practice is to write this area out to an application log if a nonzero return code is received.

HBRA-RA-PARAMETERS

The HBRA-RA-PARAMETERS section is required only on the HBRRULE API call. The HBRA-RA-PARAMETERS section provides the user data that is used to evaluate the decision. The structure is a list of triplets to pass the parameter name, a pointer to the data in working storage, and the length of the data:

- HBRA-RA-PARAMETER-NAME is the name of the parameter as it is known to the decision. This parameter name is defined as part of the rule-authoring process. The parameter name is case-sensitive and must be added as a character string, exactly as it is defined in the ruleset.
- HBRA-RA-DATA-ADDRESS is a pointer to the location of the parameter in your working storage. Normally, this element points to the address of the 01-level element in your copybook that you imported to define this parameter to the decision. It is normally set like the following example:

  ```
  set HBRA-DATA-ADDRESS(1) to address of Borrower
  ```
- HBRA-RA-DATA-LENGTH defines the length of the data structure to which the HBRA-DATA-ADDRESS points. Ensure that you set this element correctly so that all necessary data is passed across to the decision execution. You can use COBOL-intrinsic functions to calculate this value, for example:

  ```
  move LENGTH OF Borrower to HBRA-DATA-LENGTH(1)
  ```

When the data definition depends on the table structure, ensure that the supplied length of the data structure is calculated assuming the maximum length of any tables.

5.2.2 HBRCONN API call

You use the HBRCONN API call to establish a connection to the Rule Execution Server for z/OS server or for the Rules Execution Server for WebSphere Application Server on z/OS. The HBRA-CONN-AREA data structure is passed as a parameter to this call.

In CICS, when using a Rule Execution Server for z/OS stand-alone server, Rule Execution Server on WebSphere Application Server for z/OS, or the locally optimized Java virtual machine (JVM) server deployment, the connection call is made at startup or when the HBRC transaction is run and not during the HBRCONN call.

5.2.3 HBRRULE API call

The HBRRULE API call actually invokes the decision for evaluation. The HBRA-CONN-AREA data structure is passed as a parameter and must contain references to the ruleset parameter data that is required for evaluating the decision.

Multiple HBRRULE calls can be made within a single HBRCONN/HBRDISC pair.

5.2.4 HBRDISC API call

You use the HBRDISC API call to disconnect from the server after all decisions are evaluated for this application. The HBRA-CONN-AREA is passed as a parameter.

In CICS, HBRDISC does not disconnect the CICS region explicitly. Instead, for CICS, this is achieved by using the HBRD transaction.

5.3 Mapping from the COBOL copybook

This section describes the following topics:

- Structure of a COBOL-based rule project
- Supported COBOL data types
- Creating custom converters
- Mapping level-88 constructs into BOM domain types

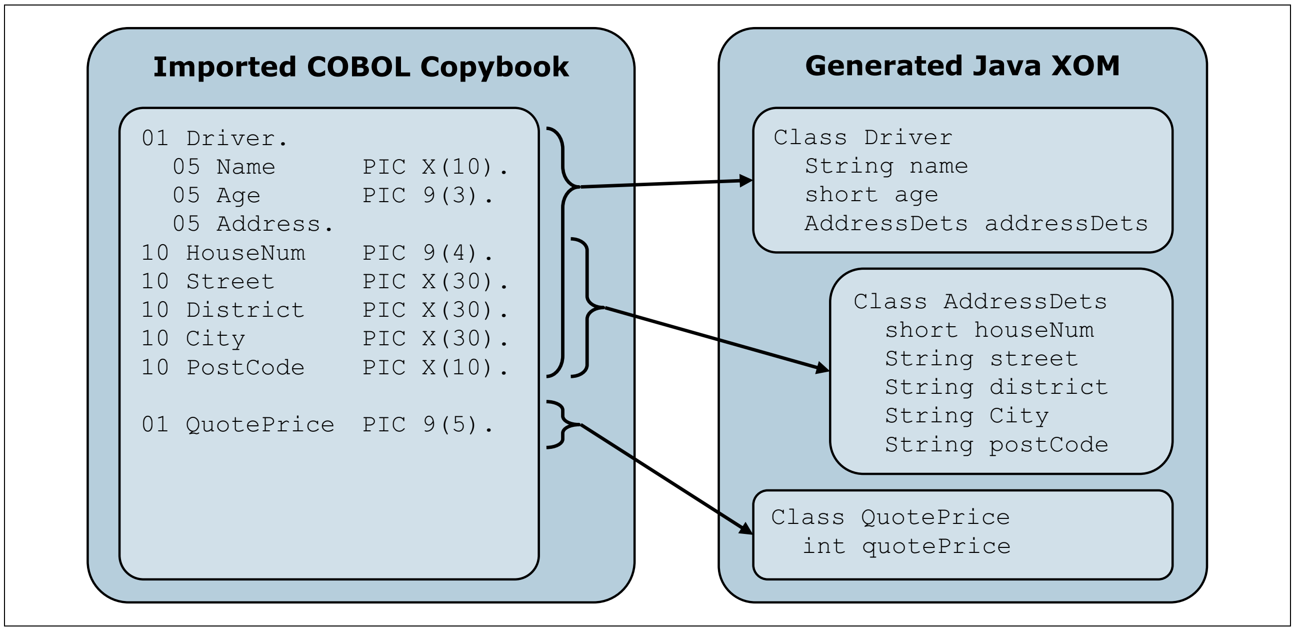
5.3.1 Structure of a COBOL-based rule project

In Chapter 3, “Getting started with business rules” on page 33, when importing a COBOL copybook, two Java projects are generated: the Java Execution Module (XOM) project and the marshaller project.

XOM project

The XOM project contains a Java representation of the structure of the data in the COBOL copybook. Each level-01 item in the copybook, both group and elementary items, is mapped to a Java class. Each non-level-01 group item is also mapped to a class.

Figure 5-1 is an example of this mapping.



```
01 Driver.
  05 Name      PIC X(10).
  05 Age       PIC 9(3).
  05 Address.
10 HouseNum    PIC 9(4).
10 Street      PIC X(30).
10 District    PIC X(30).
10 City        PIC X(30).
10 PostCode    PIC X(10).

01 QuotePrice  PIC 9(5).
```

```
Class Driver
  String name
  short age
  AddressDets addressDets
```

```
Class AddressDets
  short houseNum
  String street
  String district
  String City
  String postCode
```

```
Class QuotePrice
  int quotePrice
```

Figure 5-1 Generating a Java XOM from the COBOL copybook

Important: You can use the classes that are mapped from a level-01 group only as ruleset parameters that define the interface to the decision.

Marshalling

During the processing of requests from a COBOL application, the zRule Execution Server for z/OS server first converts the COBOL data into Java XOM objects. After executing the ruleset, the zRule Execution Server for z/OS then converts the Java objects back to COBOL data.

The marshaller classes are intended to be called by zRule Execution Server for z/OS only. You must not change a marshaller project because any change is likely to cause runtime errors.

5.3.2 Supported COBOL data types

This section describes the supported and unsupported COBOL data types.

Basic mapping

Table 5-1 lists the supported COBOL-type to Java-type mappings.

Table 5-1 Supported COBOL to Java mappings

COBOL type	COBOL usage and compile options	PICTURE string	Example	XOM Java type
AlphaNumeric	DISPLAY	Combination of A, X, and 9	PIC X(12)	String
Numeric	COMP-5 or BINARY, COMP, COMP-4 with TRUNC(BIN)	S9(1) through S9(4)	PIC S9 BINARY	short
		S9(5) through S9(9)	PIC S9(6) BINARY	int
		S9(10) through S9(18)	PIC S9(10) BINARY	long
		9(1) through 9(4)	PIC 9 BINARY	int
		9(5) through 9(9)	PIC 9(6) BINARY	long
		9(10) through 9(18)	PIC 9(10) BINARY	BigInteger
		9(10) through 9(18), with decimal (V or P)	PIC S999V9 BINARY	BigDecimal

	DISPLAY, COMP-3, PACKED-DECIMAL or BINARY, COMP, COMP-4 and not TRUNC(BIN)	S9(1) through S9(4)
		9(1) through 9(4)
				short

		S9(5) through S9(9)
		9(5) through 9(9)
				int

		S9(10) through S9(18)
		9(10) through 9(18)
				long

		S9(10) through S9(18) and 9(10) through 9(18), with decimal (V or P)		BigDecimal

	DISPLAY, COMP-3, PACKED-DECIMAL and ARITH(extend)	S9(19) through S9(31)
		9(19) through 9(31)
				BigInteger

	DISPLAY, COMP-3, PACKED-DECIMAL and ARITH(extend)	S9(19) through S9(31)
		9(19) through 9(31)
		Decimal (V or P)
				BigDecimal

DBCS	DBCS	G, B, or N with DISPLAY-1	PIC G(10)	String
InternalFloat	COMP-1			float
	COMP-2			double
Level 88				methods
National	NATIONAL	PIC N(8)		String
		PIC 999V9 SIGN LEADING SEPARATE, SIGN TRAILING SEPARATE		BigDecimal

Unsupported types

Table 5-2 lists the unsupported COBOL types. These types are not supported because there are no suitable Java types to which to map. Do not include these types in the import copybook.

If the copybook structure cannot be changed, for example, for compatibility with existing applications, consider this work-around. In the copybook for importing, change the data item to a corresponding ordinary alphanumeric or national type of the same length. Then, these data items are mapped to Java strings in the generated XOM class.

There are also two unsupported Occurs Depending On (ODO) table situations:

- ODO table within a fixed-length table
- ODO table sharing the ODO object

Consider using a fixed-length table to work around these limitations.

Table 5-2 Unsupported COBOL types

COBOL type	COBOL usage and compile options	PICTURE string	Example
Alphabetic	DISPLAY	A	PIC A(20)
AlphaNumericEdited	DISPLAY	A X 9 B 0 /	PIC XBX
NumericEdited	DISPLAY	B P V Z 9 0 / , . + - CR DB * cs	PIC 9B9
ExternalFloat	DISPLAY	+9 -9 0 . V E 9.	PIC +99V9E99
NationalExternalFloat	NATIONAL	PIC +9.9E+99	PIC NBN
NationalEdited	NATIONAL	PIC NBN PIC $9.9	

Converter

You can use type converters to change the Java type to which a COBOL data item is mapped. There are two built-in converters:

- String to boolean converter
- String to Date converter

You can also implement custom converters or set a XOM field to a custom-defined domain class.

5.3.3 Creating custom converters

In Chapter 3, “Getting started with business rules” on page 33, you learned about how to use the built-in type converters to map a COBOL data item to Java Boolean or Date type. There are cases when the built-in converter cannot meet your need. You can then write a custom converter.

Consider this example. In a COBOL program, instead of using only `T` to indicate true, the program also accepts `t`, `Y`, and `y` as true values. But the built-in Boolean converter can accept only one character as the true value. So, in this case, you need to write a custom converter.

A *custom converter* is a normal Java class. The class must be annotated with the TypeConverter annotation. In the Converter dialog, users can select those classes with only the TypeConverter annotation, along with the built-in converters.

You then need to implement an init method. This method is called by the run time to initialize the converter with user-defined properties. There is also an optional TypeConverterProperties annotation with which you can define the list of expected property keys. In this example, two keys are defined:

true-values	A comma-separated list of characters that indicate true
false-value	The default value that indicates false

In the init method, the values of these properties are retrieved. Then, the list of true values to the trueValues field is saved and the falseValue field is set to the false-value character. See Example 5-2.

Example 5-2 Custom converter code (part 1 of 2)

```
@TypeConverter
@TypeConverterProperties({ "true-values", "false-value" })
public class MyBooleanConverter {
   private List<String> trueValues;
   private String falseValue;

   public void init(Map<String, String> props) {
      String trueStr = props.get("true-values");
      trueValues = Arrays.asList(trueStr.split(","));
      falseValue = props.get("false-value");
   }
```

Now, you can define two conversion methods:

```
public <TargetType> convertToTarget(<SourceType> value)
public <SourceType> convertToSource(<TargetType> value) {
```

The <*SourceType*> is the default Java type that is directly mapped from COBOL data and the <*TargetType*> is the type that you want to use in the generated XOM. The convertToTarget is called during unmarshalling, and the convertToSource is called during marshalling.

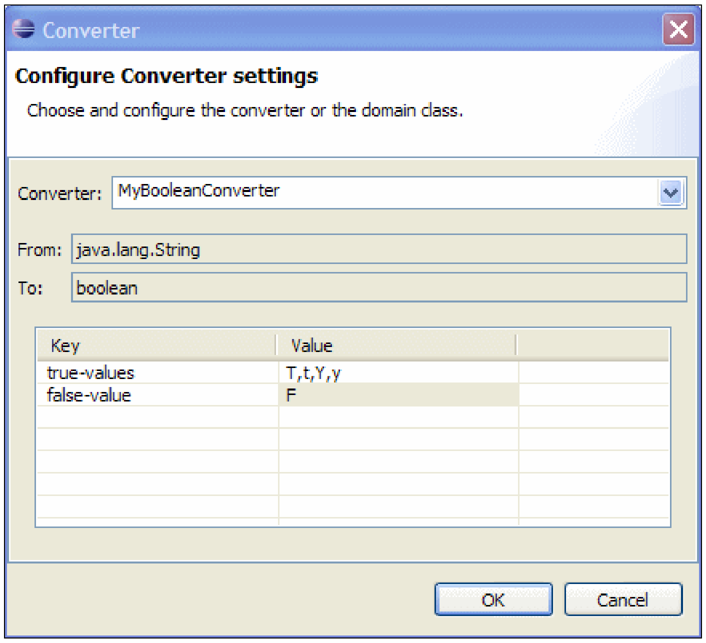
In Example 5-3, if the value that comes from COBOL is within the trueValues, the result is true. Any other values convert to false. During marshalling, if the Java value is true, the first of the possible true values is used, which is `T` in this case. If the Boolean value is false, the false value `F` is used.

Example 5-3 Custom converter code (part 2 of 2)

```
   public synchronized boolean convertToTarget(String value) {
      return trueValues.contains(value);
   }
   public synchronized String convertToSource(boolean value) {
      return value ? trueValues.get(0) : falseValue;
   }
}
```

After the custom converter is implemented, you must add the Java project to the XOM path of the rule project. Then, when adding a COBOL XOM, in the Converter dialog, you can choose the converter that you defined and set the property values (see Figure 5-2).

In the generated marshaller code, the user-provided properties in this dialog are sent as parameters in the call of the init method. Therefore, the converter is initialized before any conversion operation.



Configure Converter settings

Choose and configure the converter or the domain class.

Converter: MyBooleanConverter

From: java.lang.String

To: boolean

Key	Value
true-values	T,t,Y,y
false-value	F

Figure 5-2 Converter dialog with custom converter

5.3.4 Mapping level-88 constructs into BOM domain types

A *domain* can restrict the possible values that a type element in BOM can accept. During rule authoring, the editor suggests values according to the enumerated domains. A semantic check is also performed to check that the business rule does not use a value outside the defined domain.

The copybook that is shown in Example 5-4 was used in Chapter 3, "Getting started with business rules" on page 33.

Example 5-4 COBOL copybook

```
05  VEHICLE.
 10  VEC-ID                      PIC  X(15).
 10  MAKE                        PIC  X(20).
 10  MODEL                       PIC  X(20).
 10  VEC-VALUE                   USAGE COMP-1.
 10  VEC-TYPE                    PIC  X(2).
     88 SUV      VALUE 'SU'.
     88 SEDAN    VALUE 'SD'.
     88 PICKUP   VALUE 'PU'.
```

The vehicle type VEC-TYPE data item has three level-88 items, which define the valid values for this item: SU, SD, and PU. When importing this copybook into the Rule Designer, the level-88 items are mapped to methods. These methods are helpful to check the value of the field or to assign the correct value to the field, but they cannot prevent the field from being assigned incorrect values. Ideally, a user might use valid values only with the vehicle type, or the user can use actual vehicle types instead of codes to represent the vehicle types. A domain must be defined to meet this requirement. In the COBOL XOM import wizard, a domain converter to map the vehicle type to a Java domain type needs to be defined.

Follow these steps:

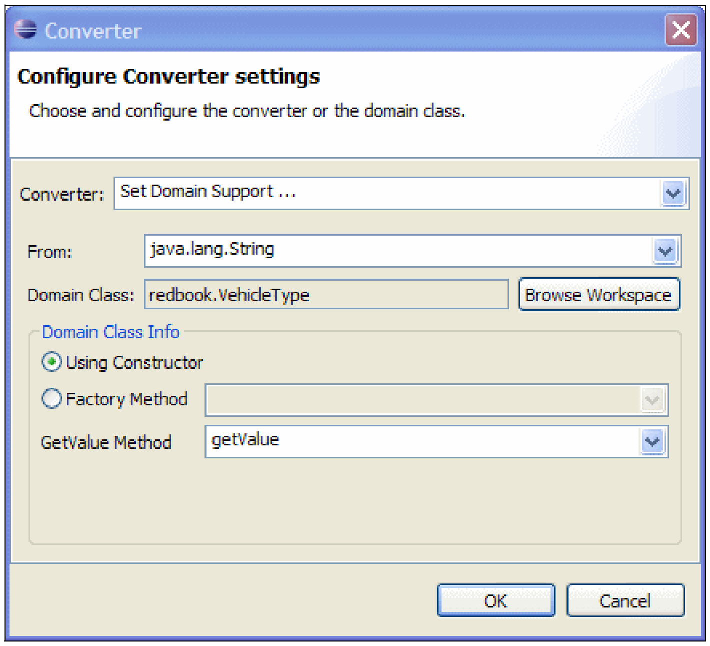
1. Implement a Java class as the XOM for the domain definition. Example 5-5 is the sample code. In this class, define a constructor with a string parameter. This constructor accepts the vehicle type codes and creates a VehicleType object. You must also implement a getValue method to retrieve the string code from the VehicleType object.

   Example 5-5 Java class for domain vehicle type

   ```
   package redbook;
   public class VehicleType {
     public final static VehicleType SUV = new VehicleType("SU");
     public final static VehicleType SEDAN = new VehicleType("SD");
     public final static VehicleType PICKUP = new VehicleType("PU");
     private String code;
     public VehicleType(String code) {
       this.code = code;
     }
     public String getValue() {
       return code;
     }
   }
   ```

2. In Rule Designer, add the project to the Java XOM path of the rule project.
3. When you import a COBOL copybook, define a converter for the vehicle-type item. In the dialog that is shown in Figure 5-3 on page 114, when you select **Set Domain Support** for the Converter field, you can select the From type. The From type is the Java type when the COBOL data is first unmarshalled and before the converter is applied.

   You need to provide the domain class name that you want to use. You can then provide detailed information for the domain class. Select **Using Constructor** to convert the string codes to the domain object and then select the **GetValue Method** to convert the domain object to string.



Converter

Configure Converter settings

Choose and configure the converter or the domain class.

Converter: Set Domain Support ...

From: java.lang.String

Domain Class: redbook.VehicleType

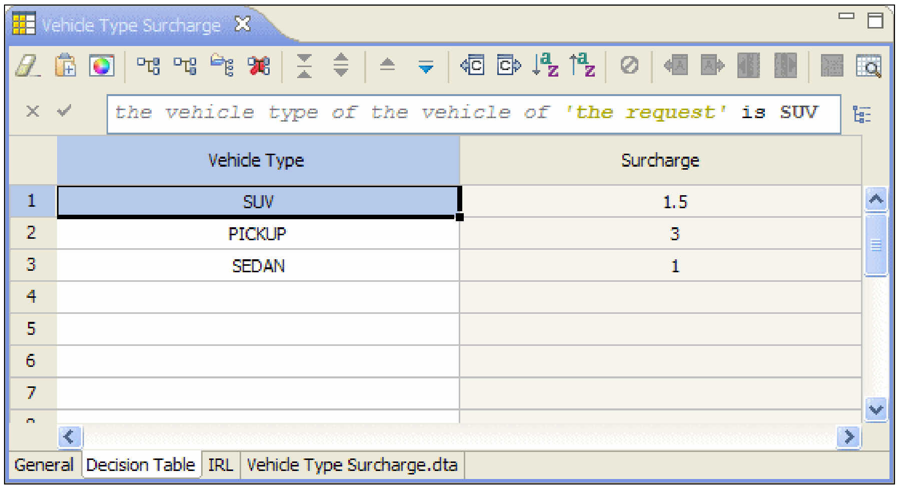
Domain Class Info

[✓ Using Constructor] Factory Method

GetValue Method: getValue

Figure 5-3 Converter dialog to set up the domain type

4. You must create a BOM entry for the domain class. Then, create a BOM entry for the COBOL XOM. With the correct verbalization, you can use the domain in rule authoring (Figure 5-4).



Vehicle Type Surcharge

```
the vehicle type of the vehicle of 'the request' is SUV
```

	Vehicle Type	Surcharge
1	SUV	1.5
2	PICKUP	3
3	SEDAN	1
4		
5		
6		
7		

Figure 5-4 A sample decision table using the COBOL domain

5. In the converter dialog, you can also use a static factory method instead of a constructor to define a domain converter. Example 5-6 is a sample implementation of a static factory method.

Example 5-6 Static factory method in the domain class VehicleType

```
public static VehicleType getVehicleType(String code) {
    if (SUV.code.equals(code))
      return SUV;
    else if (SEDAN.code.equals(code))
      return SEDAN;
    else if (PICKUP.code.equals(code))
   return PICKUP;
    else
   return null;
  }
```

5.4 Configuring the client application to reach the rule server

The job that runs the client application needs to be configured depending on the server that will make the business decision.

5.4.1 Batch application

For a batch application, the client application needs to be bound to the Operational Decision Manager stub:

```
//HBRLIB DD  DSN=++HBRHLQ++.SHBRLOAD
   INCLUDE HBRLIB (HBRBSTUB)
```

The job that runs the client application will also have the following line that tells the application which server to connect:

```
//HBRENVPR DD DISP=SHR,DSN=&HBRWDS..SHBRPARM(HBRBATCH)
```

5.4.2 IMS application

For an IMS application, the client application needs to be bound with the Operational Decision Manager stub for IMS:

```
//HBRLIB DD DSN=++HBRHLQ++.SHBRLOAD
   INCLUDE HBRLIB (HBRISTUB)
```

More information about working with IMS is in Chapter 12, “Configuring IMS to work with Operational Decision Manager” on page 221.

5.4.3 CICS application

For a CICS application, the client application needs to be bound to the Operational Decision Manager stub:

```
//HBRLIB DD DSN=++HBRHLQ++.SHBRCICS
   INCLUDE HBRLIB (HBRCSTUB)
```

More information about working with the CICS JVM is in Chapter 11, “Configuring CICS to work with Operational Decision Manager” on page 203.

5.4.4 WebSphere Optimized Local Adapters (WOLA) batch application

For a WebSphere Optimized Local Adapters (WOLA) batch application, the client application is bound as described in 5.4.1, “Batch application” on page 115. However, in addition, the job that runs the application needs to include the WOLA parameter file:

```
//HBRENVPR DD DISP=SHR,DSN=&HBRWDS..SHBRPARM(HBRWOLA)
```

More information about working with WOLA is in Chapter 13, “Configuring WebSphere Optimized Local Adapters support” on page 225.

6

Decision testing and simulation

This chapter provides an overview of the decision testing and simulation facilities that are available with Operational Decision Manager.

The following topics are covered in this chapter:

6.1 Making the right testing and simulation decisions

There are two critical aspects of all software development projects. One critical area is testing the business logic before deploying it into production. The other critical area is optimizing the decisions by running a what-if analysis on a set of reference data. You run the analysis to evaluate the effect of business logic modifications that are deployed into production.

Developers might use their traditional development toolset to run tests and simulations against a decision service. However, business users generally do not have access to these tools. Implementing and deploying a custom solution to provide them with these essential services can generate extra costs and delays to the project.

Because the business users are involved in the definition and the deployment of decision services that are developed with Operational Decision Manager for z/OS, dedicated testing and simulation tools are provided with the product using Decision Validation Services (DVS).

6.1.1 Decision Validation Services

Operational Decision Manager for z/OS provides Decision Validation Services (DVS) for handling test suites. This tool integrates using Rules Designer, Decision Center, and Rules Execution Server for z/OS (zRES). A summary of the various services is shown in Figure 6-1.

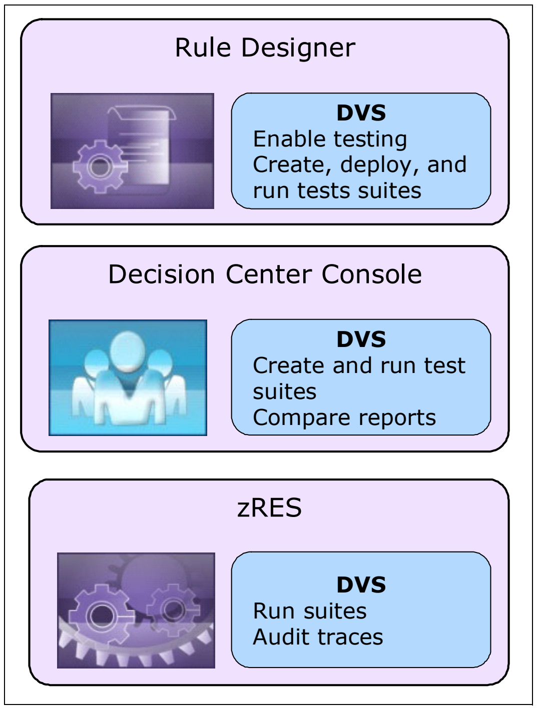


Figure 6-1 Services available in the different consoles of Operational Decision Manager

When a rule project is authored, developers can define a set of tests to be saved as artifacts of the rule project. They can then access a detailed execution report to analyze the results of the execution.

After a rule project is published into Decision Center, business users can define other tests. These are also saved as artifacts of the rule project and can be run against any version of the ruleset or any subset of its rules.

A detailed report is then available to them and includes the following information:

- Various levels of detail are available in the execution reports. A business user can generate a Microsoft Excel file that contains all the inputs and outputs of the decision service under test. The business user can reuse the file to generate custom reports using familiar business intelligence tools.
- Because the same tests and simulation scenarios can be run against various versions of the same ruleset, business users can compare two or more execution reports to spot the differences between them.
- A public Java API is available to author custom formats of tests and simulations and deploy them into Decision Center to make them available to business users. Rule Designer provides a set of dedicated editors and a wizard to streamline both the development and the deployment of a customization.
- A common use case for customization is the deployment of a custom scenario format that can run simulations using reference data stored in an enterprise database or a set of pre-existing reference files.

Scenario suite: Operational Decision Manager for z/OS uses the term *scenario suite* to refer to a set of test cases or a set of simulation cases that can be run against a ruleset. A *scenario suite* is defined as a set of scenario cases. Each scenario defines the value of the decision service inputs and optionally a set of expectations about the output (if the scenario suite is a test suite).

6.1.2 Verifying the business logic implementation by testing

Decision Verification Services assists with the initial verification of business logic by the IT department, making it easier for testing to be tracked, and also easing future regression testing by this department. Because the suite is specifically tailored toward Operational Decision Manager, the layout is ideally suited to such testing.

The availability of Decision Verification Services in the Decision Center means that during rules development, the business team can be involved, allowing them to verify that the tools exactly meet the needs of their clients. Furthermore, through a rule lifecycle, changes can be verified by the business team without involving the IT team.

Figure 6-2 shows a typical environment where a developer develops and tests rules locally and then tests the product on a development Rule Execution Server for z/OS. The rule projects are then passed to the business team who can test them from Decision Center and modify the rules. If necessary, the business team can request assistance from the development department where either more debugging is required or if a much larger modification is required in a rule project.

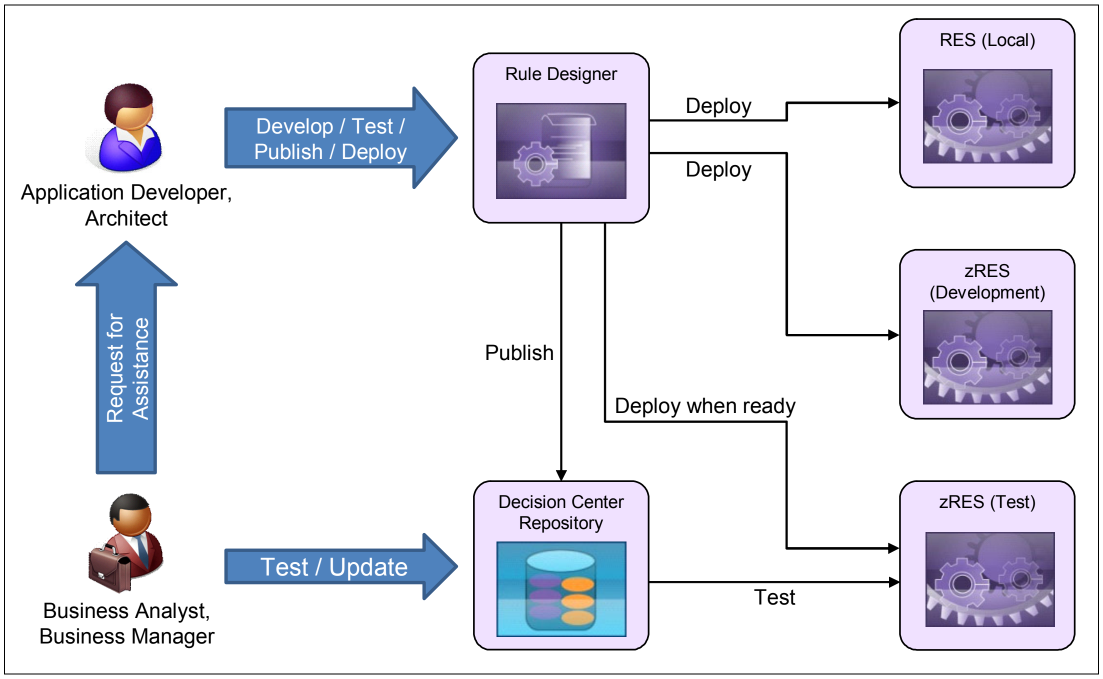


Figure 6-2 Example of the lifecycle of a rule including testing

6.2 Types of scenario suites

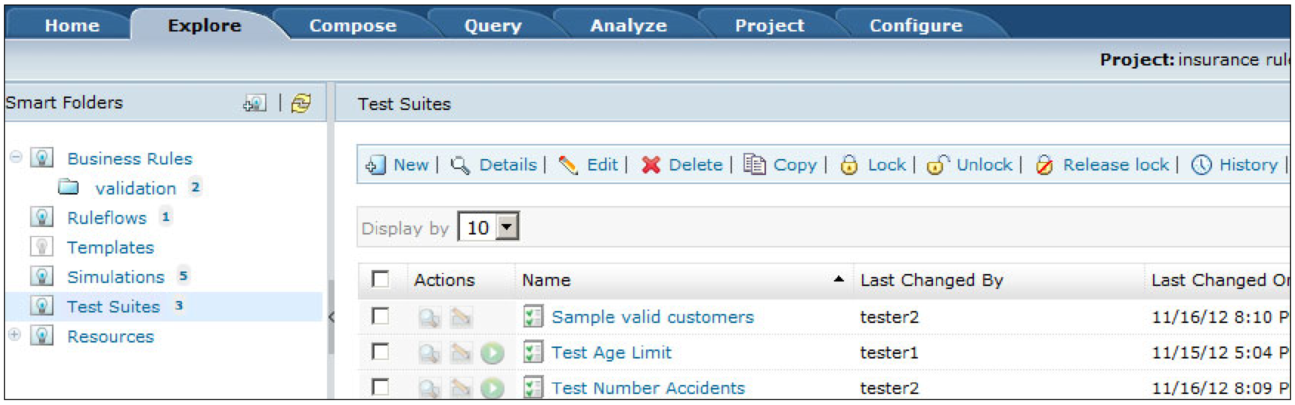
Scenario suites are customized by using the Scenario Service Provider (SSP). More details about the SSP are in 15.3, "Service scenario provider and key performance indicator architecture" on page 270. This section describes the different types of testing suites that are available and their uses.

6.2.1 Test suites

The first type of scenario suite that can be created with Operational Decision Manager for z/OS is the test suite, which defines a list of tests to run against a ruleset. Each test suite is defined as a list of test cases, and each test case is defined by these characteristics:

- Name
- The value of the input parameters to submit to the decision service
- An optional list of expectations about the output parameters of the decision service (for example, "the price of the proposal equals …", "the reward level of the customer is …", and so on)
- An optional list of expectations about the execution details

Figure 6-3 shows a list of test suites in the Explore tab of the Decision Center console.



Project: insurance rul

	Actions	Name	Last Changed By	Last Changed O
☐		Sample valid customers	tester2	11/16/12 8:10 P
☐		Test Age Limit	tester1	11/15/12 5:04 P
☐		Test Number Accidents	tester2	11/16/12 8:09 P

Figure 6-3 List of test suites in the Explore tab of the Decision Center console

Execution details provide detailed information about what is happening in the rule engine during the execution of a ruleset. When the user who defines a test suite also implements the ruleset, the user can define tests on the following execution details:

- The lists of rules that were or were not executed during the execution of the ruleset
- The total number of rules that were or were not executed during the execution of the ruleset
- The lists of tasks that were or were not executed during the execution of the ruleset
- The total number of tasks that were or were not executed during the execution of the ruleset
- The duration of the ruleset execution, which can be used for basic performance testing

In this way, the decision service author can test, for example, whether certain values for the input parameters trigger the execution of certain rules.

Initial testing

The first test suites that are generally run against a decision service are the tests that verify that the decision service behaves as expected before its deployment to production.

After the signature of a decision service has been defined, Decision Verification Services supports the creation of a test suite to support test-driven development. Operational Decision Manager test suites can be initially created with empty expectations. A first “dry” run of the test suite against the ruleset helps to populate these expectations by using copy and paste to transfer them from an execution report to the test suite definition in Microsoft Excel.

After successful testing, a decision service can be deployed into production.

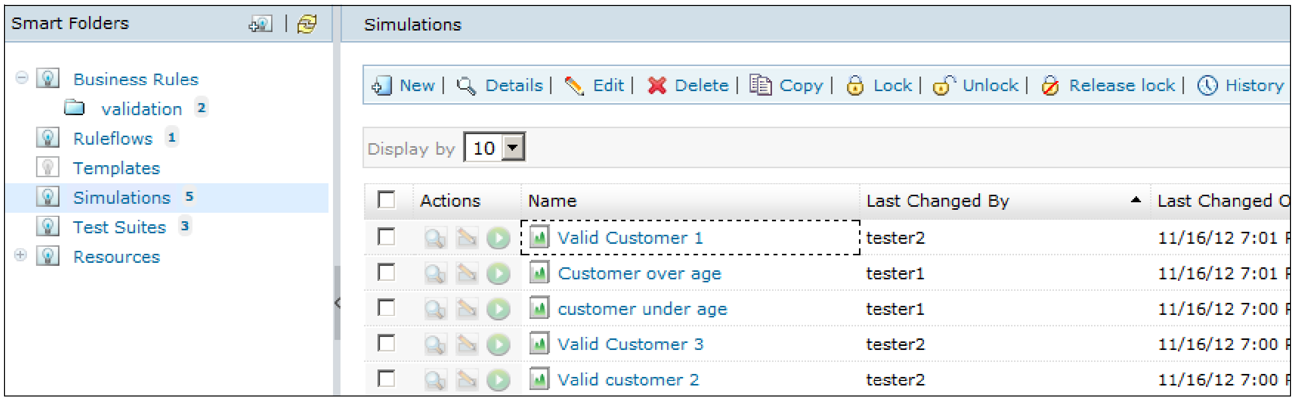
Regression testing

Subsequent changes to the decision service logic generally imply that you perform regression testing on the updated decision service before deploying these changes into production. This way, you can ensure that the new business logic does not cause unwanted side effects.

By providing versioning of the test suites and an execution reports comparison feature, Operational Decision Manager for z/OS allows users to easily run regression testing campaigns before deploying an update of the business logic of a decision service. Rules can be tested easily by business users, as well as developers, allowing for easy updates to the rule artifacts.

6.2.2 Simulation

Simulation is the second type of scenario suite that you can create with Operational Decision Manager. Figure 6-4 shows a list of simulations that are displayed in the Explore tab of the Decision Console.


Smart Folders

- Business Rules
  - validation 2
- Ruleflows 1
- Templates
- Simulations 5
- Test Suites 3
- Resources

Simulations

New | Details | Edit | Delete | Copy | Lock | Unlock | Release lock | History

Display by 10

Actions	Name	Last Changed By	Last Changed O
	Valid Customer 1	tester2	11/16/12 7:01
	Customer over age	tester1	11/16/12 7:01
	customer under age	tester1	11/16/12 7:00
	Valid Customer 3	tester2	11/16/12 7:00
	Valid customer 2	tester2	11/16/12 7:00

Figure 6-4 List of simulations in the Explore tab of the Decision console

A list of simulation cases can be defined to run against a decision service and a list of key performance indicators (KPIs) to be calculated during the execution. Key performance indicators measure rule performance and are described in “Key performance indicator” on page 134.

A simulation case is defined by its name and the values of the input parameters to submit to the decision service.

KPIs: KPIs are defined at the scenario-suite format level and are not defined by the business user.

In the simulation report, the values of all calculated KPIs are displayed. The same simulation can be run against separate versions of the same ruleset. Separate reports can be compared to check the differences between the KPI values. This capability allows the business user to fine-tune the content of the business logic between various executions of the same simulation until the desired KPI values are attained.

Regression testing can also be used to confirm that future versions of the rulesets are also providing the desired KPIs before deployment into production.

6.3 Development and authoring tools

This section describes the various development and authoring tools that are available by using DVS.

6.3.1 Rule Designer

You can perform the following operations that relate to testing and simulation from within Rule Designer by using the provided tooling:

- Check that a rule project is compatible with the Excel scenario suite formats, and correct it if necessary. A list of actions, which are required to perform this check, is provided by Rule Designer.
- Generate an Excel workbook to define a scenario suite for a rule project using one of the Excel scenario suite formats.
- Deploy the execution object module (XOM) (or repackage the Scenario Service Provider (SSP) with the XOM) before the first execution of a scenario suite for a ruleset.
- Create a custom scenario suite format manually or by extending the existing scenario suite formats using a set of specialized wizards and editors, as shown in Figure 6-5.
- Deploy a custom scenario suite into the SSP and Decision Center to make it available to the business users.
- Run a test suite by using a launch configuration. This launch configuration can access a remote server, which is required for testing on z/OS.
- Import a DVS archive produced by a Decision Center.

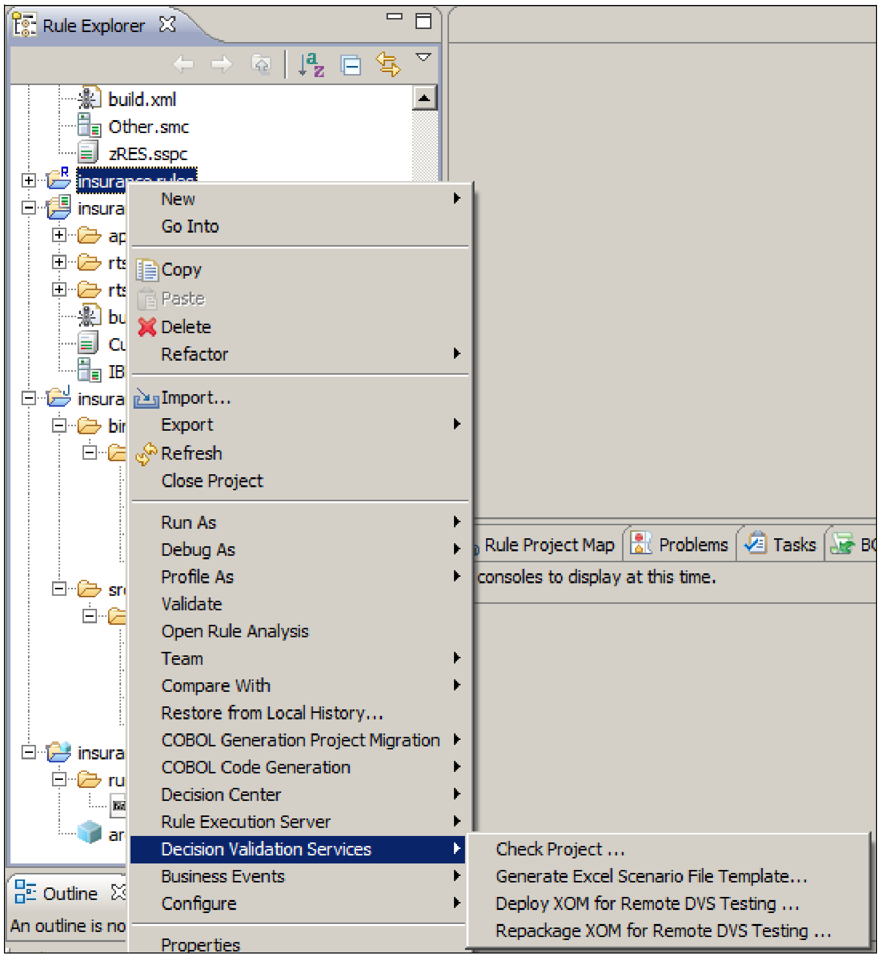


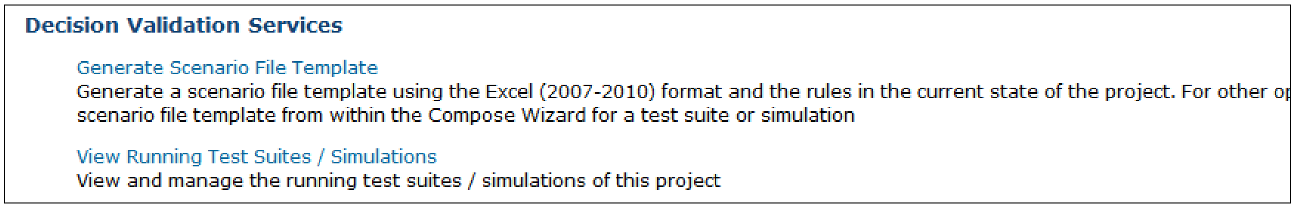
Figure 6-5 Rules Designer showing some of the DVS features

6.3.2 Decision Center

Before the Decision Center can test any rule project, that project must have been published to it by the Rules Designer. You can perform the following operations that relate to testing and simulation from within Decision Center by using the provided tooling:

- Run a test suite or simulation against all the rules of a rule project or against a subset of the rules.
- Enable or disable a custom scenario suite format for a rule project.
- Create a test suite.
- Create a simulation.
- Display a previous execution report for a test suite or simulation.
- Compare two execution reports side by side.

Test suites and simulations are created and edited on the Explore and Compose tabs of the Decision Center console. Other Decision Validation Services show on the Analyze tab, as shown in Figure 6-6.



Decision Validation Services

Generate Scenario File Template

Generate a scenario file template using the Excel (2007-2010) format and the rules in the current state of the project. For other op
scenario file template from within the Compose Wizard for a test suite or simulation

View Running Test Suites / Simulations

View and manage the running test suites / simulations of this project

Figure 6-6 Decision Validation Services on the Analyze tab of the Decision Center console

6.3.3 Rule Execution Server console and Rule Execution Server for z/OS console

You can perform the following operations that relate to testing and simulation from within Decision Center by using the provided tooling:

- Run remotely submitted tests against rule projects that have been deployed to the server.
- Audit traces that are stored in the Decision Center Warehouse database.

6.4 Testing and simulation architecture for z/OS decision services

This section describes the parts of the z/OS decision services that need to be considered for testing and simulation, including the artifacts, the formatting options, and the reports that are produced.

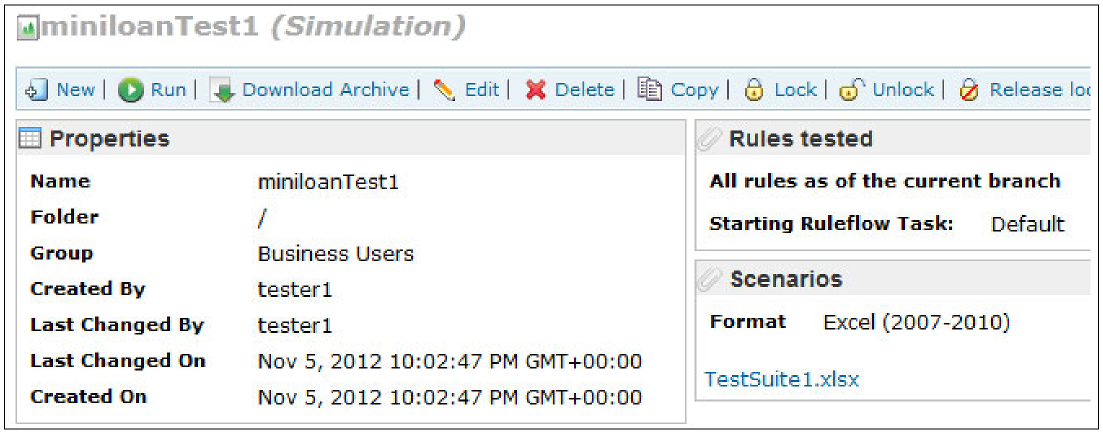
6.4.1 Test and simulation artifacts

This section details the different types of artifacts that are created.

Scenario suites

A *scenario suite* is an artifact that is created in Decision Center to define either a test suite or a simulation. A typical example is the simulation that is shown in Figure 6-7.

A scenario suite is attached to a rule project (or a ruleset), for which it defines tests or simulation cases. After its creation, you can run a scenario suite from within Decision Center at any time. Each execution creates an execution report that is stored along with the scenario suite in the Decision Center database.


miniloanTest1 (Simulation)
New | Run | Download Archive | Edit | Delete | Copy | Lock | Unlock | Release lo
Properties
Rules tested
Scenarios

Name	miniloanTest1
Folder	/
Group	Business Users
Created By	tester1
Last Changed By	tester1
Last Changed On	Nov 5, 2012 10:02:47 PM GMT+00:00
Created On	Nov 5, 2012 10:02:47 PM GMT+00:00

All rules as of the current branch

Starting Ruleflow Task: Default

Format Excel (2007-2010)

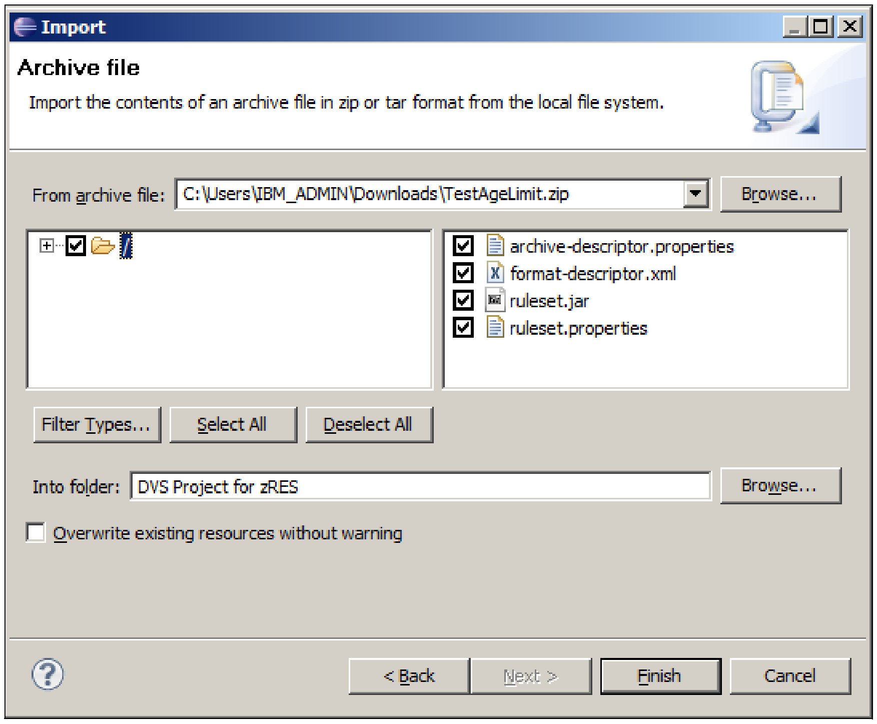
TestSuite1.xlsx

Figure 6-7 Sample simulation shown in Decision Center

Decision Validation Services archives

A *DVS archive* is an artifact that contains both a scenario suite and its associated ruleset in a single file that is easy to share. Scenario suite archives can be exported from Decision Center and imported into Rule Designer for execution and debugging by a developer.

Figure 6-8 on page 126 shows the import function, which is a standard part of the Rule Designer. This archive is used when errors occur in the ruleset during the execution of a scenario suite and the business user needs help from a support team to understand what happened.


Import

Archive file

Import the contents of an archive file in zip or tar format from the local file system.

From archive file: C:\Users\IBM_ADMIN\Downloads\TestAgeLimit.zip

- [x] archive-descriptor.properties
- [x] format-descriptor.xml
- [x] ruleset.jar
- [x] ruleset.properties

Into folder: DVS Project for zRES

- [ ] Overwrite existing resources without warning

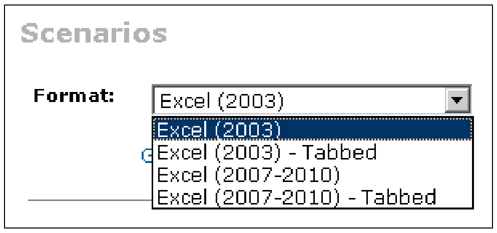
Figure 6-8 Importing a DVS archive into Rules Designer

6.4.2 Formatting options

This section describes the various types of formatting options that are available to create a test suite or simulation.

Scenario suite formats

When creating a test suite or a simulation in Decision Center, the business user is presented with a list of available supported formats by the scenario suite creation wizard (Figure 6-9). These formats are the scenario suite formats. They encapsulate the information that is required by the testing and simulation features to understand how a scenario suite is defined and needs to run.


Scenarios

Format: Excel (2003)

Figure 6-9 Selecting a scenario suite format

Scenario suite formats are created by using dedicated wizards and editors in Rule Designer. They are then published in Decision Center to be viewed by the business user.

After a scenario suite format is published in Decision Center, it needs to be enabled on a per project basis by a Decision Center administrator. A business user can then use the scenario suite format to create a test suite or simulation. For this reason, a custom scenario suite format with a list of dedicated custom KPIs that you define for running a simulation on a particular rule project is not visible in other projects.

A scenario suite format defines the following information:

- Fully qualified names for the rendered class and the launch configuration plug-in
- Fully qualified name for the scenario provider implementation
- The precision (number of meaningful digits after decimal point) to use when testing numbers
- List of execution details that can be tested when using this format to create a test suite
- Type of ruleset parameter values to be returned in an Excel file
- Location of the execution report for viewing in Decision Center
- List of KPIs to be calculated when a simulation is run using this format
- Scenario provider implementation to use at run time

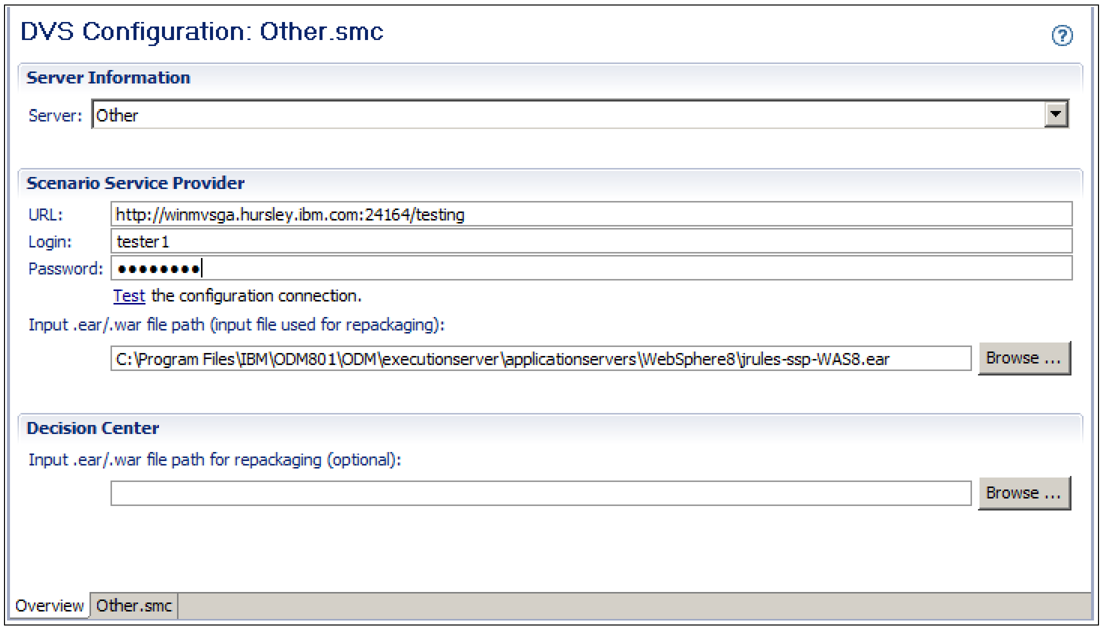
A sample scenario suite is shown in the simulation in Figure 6-7 on page 125.

Scenario provider

The scenario provider performs the following actions:

- Retrieves scenario data and returns a set of input parameter values for each scenario case.
- If the scenario is a test suite, it provides the set of tests to be run on each scenario case execution result.
- The scenario provider typically retrieves the scenario data and the test specification from a dedicated data store (a relational database or a file, for example) that is provided at creation time by the user running a test suite or simulation.

Information about the data store to use for a scenario provider is retrieved from a dedicated graphical user interface (GUI) in Decision Center and Rule Designer. For Decision Center, this user interface is defined by a scenario suite resource renderer class. For Rule Designer, this user interface is defined by an Eclipse launch configuration plug-in, as shown in Figure 6-10.


DVS Configuration: Other.smc

Server Information

Server: Other

Scenario Service Provider

URL: http://winmvsga.hursley.ibm.com:24164/testing
Login: tester1
Password: ●●●●●●●●

Test the configuration connection.

Input .ear/.war file path (input file used for repackaging):

C:\Program Files\IBM\ODM801\ODM\executionserver\applicationservers\WebSphere8\jrules-ssp-WAS8.ear

Decision Center

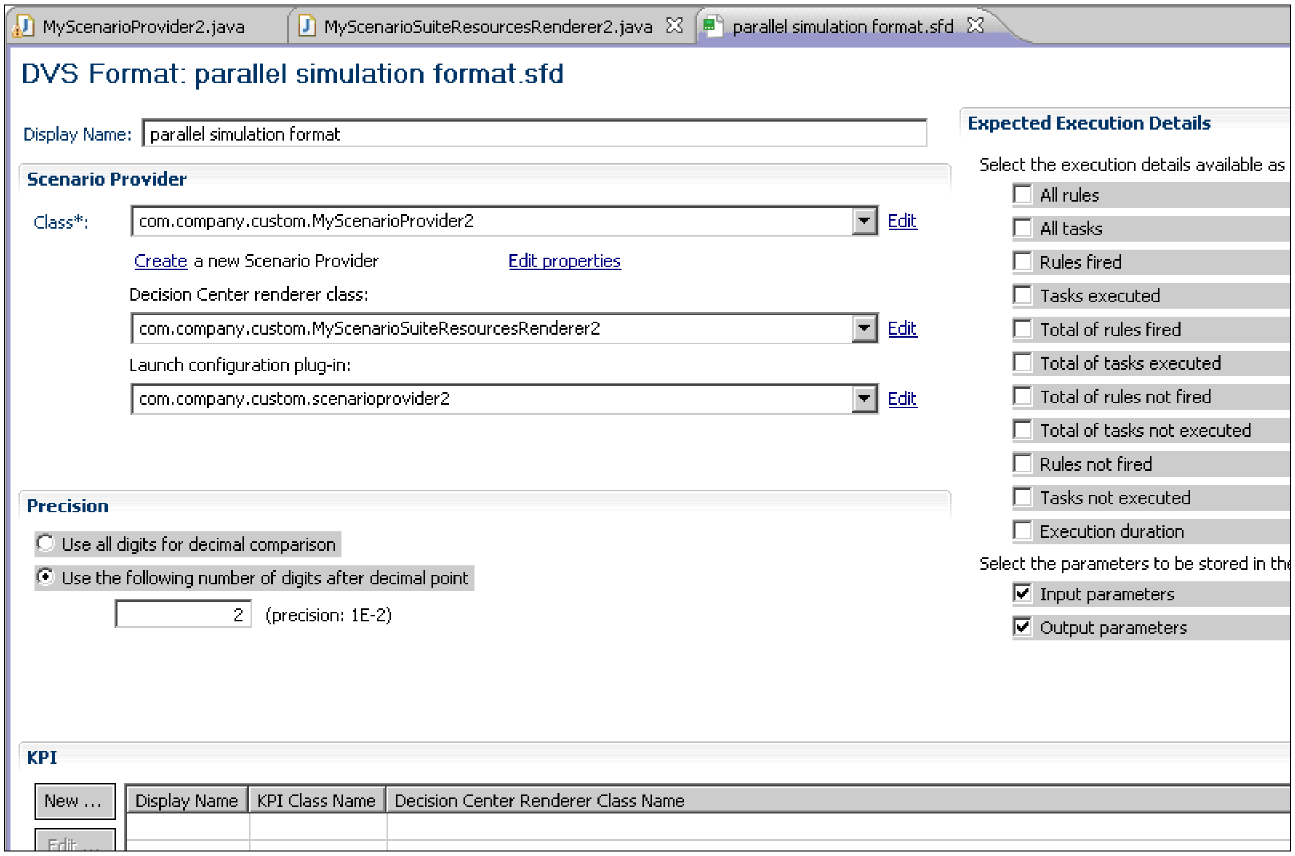
Input .ear/.war file path for repackaging (optional):

Figure 6-10 Launch configuration of the SSP in Rule Designer

For more details about the scenario provider, see 15.3, “Service scenario provider and key performance indicator architecture” on page 270.

Predefined scenario provider

Testing and simulation for DVS on z/OS comes with a predefined scenario provider. This scenario provider retrieves both the scenario data and test specifications from a Microsoft Excel workbook that was previously generated for a ruleset in Decision Center or in Rule Designer. This predefined scenario provider is referred to as the *Excel scenario provider* and is shown in Figure 6-11.


MyScenarioProvider2.java | MyScenarioSuiteResourcesRenderer2.java | parallel simulation format.sfd

DVS Format: parallel simulation format.sfd

Display Name: parallel simulation format

Scenario Provider

Class*: com.company.custom.MyScenarioProvider2 Edit

Create a new Scenario Provider   Edit properties

Decision Center renderer class:

com.company.custom.MyScenarioSuiteResourcesRenderer2 Edit

Launch configuration plug-in:

com.company.custom.scenarioprovider2 Edit

Precision

- Use all digits for decimal comparison
- [✓ Use the following number of digits after decimal point]

2 (precision: 1E-2)

Expected Execution Details

Select the execution details available as

- [ ] All rules
- [ ] All tasks
- [ ] Rules fired
- [ ] Tasks executed
- [ ] Total of rules fired
- [ ] Total of tasks executed
- [ ] Total of rules not fired
- [ ] Total of tasks not executed
- [ ] Rules not fired
- [ ] Tasks not executed
- [ ] Execution duration

Select the parameters to be stored in th

- [x] Input parameters
- [x] Output parameters

KPI

New ...
Edit ...

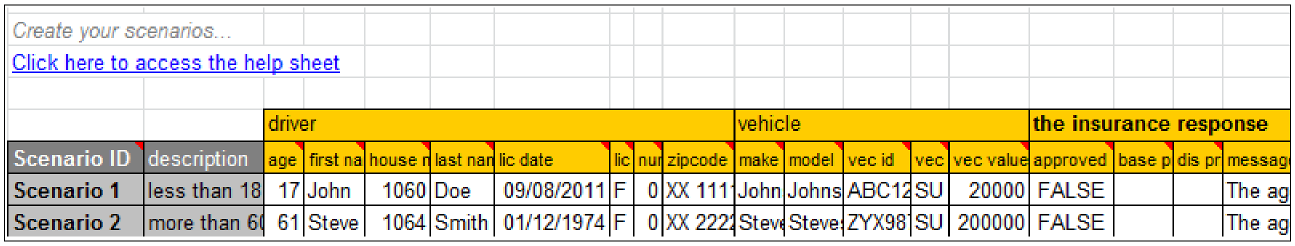
Display Name	KPI Class Name	Decision Center Renderer Class Name

Figure 6-11 Scenario suite format editor in Rule Designer

Available scenario suite formats

Several scenario suite formats are available in Decision Center after a standard installation of the product. In each scenario suite format, the user defines a test suite or simulation using a Microsoft Excel workbook. One page is dedicated to the definition of the scenario case. Each input parameter is defined in a table, one column per attribute, and the tests are defined in dedicated pages. Figure 6-12 on page 129 shows a page of such a workbook.

The default format is the Excel (2007-2010) format.



Create your scenarios...

Click here to access the help sheet

		driver								vehicle					the insurance response			
Scenario ID	description	age	first na	house n	last nan	lic date	lic	nur	zipcode	make	model	vec id	vec	vec value	approved	base p	dis pr	messag
Scenario 1	less than 18	17	John	1060	Doe	09/08/2011	F	0	XX 111	John	Johns	ABC12	SU	20000	FALSE			The ag
Scenario 2	more than 6(	61	Steve	1064	Smith	01/12/1974	F	0	XX 222	Stev	Steve:	ZYX98	SU	200000	FALSE			The ag

Figure 6-12 Sample spreadsheet with scenarios

All Excel formats are suitable for running test suites initially, but they do not define any KPIs. The KPIs are specific to a rule project. You must customize the KPIs for these formats before you use the formats for simulations.

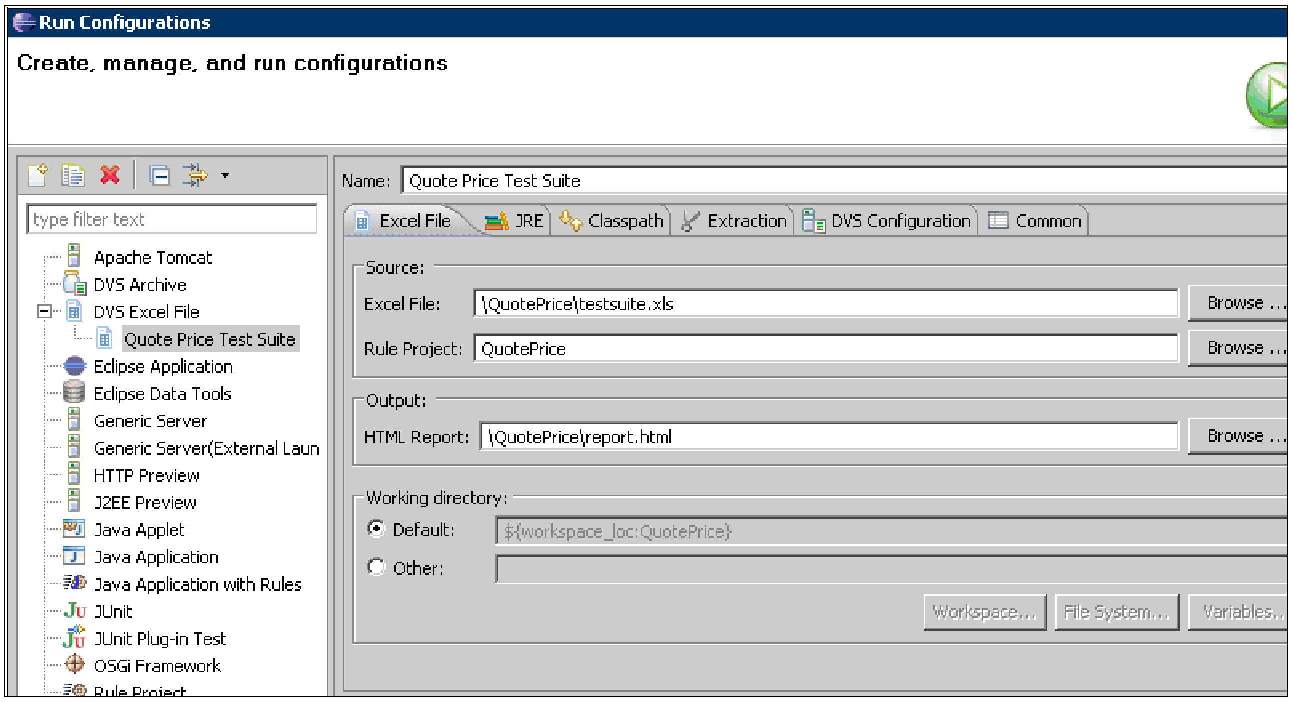
More information about the different supported Excel formats is available in the *Excel scenario files* topic of the Operational Decision Manager Decision Server V8.0.1 Information Center:

http://pic.dhe.ibm.com/infocenter/dmanager/v8r0m1/index.jsp?topic=%2Fcom.ibm.wodm.dserver.rules.designer.test%2Ftopics%2Fcon_excelfiles.html

DVS launch configurations

In Rule Designer, you can only run test suites (simulations can only be run from Decision Center). Two types of Eclipse launch configurations are provided to run test suites:

- The DVS Excel File launch configuration is provided so that you can run a test suite that is defined in a Microsoft Excel workbook locally or against a remote server, as shown in Figure 6-13.



Name: Quote Price Test Suite

Source:

Excel File: \QuotePrice\testsuite.xls

Rule Project: QuotePrice

Output:

HTML Report: \QuotePrice\report.html

Working directory:

(●) Default: ${workspace_loc:QuotePrice}

( ) Other:

Figure 6-13 Launch configuration for an Excel test suite in Rule Designer

- The DVS Archive launch configuration is provided so that you can run a test suite that is defined in a DVS archive that is retrieved from Decision Center locally or against a remote server.

In addition, you can create custom launch configurations using the Eclipse plug-in extension to run test suites, which are defined using a custom scenario suite format. They are not based on the Excel scenario provider.

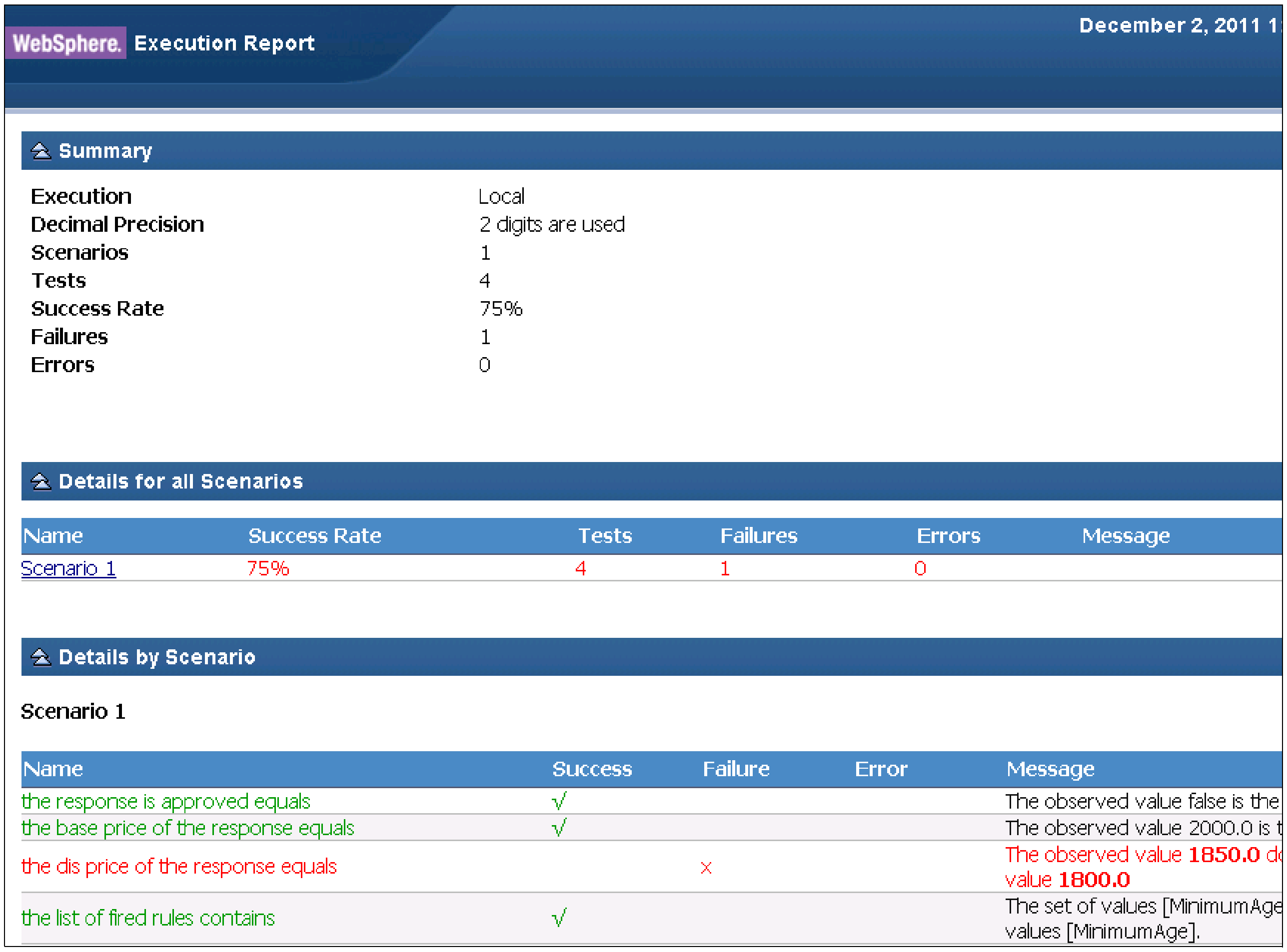
6.4.3 Test and simulation reports

Two types of test and simulation execution reports are provided. One type is for developers running test suites from Rule Designer. The more extensive type is for business users running test suites or simulation from Decision Center.

Execution reports for developers

The test suite execution report for developers is generated as an HTML file in the Rule Designer workspace after the execution of a test suite (Figure 6-14 on page 131). It includes the following information:

- Date and time that the test suite was run
- Type of execution (local or remote)
- Number of digits after the decimal point that is used for number comparisons
- Total number of scenarios in the test suite
- Total number of tests in the test suite
- Success rate for the test suite (percentage of test cases that were successfully verified)
- Total number of test failures
- Total number of test errors
- For each scenario, the name of the scenario, the success rate for the scenario, the number of tests in the scenario, the number of failures, and the number of errors
- For each test of a scenario, the name of the test, the result of the test execution (success, failure, or error), and a message that explains the result of the test execution


WebSphere. Execution Report

December 2, 2011 1

Summary

Execution	Local
Decimal Precision	2 digits are used
Scenarios	1
Tests	4
Success Rate	75%
Failures	1
Errors	0

Details for all Scenarios

Name	Success Rate	Tests	Failures	Errors	Message
Scenario 1	75%	4	1	0	

Details by Scenario

Scenario 1

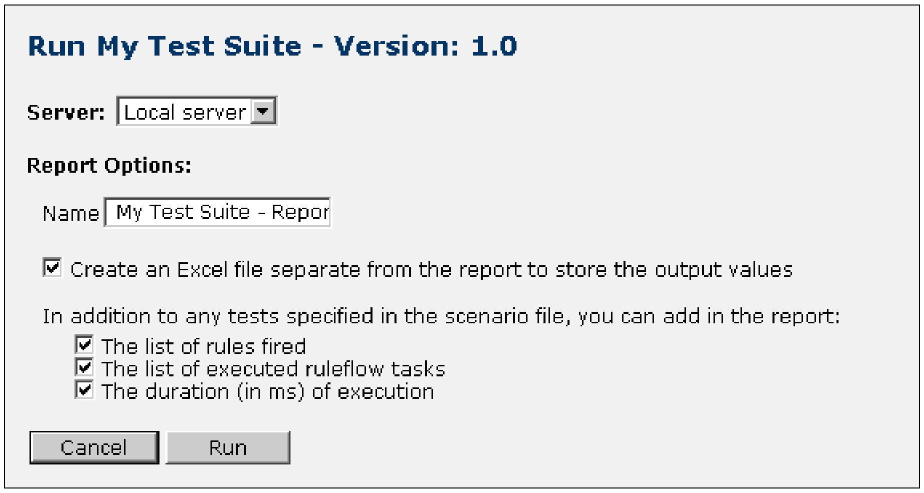
Name	Success	Failure	Error	Message
the response is approved equals	√			The observed value false is the
the base price of the response equals	√			The observed value 2000.0 is t
the dis price of the response equals		x		The observed value **1850.0** d value **1800.0**
the list of fired rules contains	√			The set of values [MinimumAge values [MinimumAge].

Figure 6-14 A test suite report in Rule Designer

Execution reports for business users

In Decision Center, an execution report is generated internally and stored in the Decision Center database along with the scenario suite information every time that a scenario suite is run. A report can be retrieved at anytime from the scenario suite artifact in the Decision Center Explorer. Various reports for separate versions of the same ruleset can be compared to help you detect regressions.

When running a scenario suite from Decision Center, many options are available to customize the content of the scenario suite execution report (Figure 6-15).


Run My Test Suite - Version: 1.0

Server: Local server

Report Options:

Name: My Test Suite - Repor

[✓] Create an Excel file separate from the report to store the output values

In addition to any tests specified in the scenario file, you can add in the report:

[✓] The list of rules fired
[✓] The list of executed ruleflow tasks
[✓] The duration (in ms) of execution

Figure 6-15 Detailed reporting options for a test suite in Decision Center

The additional reporting options are useful to analyze the results of an execution in detail. However, you must use them sparingly, because they have a large impact on the performance of the runtime execution.

A scenario suite execution report for business users displays the following information:

- Name of the report.
- Version of the test suite or the simulation for which the report was generated.
- If the scenario suite uses the Excel scenario provider, a link is provided in the report to download the Excel workbook.
- Date and time that the scenario suite was run.
- Name of the user that launched the scenario suite.
- The rule selector that was applied to the set of rules for running the scenario suite and the baseline that was used.
- The starting ruleflow task for the execution.
- The testing and simulation server against which the scenario suite was run.
- Total number of scenarios and the success rate for the execution.
- The following information is included for each scenario:
  - Name of the scenario
  - Status of the scenario
  - List of rules fired
  - List of executed ruleflow tasks
  - Duration of the execution, if requested
  - For test suites, a list of tests with the status of each test and a message explaining the status.
  - For simulation, the value of the KPI that was calculated during the simulation

6.4.4 Runtime components

At the heart of the runtime architecture for testing and simulation is the Scenario Service Provider (SSP) component. The SSP is a Java Platform, Enterprise Edition (Java EE) web application to be deployed into a WebSphere Application Server on z/OS application server (on the same server with Decision Center or another instance). The SSP provides a test and simulation runtime execution service to both Rule Designer and Decision Center. It can be deployed on multiple servers to dispatch the test and simulation load across several nodes. The diagram that is shown in Figure 6-16 shows the connections among the different runtime components.

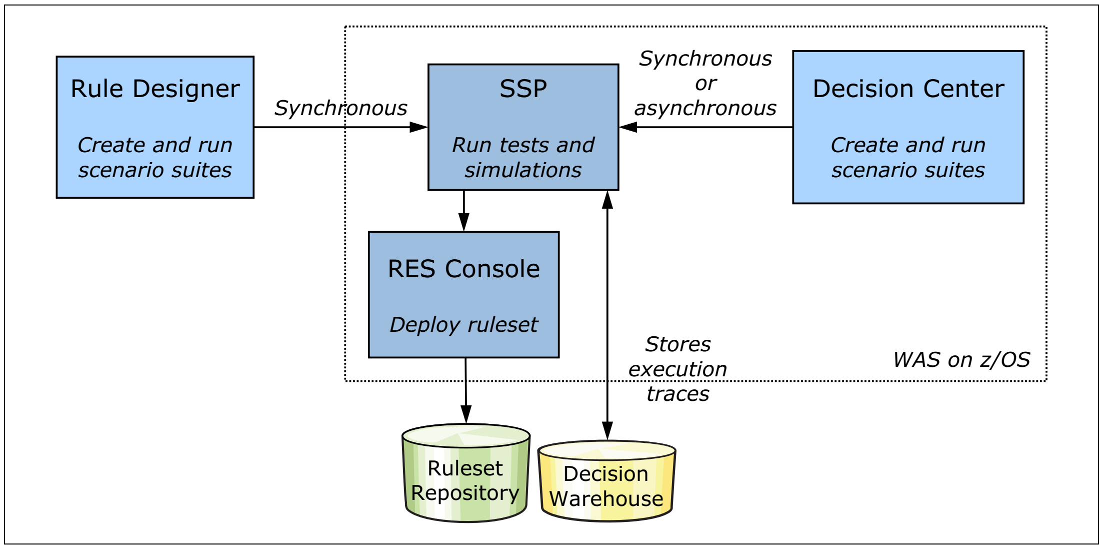
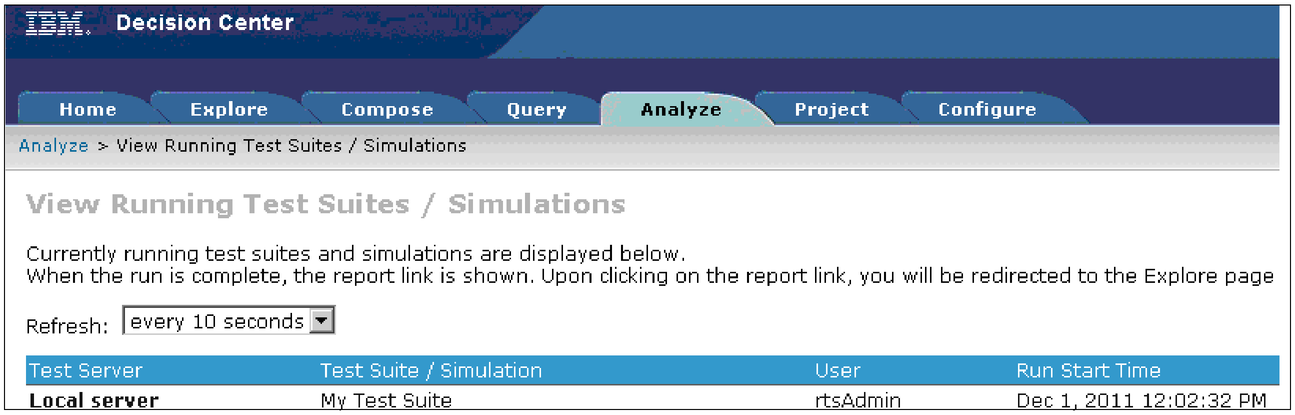


Figure 6-16 Testing and simulation runtime components

The SSP is a stand-alone component. In heavy workloads, for example, when your business users plan to run many tests and simulations on a resource-intensive decision service, you can size the SSP independently of Decision Center.

The SSP is a back-end component that does not provide a graphical user interface. A test or simulation execution is always triggered from either Rule Designer, Decision Center, or a Java application that uses the Java 2 Platform, Standard Edition (J2SE) testing and simulation runtime API.

The SSP provides both a synchronous and an asynchronous execution service. The asynchronous execution service is available from Decision Center and the testing and simulation runtime client API only. Figure 6-17 on page 134 shows the monitoring tool for an asynchronous scenario suite in Decision Center.



View Running Test Suites / Simulations

Currently running test suites and simulations are displayed below.
When the run is complete, the report link is shown. Upon clicking on the report link, you will be redirected to the Explore page

Refresh: every 10 seconds

Test Server	Test Suite / Simulation	User	Run Start Time
Local server	My Test Suite	rtsAdmin	Dec 1, 2011 12:02:32 PM

Figure 6-17 Monitoring asynchronous scenario suite execution in Decision Center

When a scenario suite execution is triggered, either synchronously or asynchronously, the SSP performs the following operations:

- Deploys the ruleset under test or simulation into the ruleset repository that is defined for the application server execution unit (XU)
- Executes each scenario case, using the input parameters that are defined in the scenario suite, using the XU component that is deployed in the application server
- If the scenario suite is a test suite, performs tests on the execution results
- If the scenario suite is a simulation suite, performs KPI calculations on the input parameters and execution results
- Undeploys the ruleset under test or simulation
- Returns an execution report

Important: To execute the ruleset under test or simulation, the SSP needs to have access to its Java Execution Module (XOM). Prior to running a scenario suite or a simulation on a ruleset, its XOM must be deployed in the XOM repository or repackaged into the SSP archive by using the tooling that is provided in Rule Designer. You execute this operation only one time, unless the XOM is updated, in which case, it must be redeployed.

Decision Warehouse database

Because the SSP is not a native z/OS component, it benefits from the Decision Warehouse database that is available for executions using Decision Server in a WebSphere Application Server on z/OS application server. The SSP uses the Decision Warehouse database to store additional reporting data when requested by a business user in Decision Center. Whenever a business user requests the creation of an Excel file with the values of the output parameters, or any execution traces, such as the list of rules fired or the execution time, the corresponding traces are stored in the Decision Warehouse. The information stays in the Decision Warehouse so that the Decision Center database is not overloaded with extra report data.

Key performance indicator

A *key performance indicator* (KPI) defines how to calculate the performance of a simulation. KPIs are used to compare the performance of the executions of rules. You can use multiple KPIs on the same simulation. They can then be configured to correctly present the results as required in simulation reports.

For information about the architecture and use of KPIs, see 15.3, “Service scenario provider and key performance indicator architecture” on page 270.

6.5 Testing and simulation lifecycle

The lifecycle of a project undergoing testing during its development is similar to the example that is shown in Figure 6-18.

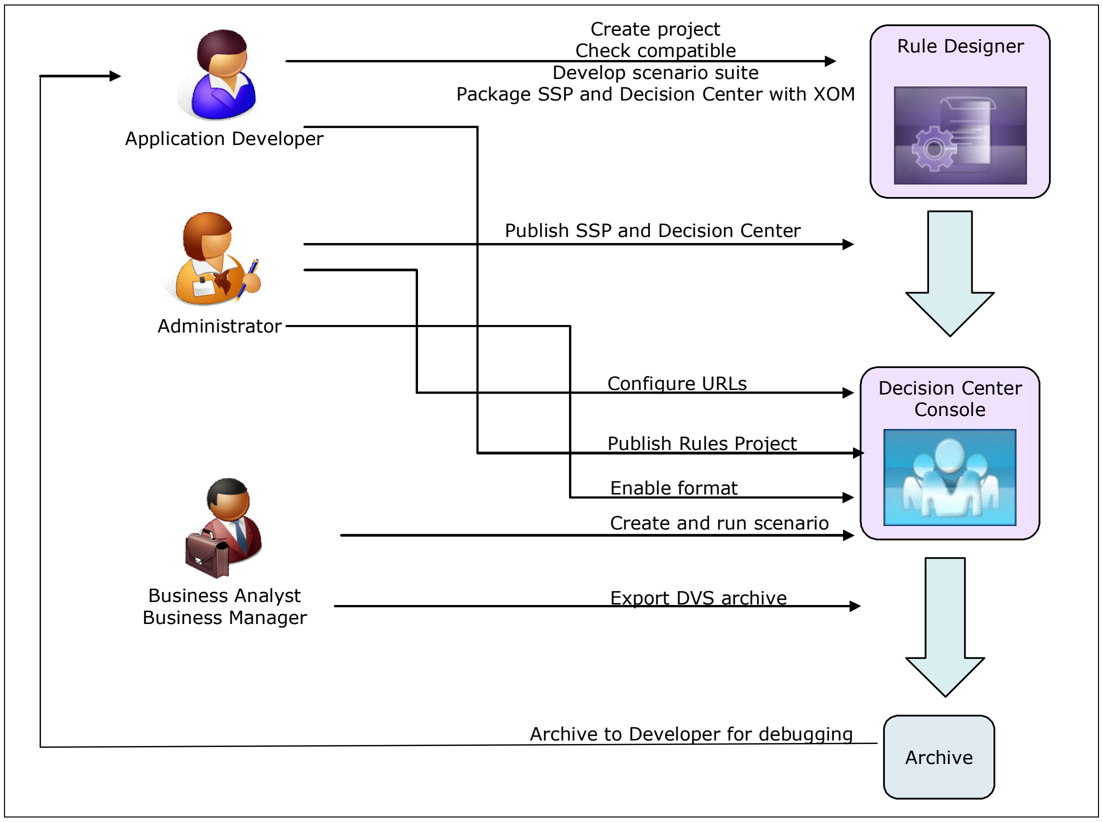


Figure 6-18 Testing and simulation lifecycle

6.5.1 Early development

The rule project is not available in Decision Center. The *Developer* performs these tasks:

1. Creates the rule project in Rule Designer.
2. Checks that the project is compatible with the scenario provider so that tests and simulations can be run (and corrects, if necessary).
3. Develops custom scenario suite formats with custom KPIs using the wizards and editors that are provided in Rule Designer.
4. Repackages the SSP and Decision Center with the XOM of the rule project and the client scenario format.

6.5.2 Project ready for deployment

The project is now ready to be published to Decision Center for the first time:

1. The *Administrator* publishes the SSP and Decision Center to WebSphere Application Server for z/OS.
2. The *Administrator* configures the URLs of the SSP servers that will be used to perform testing and simulation in Decision Center.

3. The *Developer* publishes the rule project into Decision Center.
4. The *Administrator* enables the custom scenario suite formats for the rule project in Decision Center.

6.5.3 Project deployed and enabled

The project is deployed and enabled in Decision Center, with the scenario suite formats to test it or run simulations. These steps occur:

1. The *Business user* creates and runs scenario suites.
2. When a problem occurs, the *Business User* exports the DVS archive and requests assistance from the *Developer* in debugging this archive.

7

Advanced topics for decision authoring

This chapter describes the authoring rules for deployment to z/OS. It does not explain general rule authoring. However, it does describe in more detail the mapping between the COBOL data structures that is passed into the decision run time and the Java structures that are used to execute the rules. This chapter also explains how you can customize that mapping.

This chapter describes starting a new project for deployment from both a COBOL copybook and an existing Java-based rule project. It then explains how you can extend the capabilities of the decision execution by adding custom methods into the business object model (BOM).

The following topics are covered in this chapter:

© Copyright IBM Corp. 2013. All rights reserved.

7.1 Starting from an existing Java-based BOM project

If rule projects are currently in production on distributed systems, and you want to migrate the rule application to the mainframe, you can enable the business object model (BOM) for a COBOL application and generate a COBOL copybook.

Important: Only a BOM that originated from a Java execution object module (XOM) is supported. A BOM that originated from an XML-based dynamic XOM is not supported.

7.1.1 Mapping Java data structures to COBOL

This section explains mapping Java data structures to COBOL.

Aggregation data structure

When mapping Java BOMs to COBOL, only BOM classes with aggregation relationships are supported. If there are object references, a simple hierarchical data structure is supported. Complex object graphs are not supported. For example, in the sample application that is used in this document, the Insurance class has a field called Vehicle of type Vehicle. The vehicle information is part of the insurance data, and it is a simple hierarchical data structure. This example is supported.

But, if the Vehicle class references Insurance either directly or indirectly through other classes, the data structure is not hierarchical. It contains loops. In this case, the BOM to COBOL mapping is not supported. In addition, class inheritance is not supported. The BOM must not include the following usage:

- Inheritance
- Loop reference, including self-reference
- Static attribute

Also, ensure that the BOM classes follow JavaBeans naming guidelines, such as well-formed getters and setters; otherwise, the generated marshaller classes contain incorrect code.

General mapping rule

A BOM class is mapped to a COBOL group. Fields of the basic Java type are mapped to child elementary data items. And, fields of the class type are mapped as subgroups (Example 7-1).

Example 7-1 A BOM with two classes

```
package xom;
public class Request {
    public xom.Driver primaryDriver;
    public xom.Driver secodaryDriver;
}
public class Driver {
public short age;
public string name;
}
```

When this BOM is mapped to a COBOL copybook, the primaryDriver and secondaryDriver fields are generated as two groups with the same structure (Example 7-2).

Example 7-2 Copybook with two similar groups

```
01 request.
           02 primaryDriver.
               03 age pic S9(5).
               03 name pic X(20) value SPACE.
           02 secodaryDriver.
               03 age pic S9(5).
               03 name pic X(20) value SPACE.
```

An array is mapped to a table, and a collection is mapped to a size data item and a table (Example 7-3).

Example 7-3 A BOM with an array and list

```
package xom;
public class Request {
    public xom.Driver[] drivers;
    public java.util.List vehicles domain 0,* class xom.Vehicle;
}
public class Driver {
public short age;
public string name;
}
public class Vehicle {
    public string vehicleId;
    public double vehicleValue;
}
```

The drivers field is an array, so the COBOL data item is a fixed-length table. The vehicles field is a list of Vehicle objects. So, in the generated copybook, vehicles-Num is used as the actual size of the table vehicles (Example 7-4).

Example 7-4 Copybook for array and list sample

```
01 request.
           02 drivers Occurs 10 Times.
               03 age pic S9(5).
               03 name pic X(20) value SPACE.
           02 vehicles-Num pic 9(9).
           02 vehicles Occurs 10 Times.
               03 vehicleId pic X(20) value SPACE.
               03 vehicleValue usage COMP-2.
```

Mapping the basic Java type

Table 7-1 lists the Java to COBOL mapping.

Table 7-1 Java to COBOL mapping

Java type	Default COBOL mapping	Configuration
byte	S9(3)	Sign and length USAGE BINARY, PACKED-DECIMAL, COMP-5
short	S9(5)	
int	S9(10)	
long	S9(18)	
java.math.BigInteger	S9(18)	
float	COMP-1	Sign and length USAGE BINARY, PACKED-DECIMAL, COMP-5, and COMP-1
double	COMP-2	
java.math.BigDecimal	S9(9)V9(8)	
java.lang.String	X(20)	X/N, length
java.util.Date	9(8) [*yyyyMMdd*]	9/X, date format
boolean		

You can change the default Java to COBOL mapping in the Default COBOL Type Setting tab on the COBOL Code Generation property page.

The following classes are unsupported Java types:

- Any Java classes, except the classes that are listed in Table 7-1
- The java.lang.Object class
- Classes that are defined in another BOM entry

7.2 Extending the capability of the rule execution with BOM methods

The Business Action Language (BAL) that is used to define rules is flexible and extensible. Generally, business rules are written from a vocabulary that is based on the structure of the data that is passed in, for example:

```
If the age of the borrower is less than 18
then ……
```

It is also possible to verbalize methods to be invoked from rules, as well. These methods can come from either methods from the imported Java XOM classes or be defined directly in the BOM as virtual methods, for example:

```
public void rejectTheLoan()
{
    this.approved = false;
}
```

The method in this example can be verbalized as reject the loan and then used for rule authoring:

```
If the age of the borrower is less than 18
then reject the loan ;
```

Here, the method ‘reject the loan’ can be used in place of the BAL statement:

```
make it false that the loan is approved ;
```

This approach greatly simplifies the rule authoring experience.

Although BOM methods are useful, ensure that the business decision can still be managed by the business and that the decision is still reusable across multiple platforms. The next section, 7.2.1, “Preferred practices for using virtual methods” on page 141, describes several of the preferred practices for BOM methods and shows you an example of using them.

7.2.1 Preferred practices for using virtual methods

To the IT-focused decision developer, the BOM methods might appear to be an attractive way to augment the capabilities of a business decision. However, you can negate the value of externalizing a business decision into a business rule engine if you do not use the BOM methods correctly. This section describes a few of the preferred practices to help you avoid misusing the BOM methods.

Do not bury business logic in the business decision

When you realize that you can access custom Java code from within a business decision call, it can be tempting to add business logic to the decision. Adding business logic to the decision is generally a bad idea. This statement might seem counterintuitive, because often the terms business logic and business rules are used interchangeably. When looking at the business decisions, you must consider them only the *rules* part of the business application.

Figure 7-1 shows a simplified representation of a business application.

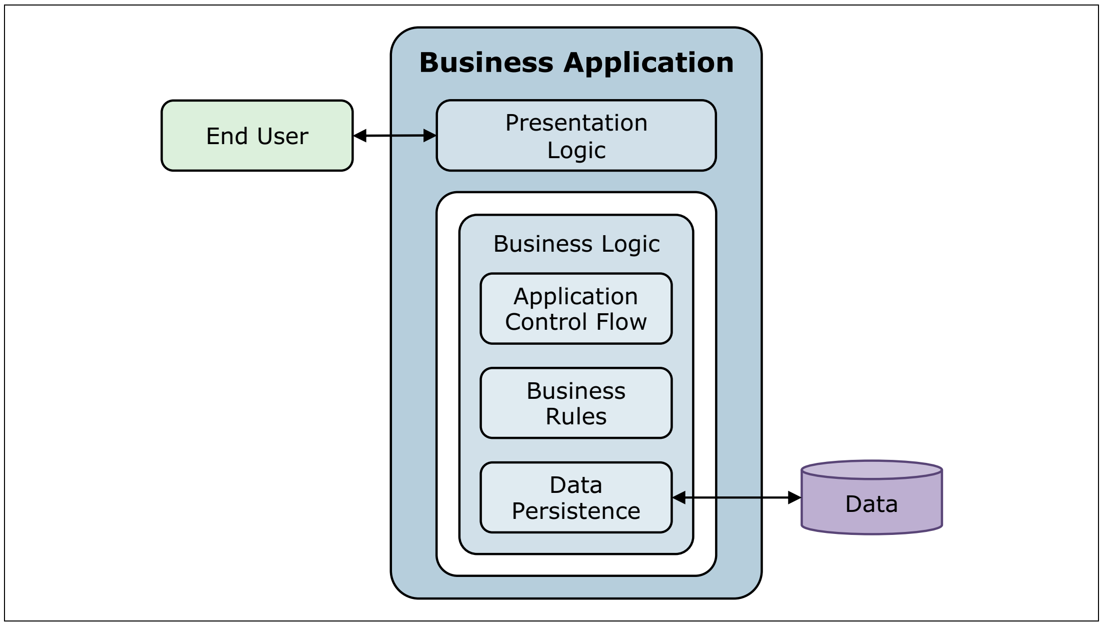


Figure 7-1 The structure of a business application

Figure 7-1 on page 141 shows that a business application is normally made up of the following elements:

- Presentation logic handling the user interaction
- Application control flow handling the flow of the logic through the application
- Business rules that are the implementation of the business behavior
- Data persistence layer that handles interaction with data sources

It is important to point out that the business rules do not interact with anything outside of the application. If you want to be able to hot-deploy new versions of decisions, you must be certain that changes to the business rules cannot break the application. If the business rules call out to external data sources, any changes must be tested within the full application, forcing a full regression test.

If the changes to the business rules change only the business behavior within the application, you can test the business rules in isolation to the full system. You only test to ensure that the rules implement the business requirement correctly. Ensure that the rules do not cause an issue for the application, for example, by forcing a divide by zero error. This level of testing is only appropriate if the business rules are encapsulated within the application.

Do not add platform-specific logic if sharing rules

In most cases, Java is platform-independent. However, it is still possible to code Java methods that only run in certain environments. For example, the JzOS Java libraries, which are part of the base Java Runtime Environment (JRE) 6.0 for z/OS, provide a collection of methods that are useful when coding Java on the mainframe. They contain methods for accessing z/OS resources and formatting data, plus other useful features. After you code a BOM method that uses the JzOS libraries, this business decision cannot be reused on a distributed platform. The only way to share is to create either two versions of the Java XOM or two separate rulesets, each containing a separate implementation of the BOM method, depending on where the method is coded. Both of these options lead to greater complexity in the rule management that is required to keep consistent decisioning across the platforms.

Use BOM methods sparingly

One of the key values of externalizing your business decisions as business rules is the ability to author them in natural language, making them accessible to the business team. If you use too many BOM methods, the result is recoding the business decision from the application into BOM methods in the business rules rather than as BAL rules that are accessible. In the extreme case, it is possible to code so much of the logic in BOM methods that the externalized business decision is no more accessible to the business team than the original business application from which the decision was extracted. The key is to use BOM methods sparingly where they add value to the rule authoring by simplifying the language and the logic required to author the decision.

BOM methods are useful in the following examples:

- Coding a formula that does not change but is used repeatedly within the decision.

  One example is calculating the after-tax income of a client where the tax amount is available as a variable to the BOM method.

- Simplifying the language that is required to perform a business operation to abstract from the data model.

  An example of this BOM method is where the business user has this capability:

  ```
  reject the loan rather than having to know the relevant part of the data model
  to alter to create this behavior
  ```

- Handling more complex data structures within the data model.

  The XOM model can contain more complex data structures, such as ordered lists. You can use a simple BOM method, such as addMessage(), to isolate the business user from this complex data structure.

7.3 Considerations for sharing rules between z/OS and distributed applications

One advantage of externalizing your business decisions is that you can identify decisions that are duplicated across multiple applications. The next logical step is to remove the duplication and manage the duplicated decisions as one decision to ensure consistency across the solution. Situations can occur where a decision is deployed for both a z/OS-based application and a distributed application. This type of deployment is possible, although certain considerations exist.

7.3.1 Sharing a COBOL-based project with Java applications

When you start from a COBOL copybook as your definition for the data that is passed into the decision (the XOM), the tooling initially generates a Java representation of the COBOL data structure. These Java objects are used at run time by the decision server for evaluating the business rules. After you import the copybook and develop your rules, the following artifacts are left:

- The copybook
- A Java project that represents the data from the copybook
- A Java marshaller project
- One or more rulesets that define the rules in the decision

All these artifacts are required to deploy the rules to the zRule Execution Server for z/OS environment, as shown in Figure 7-2.

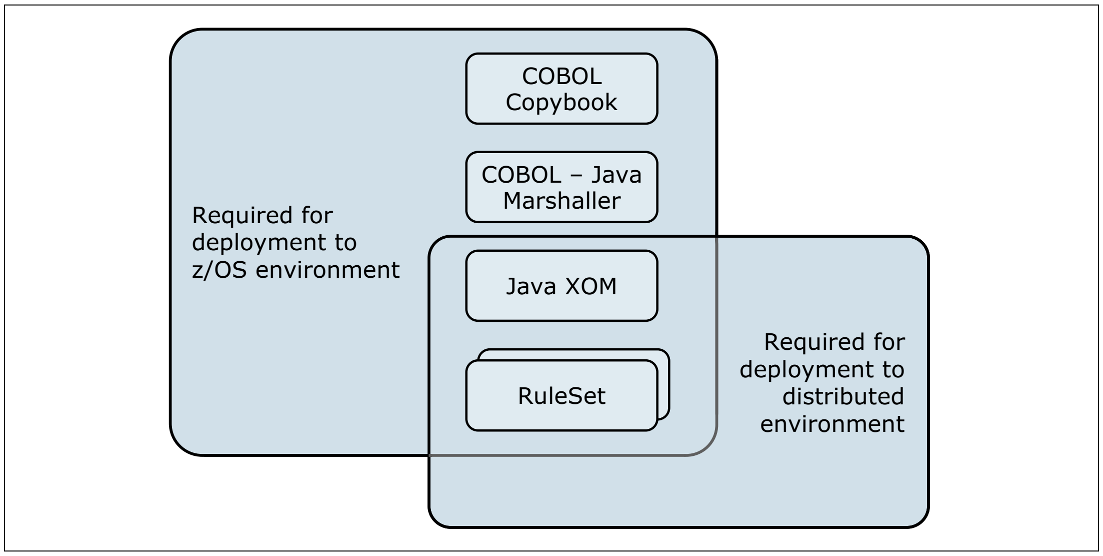

COBOL Copybook
COBOL – Java Marshaller
Required for deployment to z/OS environment
Java XOM
RuleSet
Required for deployment to distributed environment

Figure 7-2 Artifact deployment

If you want to reuse this decision in a distributed environment, deploy the Java project that was created from the COBOL and the rulesets together to a distributed version of Operational Decision Manager. In this case, you use the standard Java APIs to access the decision. The client passes the data into and out of the decision using the Java objects that are generated from the COBOL copybook.

You must not edit or change the generated Java classes in any way. Any change to the COBOL copybook requires the regeneration of the Java XOM. Any changes that you made are lost. In this example, consider the COBOL copybook as the master copy of the data model. Any required changes must be made to the COBOL copybook, and all other artifacts regenerated.

7.3.2 Sharing a Java BOM-based project with COBOL applications on z/OS

Section 7.1, “Starting from an existing Java-based BOM project” on page 138, describes the process of enabling a Java BOM-based project for use with COBOL applications on z/OS. When planning to enable a Java-based BOM project, it is important to consider the restrictions on the Java types that can be supported in this process. It is also important to note that new artifacts are created in this process. The most important artifact to the run time is the Java project that contains the code to marshal between the COBOL data structures and the Java used at run time. This artifact must be deployed to the zRule Execution Server for z/OS run time with the Java XOM and ruleset projects.

When any changes are made to the Java XOM or to the generated BOM, you must rerun the process to update the COBOL artifacts to synchronize them with the Java changes. In this case, consider the Java XOM as the master data model. You must not change the COBOL copybook after it is generated.

When using an existing Java project as a starting point to deploy to z/OS, ensure that no platform-specific code is in the Java XOM. As described in 7.2, “Extending the capability of the rule execution with BOM methods” on page 140, rules can invoke methods that exist on classes in the Java XOM. Ensure that if any methods are used in the rules, the methods do not invoke any platform-specific code. Java code is independent of any platform. However, Java code can become specific to a platform if it tries to access a data source that exists only in the solution on particular servers. For example, the solution might use a client record database that is hosted locally to the decision execution on IBM AIX® server1 but is not accessible to z/OS server2 due to the firewall configuration.

7.4 Authoring considerations for performance

When authoring the rules, consider the implications on the performance of the decision. For more information, see the IBM Redbooks publication *Proven Practices for Enhancing Performance: A Q & A for IBM WebSphere ILOG BRMS 7.1*, REDP-4775, which provides guidance about configuring and authoring for performance.

8

Decision Server events

This chapter enhances the request for a quote scenario that was introduced in 1.8, “Overview of the scenario used in this book” on page 11 using IBM Operational Decision Manager for z/OS to perform event pattern detection.

The following topics are covered in this chapter:

© Copyright IBM Corp. 2013. All rights reserved.

8.1 Scenario overview

In this version of the scenario, the request for quote application running on CICS is designed to emit (create) a business event each time that a quote is requested.

Using this event, Operational Decision Manager identifies the situation where a client submits more than three requests for a quotation within the same hour. When this situation is identified, Operational Decision Manager generates an action that notifies the sales team to follow up with the client to ensure that the policy is purchased.

Figure 8-1 shows the event processing overview for the request for a quote scenario.

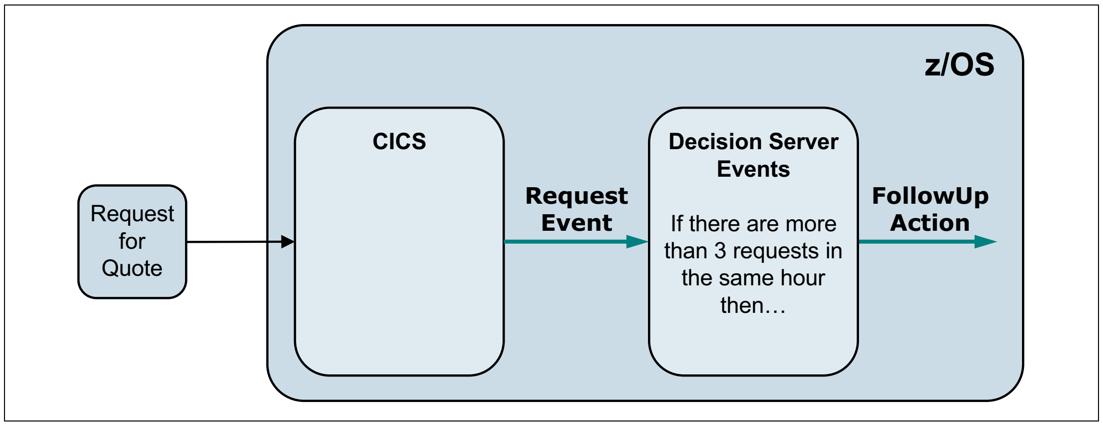


Figure 8-1 The event processing overview for the request for quote scenario

The business event that is generated for each quotation request is modeled on the original COBOL copybook, which is described in Chapter 3, "Getting started with business rules" on page 33. The original COBOL copybook is included in Example 8-1.

Example 8-1 The COBOL copybook describing a client's request

```
01  REQUEST.
    05  DRIVER.
        10  FIRST-NAME               PIC  X(20).
        10  LAST-NAME                PIC  X(20).
        10  ZIPCODE                  PIC  X(8).
        10  HOUSE-NUM                PIC  9(8).
        10  AGE                      PIC  9(2) USAGE COMP-3.
        10  LIC-DATE                 PIC  X(8).
        10  LIC-STATUS               PIC  X.
        10  NUMBER-ACCIDENTS    PIC  99.
    05  VEHICLE.
        10  VEC-ID                   PIC  X(15).
        10  MAKE                     PIC  X(20).
        10  MODEL                    PIC  X(20).
        10  VEC-VALUE                USAGE COMP-1.
        10  VEC-TYPE                 PIC  X(2).
            88 SUV     VALUE 'SU'.
            88 SEDAN   VALUE 'SD'.
            88 PICKUP  VALUE 'PU'.
```

Additional resource: You can find the `INSDEMO.cpy` copybook file in the additional information that is included in this book in the `code/Chapter3/CopybookBased` directory. See Appendix C, “Additional material” on page 335. You can ignore the 01 level RESPONSE structure, because it is required only for rule executions.

The copybook describes the data structure of both the driver and the vehicle in the scenario. The copybook is used in the Event Designer to build an event application. Section 8.2, “Building the event application” on page 147, describes the steps to build the application. Section 8.3, “Deploying the event application to the event run time” on page 154, describes how to deploy the event application to the Decision Server Events run time.

In 8.4, “Emitting events from CICS” on page 157, the CICS Event Binding Editor is used to build an event within a CICS bundle that tells CICS how to emit the business event when a client requests a quotation. Section 8.5, “Running the scenario” on page 168, then describes how to deploy a CICS COBOL application to trigger the emission of the request for quote event. Decision Server Events tooling is used to show the events that are received and the action that is fired.

Finally, 8.6, “Using connectors to receive events from various z/OS sources” on page 172 describes how business events can be received from other sources on z/OS, such as files and message queues.

8.2 Building the event application

This section explains how to use the Event Designer to build the event application. The Event Designer is a Decision Server Events component that supports the definition of the metadata layer that is required for business event processing (BEP). You can use the Event Designer to create all the building blocks for your event application, including events, business objects, actions, and event rules. The Event Designer is an Eclipse-based tool that can be run on a Microsoft Windows or Linux workstation.

8.2.1 Event project overview

The Event Designer uses an *event project* to store all the artifacts that are required for event processing. Figure 8-2 illustrates an overview of the artifacts that can be created in an event project.

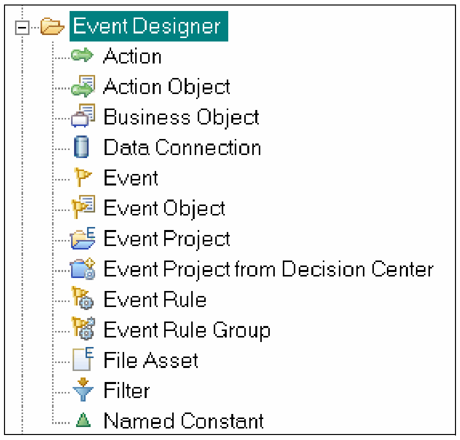

Event Designer
- Action
- Action Object
- Business Object
- Data Connection
- Event
- Event Object
- Event Project
- Event Project from Decision Center
- Event Rule
- Event Rule Group
- File Asset
- Filter
- Named Constant

Figure 8-2 Event project artifacts

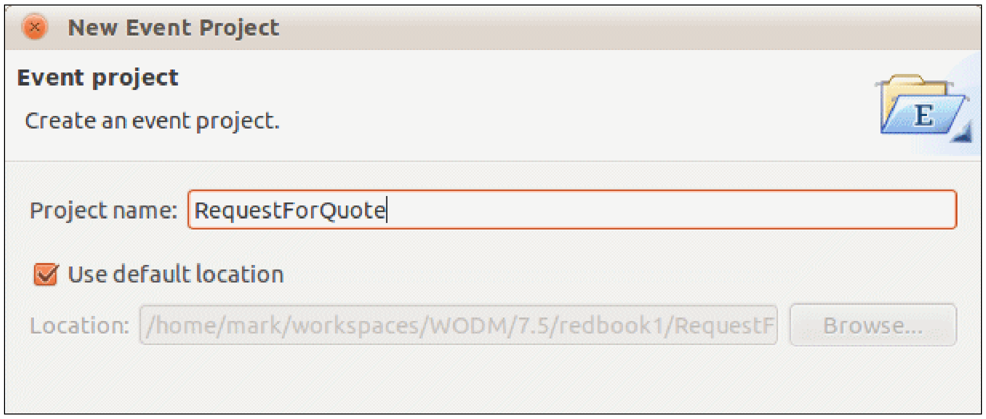
For more information about the artifacts that are created in an event project, review the *Event projects* topic in the Operational Decision Manager Information Center:

http://pic.dhe.ibm.com/infocenter/dmanager/v8r0m1/index.jsp?topic=%2Fcom.ibm.wodm.dserver.events.ref%2Ftopics%2Fref_dse_eventprojects.html

8.2.2 Creating the event project

To create the event project in the Event Designer, follow these steps:

1. Start the Event Designer in Windows by selecting **Start** → **All Programs** → **IBM Operational Decision Management V8.0.1** → **Event Designer**.
2. From the Event Designer, select **File** → **New** → **Event Project**.
3. On the New Event Project dialog, provide the name of the new event project as `RequestForQuote` (see Figure 8-3) and click **Finish**.


New Event Project

Event project

Create an event project.

Project name: RequestForQuote

☑ Use default location

Location: /home/mark/workspaces/WODM/7.5/redbook1/RequestF

Figure 8-3 Creating the event project

8.2.3 Creating the business objects and event from a COBOL copybook

In the new event project, you create the business objects that represent the vehicle and driver structures in the COBOL copybook. In Rule Designer, which is described in Chapter 3, “Getting started with business rules” on page 33, the copybook can be imported to automatically generate the required objects. There is no equivalent functionality in the Event Designer, and therefore, you must create the business objects manually or import an XML representation of the business event and create the business objects from this representation.

The `.xsd` file that is used to create the business event and business objects can be created as described in 8.4.3, “Creating the CICS Bundle project” on page 157. You can obtain the `Request.xsd` file in the additional material provided with this publication (see Appendix C, “Additional material” on page 335). This `.xsd` file is used to define the event and business objects.

Creating the event and business objects

The *event object* stores the information that is required to populate the business objects. The business event is received from CICS. The *event* contains an event object that stores the information that is required to populate the business objects.

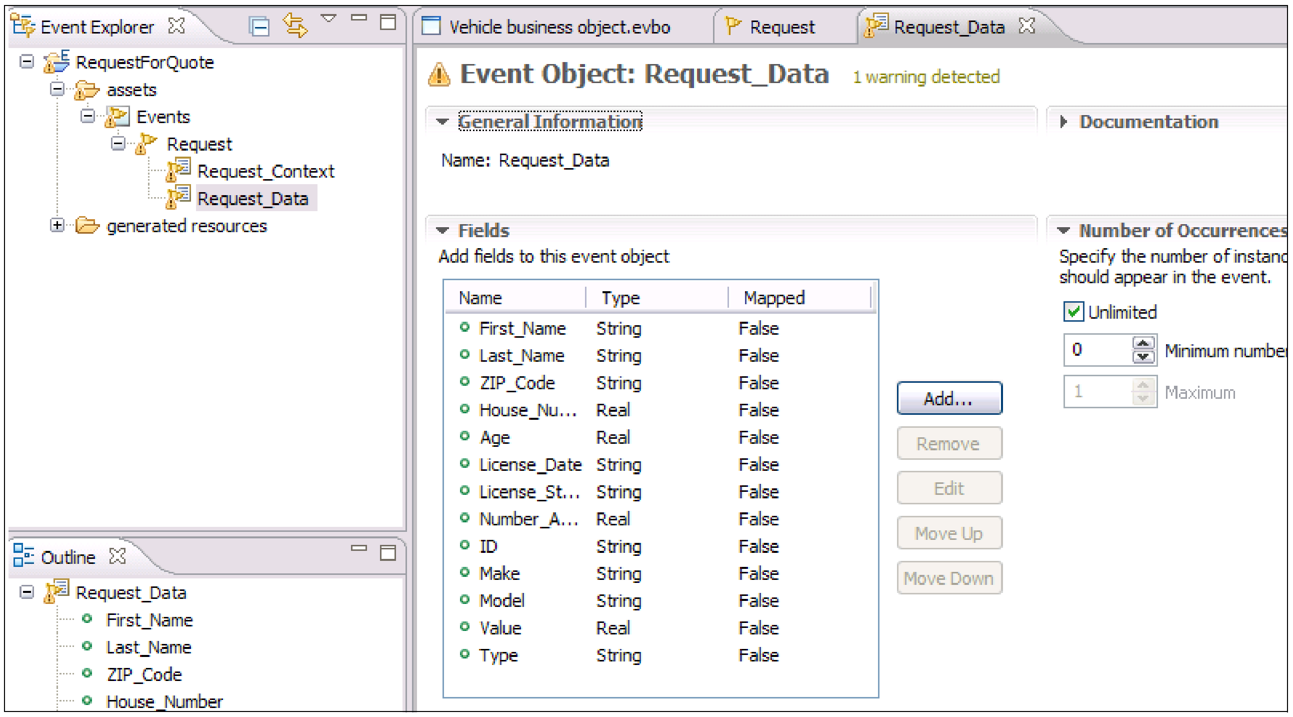
Request_data event object: When emitting events from CICS, you can have only one event object in which all the data fields are stored. This event object is called *Request_Data*. This limitation means that the fields for both the driver and the vehicle are placed in the Request_Data event object and then mapped to the correct fields of the necessary business object.

To define the event and business object, follow these steps:

1. Right-click the new **RequestForQuote** project, and select **New** → **Event**.
2. Select the option **Create an event based on an XML Schema**.
3. Navigate to the path where the XML file is located. Click **Next,** and then click **Finish**.

Note: The XML file, `Request.xsd`, that is used in this example is supplied in the additional resources for this book. For details, see Appendix C, “Additional material” on page 335.

4. An event called Request will be created as shown in Figure 8-4.



Name: Request_Data

Name	Type	Mapped
First_Name	String	False
Last_Name	String	False
ZIP_Code	String	False
House_Nu...	Real	False
Age	Real	False
License_Date	String	False
License_St...	String	False
Number_A...	Real	False
ID	String	False
Make	String	False
Model	String	False
Value	Real	False
Type	String	False

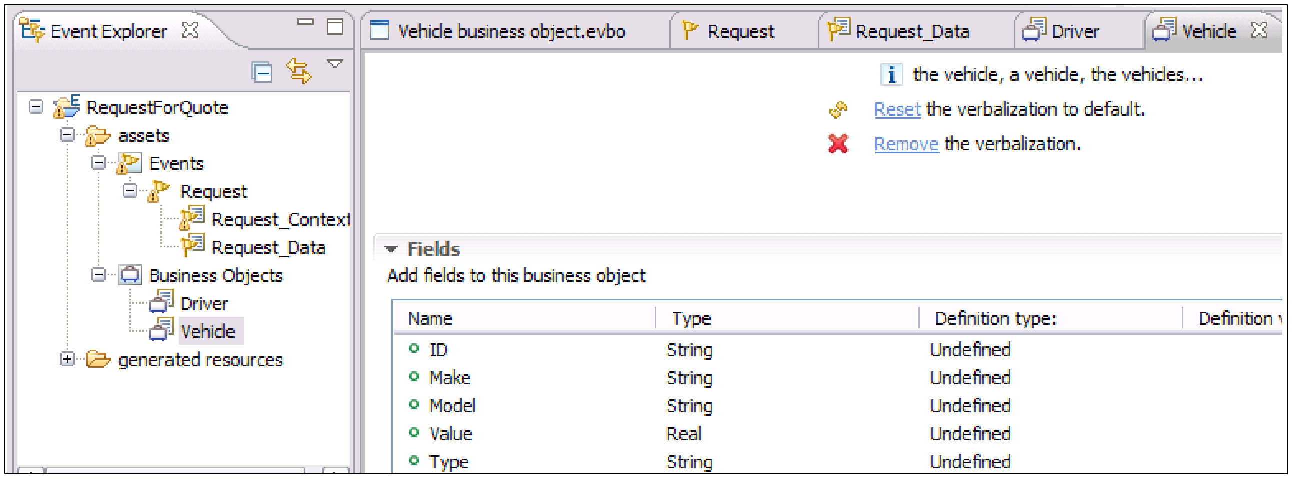
Unlimited: ✓
Minimum number: 0
Maximum: 1

Request_Data
- First_Name
- Last_Name
- ZIP_Code
- House_Number

Figure 8-4 Creating the event objects

5. Right-click the **Request_Data** → **Create Business Object from the Event Object**.
6. Enter the name `Driver` to create the `Driver` business object. Click **Next**.
7. Clear the `ID`, `Make`, `Model`, `Value`, and `Type`. Click **Finish**.
8. Repeat steps 5 to 7 to create a business object called `Vehicle`. This time, clear the fields `First_Name` through to `Number_Accidents`. Click **Finish**.

9. Two business objects, Driver and Vehicle, are created, as shown in Figure 8-5.



the vehicle, a vehicle, the vehicles...

Reset the verbalization to default.

Remove the verbalization.

Fields

Add fields to this business object

Name	Type	Definition type:	Definition
ID	String	Undefined	
Make	String	Undefined	
Model	String	Undefined	
Value	Real	Undefined	
Type	String	Undefined	

Figure 8-5 Creating the business objects

8.2.4 Creating the action

The action that fires when the event pattern is identified is now created from the business object.

Creating the action

To create the action, follow these steps:

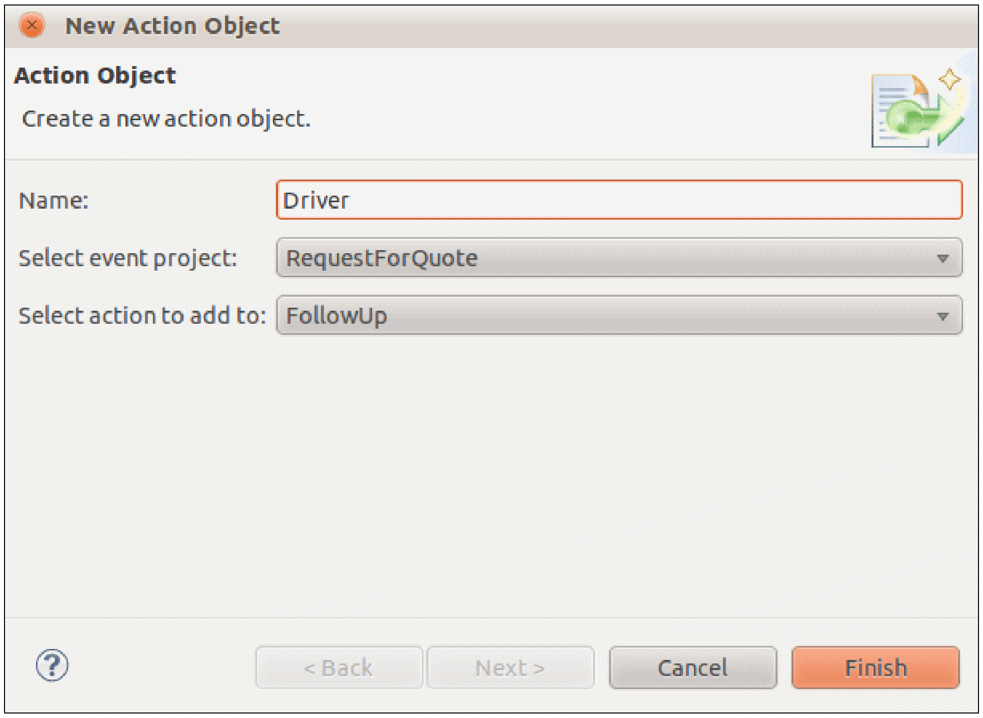
1. Right-click the **RequestForQuote** project, and select **New** → **Action**.
2. Select **Create a blank Action**, and click **Next**.
3. Enter the name `FollowUp` for the Action, and click **Next**.
4. Leave the connector set to `None`, and click **Finish**.

Creating the action objects

To create the action objects, follow these steps:

1. Right-click the **Driver** business object, and select **Create Action Object** from this business object.

2. Enter the name `Driver`, and select the **RequestForQuote** project and **FollowUp** action, as shown in Figure 8-6.



Name: Driver

Select event project: RequestForQuote

Select action to add to: FollowUp

Figure 8-6 Creating the Action Object from the business object

3. Click **Finish**, and the action object is generated. Using this method ensures that the action object has all of the required field constructors to map the data from the business object to the action. This technique also avoids the manual mapping steps that are required for the event object fields.
4. Repeat this step to create the Vehicle Action Object, specifying `Vehicle` as the name for the new action object.

8.2.5 Creating the event rule

All the necessary artifacts for the event application are created. Event logic can now be written. It is triggered by events. Then, it acts on the business objects and emits actions.

Creating the event rule

The event logic is stored in an *event rule*. To create the event rule, follow these steps:

1. To create the event rule, right-click the **RequestForQuote** project, and select **New** → **Event Rule**.
2. Enter `IdentifyFollowUp` as the name of the event rule, and click **Next**.
3. Select the **Request** event as the event to trigger this event rule, and click **Next**.
4. Select the **ID of the vehicle** as the context relationship, and click **Finish**.

Using a context relationship for the event rule

In this scenario, Decision Server Events must identify whether the same client requested more than three quotes in the same hour. To enable this function, a context relationship must be defined so that Decision Server Events has a field that it can use to identify events from the same client.

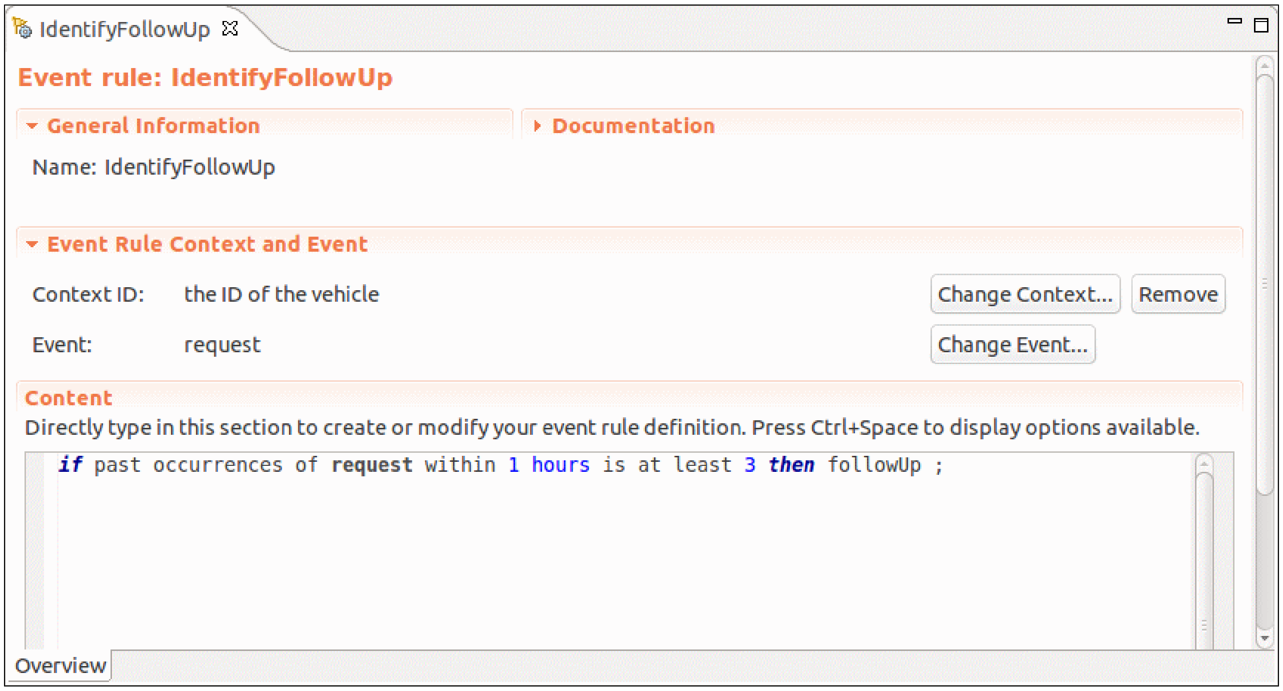
For this scenario, the ID of the vehicle was selected as the unique identifier to allow you to identify the pattern of more than three quotes within the same hour.

Writing the event rule

The event rule is entered directly in to the Event Rule editor, as shown in Figure 8-7.

The event uses the following logic:

```
if past occurrences of request within 1 hour is at least 3 then followUp;
```



Name: IdentifyFollowUp

Context ID: the ID of the vehicle

Event: request

Directly type in this section to create or modify your event rule definition. Press Ctrl+Space to display options available.

```
if past occurrences of request within 1 hours is at least 3 then followUp ;
```

Figure 8-7 Entering the event logic in the Event Rule editor

8.2.6 Configuring the technology connectors

The final authoring stage for the event application is to configure the event and action technology connectors (connectors). The *connectors* define which protocol the Decision Server Events run time uses to receive the event and send the action.

Decision Server Events supports the following multiple event and action connectors, as described in the *Defining technology connectors* topic of the Decision Server V8.0.1 Information Center:

http://pic.dhe.ibm.com/infocenter/dmanager/v8r0m1/index.jsp?topic=%2Fcom.ibm.wodm.dserver.events.dev%2Ftopics%2Ftsk_dse_definingtechnologyconnectors.html

- Available event connectors:
  - Email: Receive an event by POP3
  - File: Receive an event when a file is added or modified in a folder

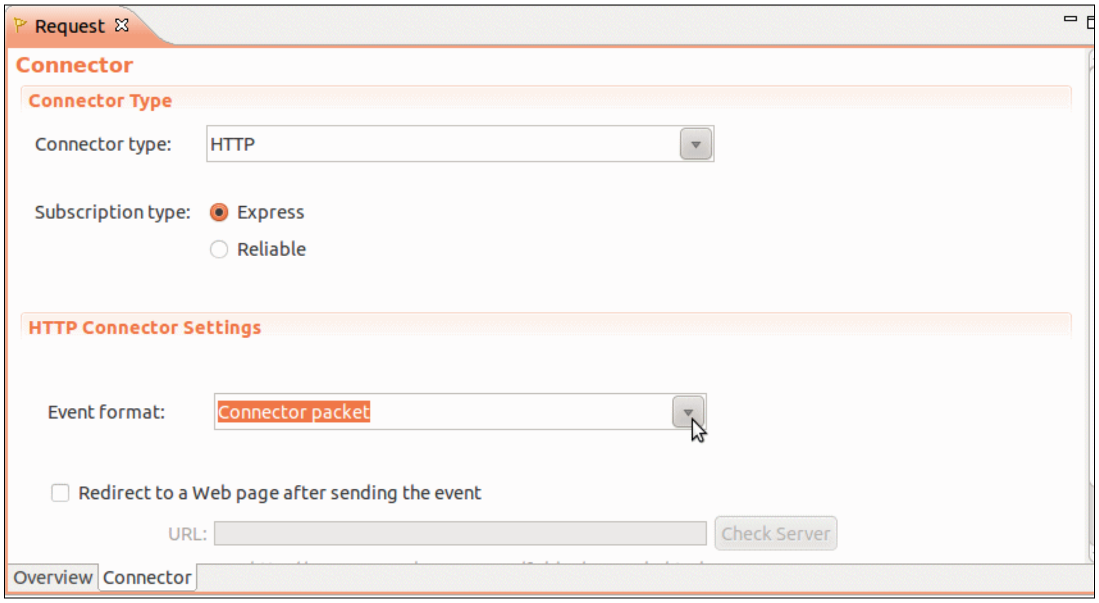
– FTP: Receive an event when a file is added or modified on an FTP site
– HTTP: Receive an event by HTTP
– Java Database Connectivity (JDBC): Receive an event by executing an SQL statement
– Java Message Service (JMS): Receive an event directly from a JMS destination
– SOAP: Receive events by SOAP

- Available action connectors:
  – Email: Send an action by Simple Mail Transfer Protocol (SMTP)
  – File system: Save the action as a file in a folder
  – FTP: Save the action as a file on an FTP site
  – HTTP: Send the action by HTTP
  – JDBC: Send an action by executing an SQL statement
  – JMS: Send an action directly to a JMS destination
  – Representational State Transfer (REST): Send an action to an application by using the REST interface
  – SOAP: Invoke a web service by using SOAP 1.1
  – User console: Send an action to the user console for review and response

Configuring the event connector

This example uses the HTTP connector to receive the Request Event from CICS. Follow these steps:

1. Double-click the **Request** event to open the Event editor.
2. Select the connector tab to navigate to the Connector panel, as shown in Figure 8-8.
   a. From the Connector Type drop-down menu, select **HTTP** as the connector type for the Request event.
   b. From the Event format drop-down menu, select the **Connector packet** option.
   c. Save the changes, and close the Event editor.



Connector type: HTTP

Subscription type: [✓ Express] Reliable

Event format: Connector packet

☐ Redirect to a Web page after sending the event

URL:

Figure 8-8 The Connector panel of the Event editor

Configuring the action connector

For this scenario, you do not configure a connector for the FollowUp action. Instead, the Events tooling is used to verify that the action fired when the business rule is met.

In an actual scenario, an appropriate connector is used to send the generated action to the required system. For example, the FollowUp action can be delivered through email to the sales team to notify them to call the client to follow up on the client's requests for a quotation.

8.3 Deploying the event application to the event run time

The Decision Server events run time on z/OS is a WebSphere application that is hosted inside the WebSphere Application Server for z/OS.

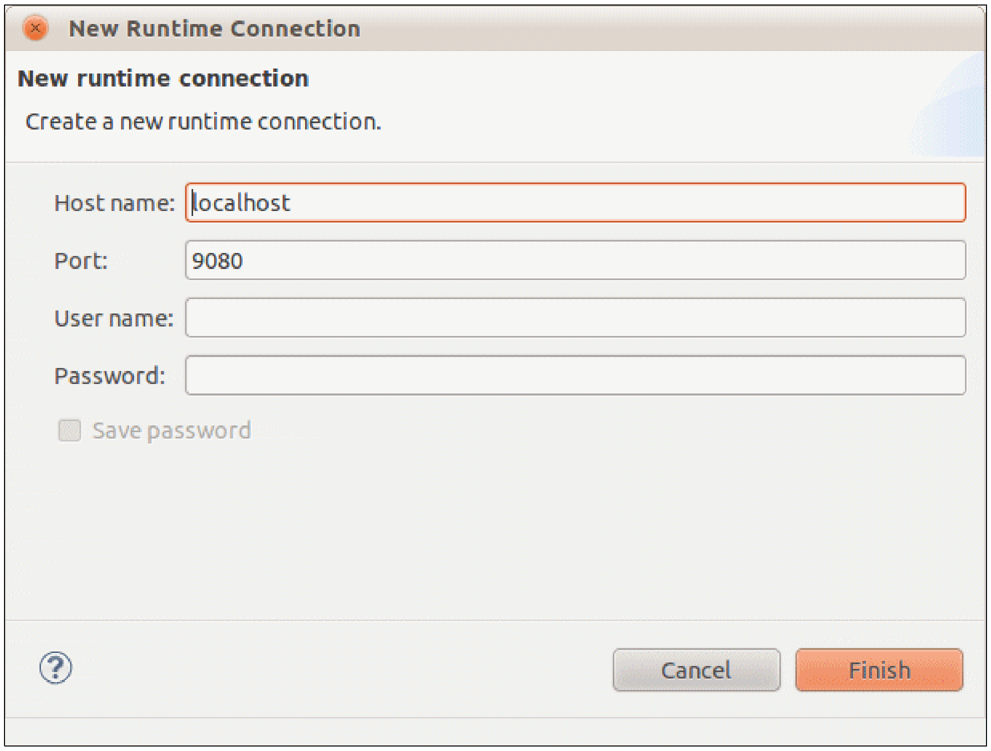
8.3.1 Creating the event runtime connection

To deploy the event application to Decision Server Events, the Event Designer creates an event runtime connection. Using this connection, you can both deploy and view assets on the server.

To create the event runtime connection, follow these steps:

1. Select the **Event Runtimes** tab, and right-click in the empty white space.
2. Click **New runtime connection**, and enter the server information, as shown in Figure 8-9:
   - For Host name, enter the name of your z/OS logical partition (LPAR).
   - For Port, enter the appropriate port number for the Events runtime server.

   Click **Finish** to create the connection and connect the Event Designer to the Decision Server Events run time.



New Runtime Connection

New runtime connection

Create a new runtime connection.

Host name: localhost

Port: 9080

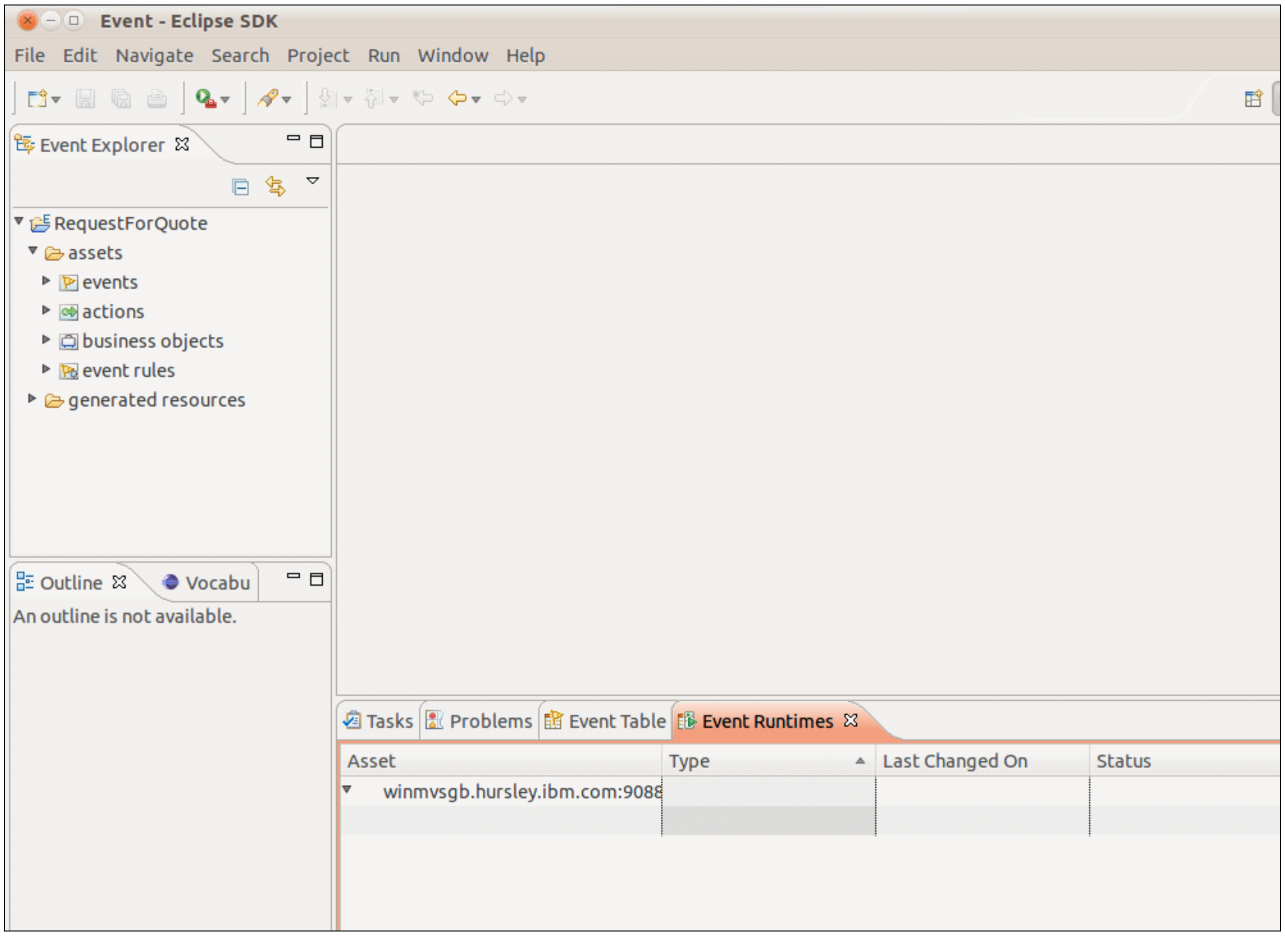
User name:

Password:

Save password

Figure 8-9 Entering the information for a new runtime connection

3. Figure 8-10 shows the Event Designer connected to the Decision Server Events run time, which has no assets deployed.



An outline is not available.

Asset	Type	Last Changed On	Status
winmvsgb.hursley.ibm.com:9088			

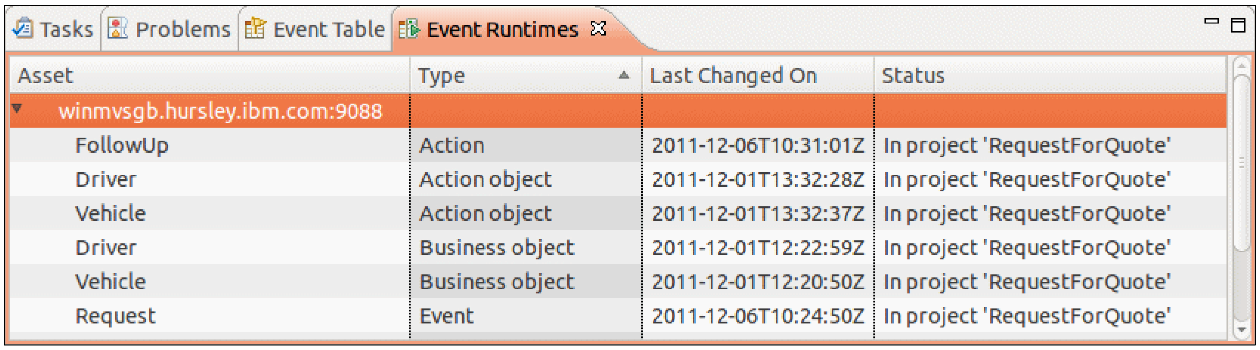
Figure 8-10 The new Event Runtimes tab with an empty repository

8.3.2 Deploying the event project to the event run time

To deploy the event project to the event run time, follow these steps:

1. Right-click the **RequestForQuote** event project, and select **Deploy**.
2. Select **Deploy all assets**, and click **Next**.
3. Select **Use a known runtime**, and highlight the recently created runtime definition.
4. Click **Finish** to deploy the assets.

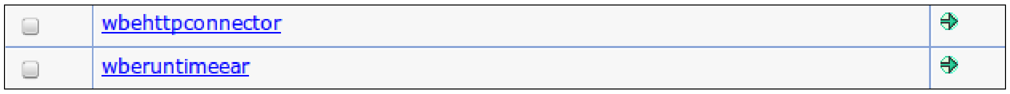
After the deployment completes, the assets are shown in the Event Runtimes tab, as shown in Figure 8-11.



Asset	Type	Last Changed On	Status
winmvsgb.hursley.ibm.com:9088			
FollowUp	Action	2011-12-06T10:31:01Z	In project 'RequestForQuote'
Driver	Action object	2011-12-01T13:32:28Z	In project 'RequestForQuote'
Vehicle	Action object	2011-12-01T13:32:37Z	In project 'RequestForQuote'
Driver	Business object	2011-12-01T12:22:59Z	In project 'RequestForQuote'
Vehicle	Business object	2011-12-01T12:20:50Z	In project 'RequestForQuote'
Request	Event	2011-12-06T10:24:50Z	In project 'RequestForQuote'

Figure 8-11 The deployed assets as shown in the Event Runtimes tab

Because this event project uses the HTTP connector, you now see that a new application is automatically deployed to your WebSphere Application Server, as shown in Figure 8-12.



☐	wbehttpconnector	➔
☐	wberuntimeear	➔

Figure 8-12 The wbehttpconnector web application on WebSphere Application Server

The HTTP event connector is used to receive an event by using an HTTP POST and to send an action by using HTTP.

Configuring the HTTP connector for security

When Decision Server Events is installed on a WebSphere Application Server with security enabled, you must specify a user role mapping for the wbehttpconnector application. The *user role mapping* determines which users have permission to send events through HTTP to the connector.

To specify the user role mapping by using the WebSphere Application Server administrative console, complete the following steps:

1. Log in to the WebSphere Application Server administrative console.
2. In the menu, expand **Applications** → **Application Types**, and click **WebSphere enterprise applications**.
3. In the main panel, click the HTTP connector application, **wbehttpconnector**.
4. Under Detailed Properties, click **Security role to user/group mapping**. Select the **HTTPEventConnectorUser** role, and map one or more users to the role. To allow unrestricted access when security is enabled, map the role to the **Everyone** special subject.
5. Click **Save**.

Note: These steps are required only when an event project that uses the HTTP connector is first deployed. When subsequent projects are deployed, the existing mappings are maintained.

8.4 Emitting events from CICS

The event project is now deployed to the event run time, and Decision Server Events is waiting to receive events through HTTP. For this example, CICS is configured to send the Request event each time that a client requests a quotation.

8.4.1 CICS event support

Given the considerable amount of business processing that is performed in CICS systems across the world (over 30 billion transactions a day), CICS is a significant source of business events. Events can provide enhanced business flexibility and help to meet governance and compliance regulations.

Using event specifications that are defined to CICS, events can be captured from existing business application programs without altering the original code. These capture points include relevant CICS application programming interface (API) calls and when a program starts. Each time that a program executes a capture point, CICS checks each enabled capture specification that matches the capture point.

Each matching capture specification contains optional filters to compare against the application context, several command options on the API call, and data passed on the API call. If the filters match, CICS collects the payload information from information sources described in the capture specification, enriches it with context data, and then queues the event for dispatch so that the application can continue to execute quickly.

8.4.2 CICS Event Binding Editor

The CICS Event Binding Editor in IBM CICS Explorer® is used to create an event binding that contains an event specification for our program. The following instructions contain the information that is required to create the CICS Bundle project for this scenario. For information about CICS event processing, see the IBM Redbooks publication *Implementing Event Processing with CICS*, SG24-7792.

You can download CICS Explorer onto your workstation from the CICS Explorer website:

`http://www-01.ibm.com/software/htp/cics/explorer/`

8.4.3 Creating the CICS Bundle project

To create the CICS Bundle project, follow these steps:

1. Start CICS Explorer from the location to which it was downloaded.
2. Switch to the Resource perspective in CICS Explorer.
3. Create a CICS Bundle project by selecting **Explorer** → **New Wizards** → **CICS Bundle project**.
4. Enter `RequestForQuote` as the name, and click **Finish**.

8.4.4 Creating the event binding

This section describes the steps for creating a business event; however, files from the supporting material can be used to populate CICS Explorer.

From CICS Explorer, click **Import** → **File**, select the **RequestForQuote** directory, and click **Finish**. The result supplies the CICS bundle if this route has been chosen. Continue with 8.4.8, “Deploying the bundle to CICS” on page 164.

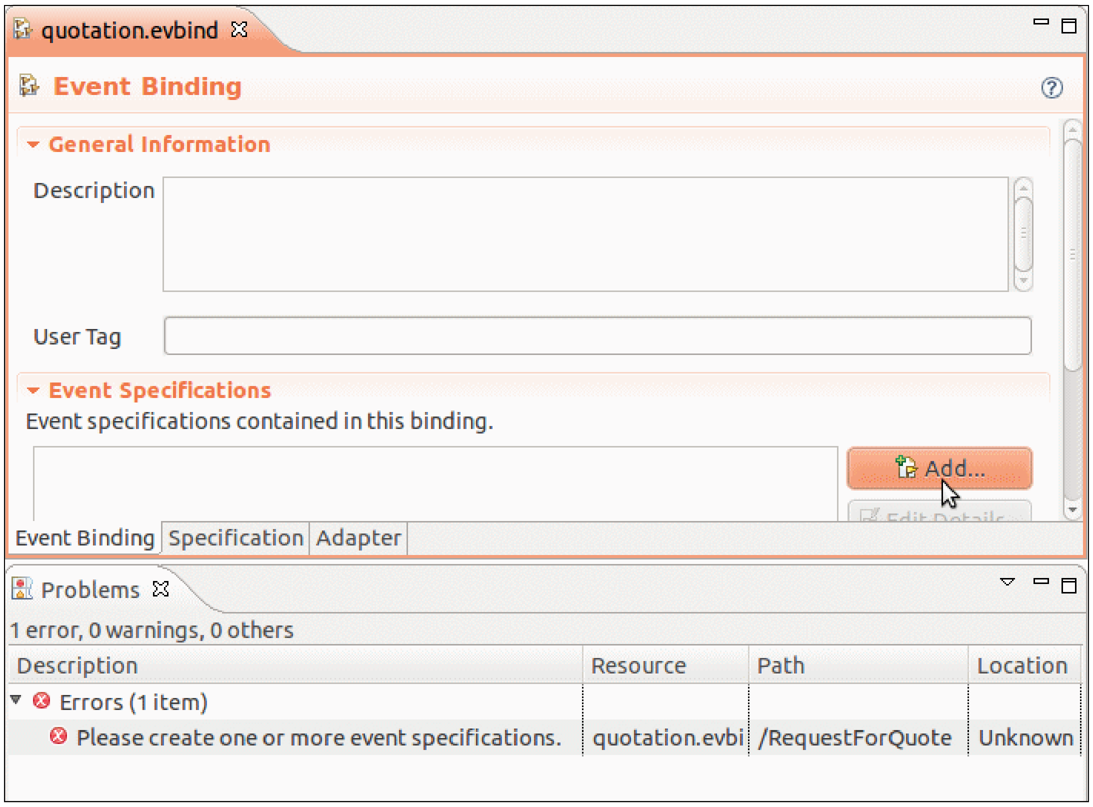
Alternatively, continue with the following steps in this section.

An *event binding* defines a business event to CICS. An event binding can be created by using the CICS Event Binding Editor by business analysts and developers or by an application analyst in response to a business requirement. An event specification describes an event and its processing in natural language.

To create the Event Binding, follow these steps:

1. Right-click the **RequestForQuote** project, and select **New** → **CICS Event Binding**.
2. Enter `quotation` as the name, and click **Finish**.

A new event binding is created and an error message is also displayed, as shown in Figure 8-13. A new event specification must now be created.



1 error, 0 warnings, 0 others

Description	Resource	Path	Location
Errors (1 item)			
Please create one or more event specifications.	quotation.evbi	/RequestForQuote	Unknown

Figure 8-13 The new event binding with an error message displayed

8.4.5 Creating the event specification

The *event specification* describes to CICS the information that is contained in the event that is emitted to the Decision Server Events run time.

To create the event specification, follow these steps:

1. Click **Add**, as shown in Figure 8-13.
2. Enter `Request` as the name for the specification, and click **Finish**.

3. The event specification is now created and must be edited. Click **Edit Details**, as shown in Figure 8-14.

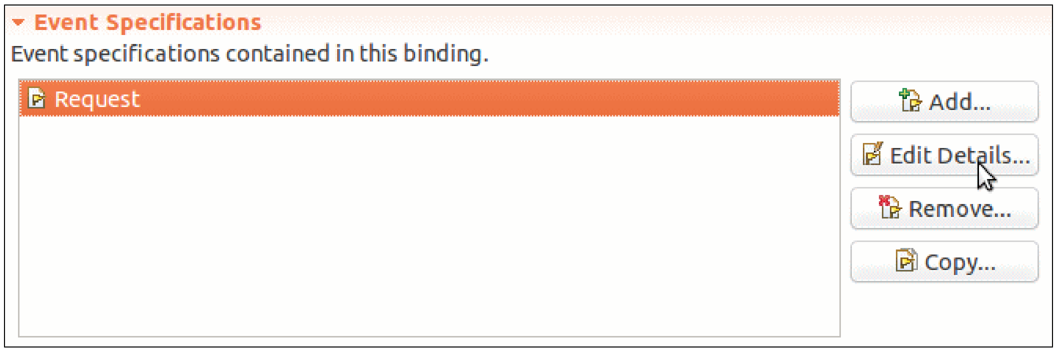


Figure 8-14 The newly created event specification

4. You must define the fields that CICS emits in this event. To add the first field, click **Add** in the Emitted Business Information table, as shown in Figure 8-15.

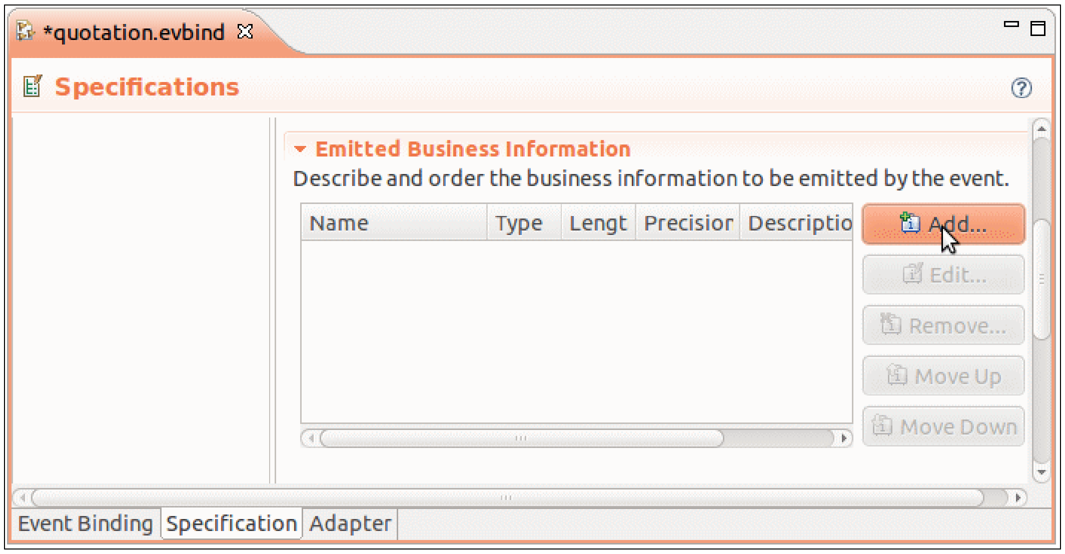


Figure 8-15 The emitted business information is created

5. Enter the first field as `First_Name` with the data type as `Text` and click **OK**.
6. Repeat step 5 with the field names and data types that are listed in Table 8-1.

Table 8-1 The fields defined in the event specification

Field name	**Field data type**
First_Name	Text
Last_Name	Text
ZIP_Code	Text
House_Number	Numeric
Age	Numeric
License_Date	Text

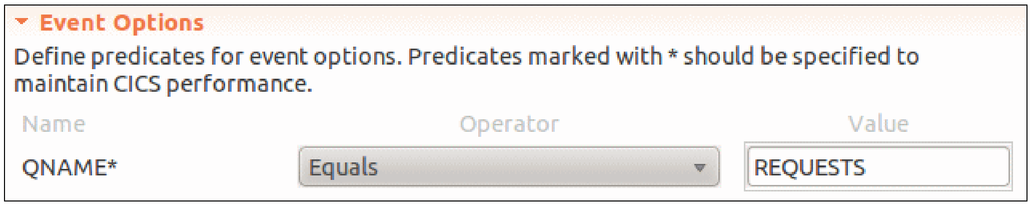
Field name	Field data type
License_Status	Text
Number_Accidents	Numeric
ID	Text
Make	Text
Model	Text
Value	Numeric
Type	Text

8.4.6 Creating the capture specification

Now that the event specification is complete, you can tell CICS how to capture the event and how to emit it to the Decision Server Events run time.

To create the capture specification, follow these steps:

1. Click **Add a Capture Specification** on the Event specification tab.
2. Enter `Request` as the name, and click **Finish**.
3. Click the newly created **Request** capture specification.
4. Scroll down and select **WRITEQ TS** as the Application Capture Point.
5. Next, click the **Filtering** tab at the top. In the Event Options section, specify that the QNAME **Equals** `REQUESTS`, as shown in Figure 8-16.

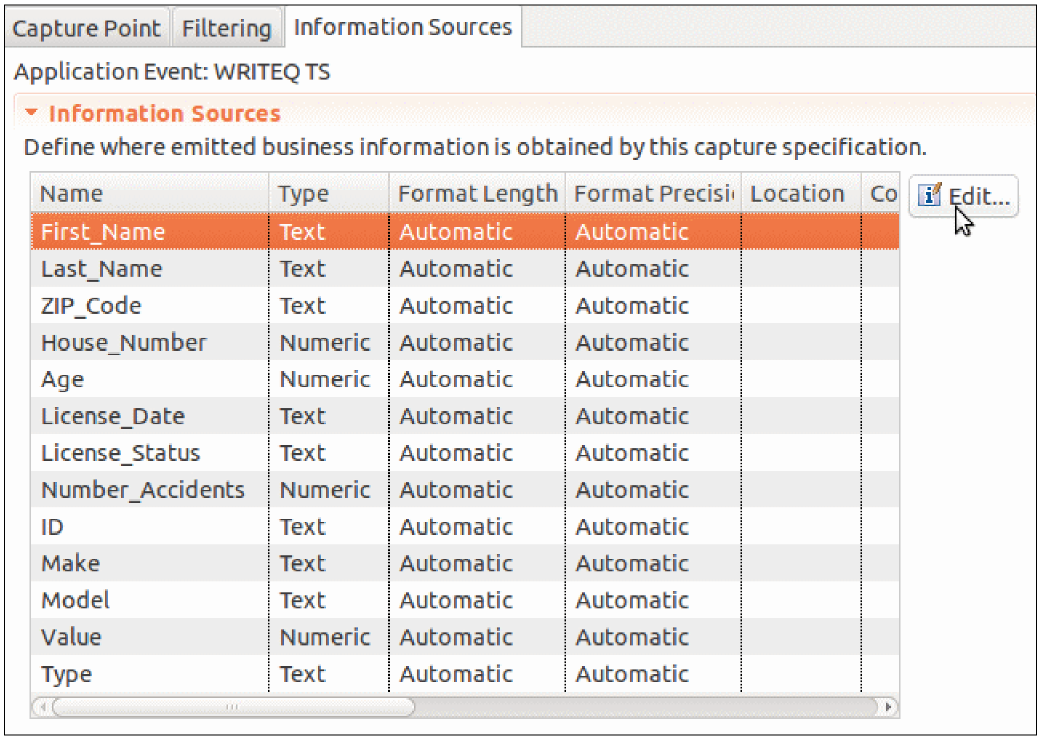


Define predicates for event options. Predicates marked with * should be specified to maintain CICS performance.

Name	Operator	Value
QNAME*	Equals	REQUESTS

Figure 8-16 Filtering which TS Queue causes the event to be emitted

6. Click the **Information Sources** tab to display the event data fields. The COBOL copybook is used to describe how the data that is written to the TS queue is mapped in to the event fields.
7. Select the first row in the table, the **First_Name** field, and click **Edit**, as shown in Figure 8-17.



Application Event: WRITEQ TS

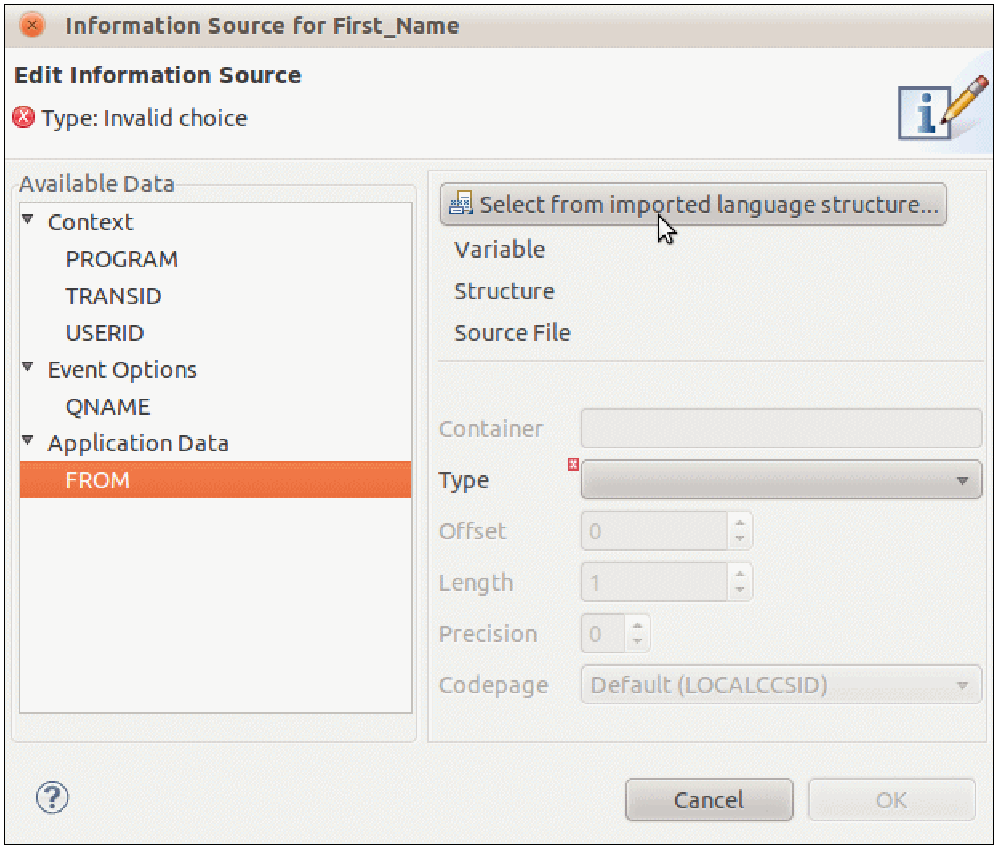
Information Sources

Define where emitted business information is obtained by this capture specification.

Name	Type	Format Length	Format Precisi	Location	Co
First_Name	Text	Automatic	Automatic		
Last_Name	Text	Automatic	Automatic		
ZIP_Code	Text	Automatic	Automatic		
House_Number	Numeric	Automatic	Automatic		
Age	Numeric	Automatic	Automatic		
License_Date	Text	Automatic	Automatic		
License_Status	Text	Automatic	Automatic		
Number_Accidents	Numeric	Automatic	Automatic		
ID	Text	Automatic	Automatic		
Make	Text	Automatic	Automatic		
Model	Text	Automatic	Automatic		
Value	Numeric	Automatic	Automatic		
Type	Text	Automatic	Automatic		

Figure 8-17 Editing from where the event fields receive their data

8. In the new window, select **FROM** in the Available Data view. Click **Select from imported language structure**, as shown in Figure 8-18.



Information Source for First_Name

Edit Information Source

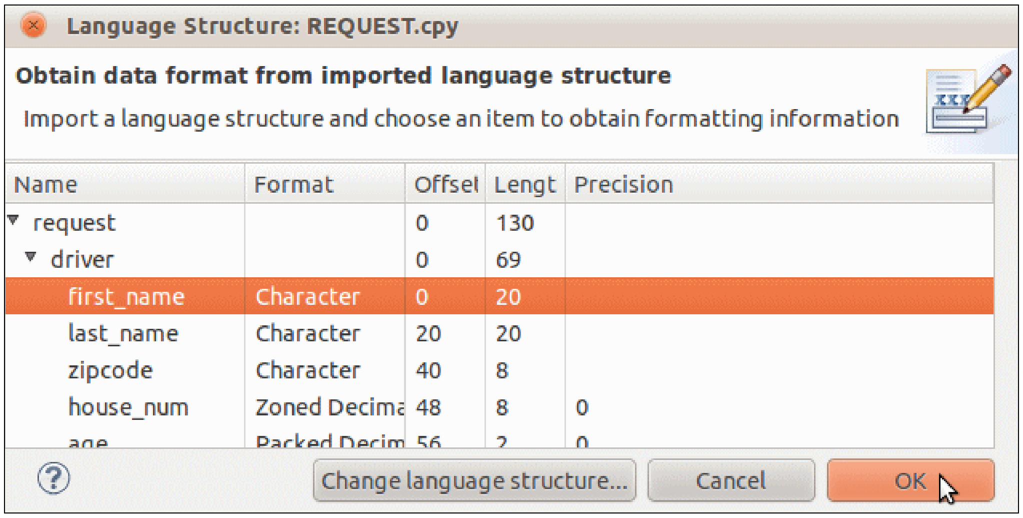
Type: Invalid choice

Figure 8-18 Importing the COBOL copybook

9. Click **Choose Language Structure File**, navigate to the COBOL copybook that is used for this scenario, and click **OK**.

10. Ensure that the Source Language is specified as COBOL, and click **OK**.

11. In the new Language Structure window, expand the **driver** section and select the **first_name** field, as shown in Figure 8-19.


Language Structure: REQUEST.cpy

Obtain data format from imported language structure

Import a language structure and choose an item to obtain formatting information

Name	Format	Offset	Lengt	Precision
request		0	130	
driver		0	69	
first_name	Character	0	20	
last_name	Character	20	20	
zipcode	Character	40	8	
house_num	Zoned Decima	48	8	0
age	Packed Decim	56	2	0

Figure 8-19 The COBOL copybook is used to define the event data mapping

12. Click **OK** to finish the mapping of the COBOL data to the event field.
13. Repeat these steps for the remaining fields on the Information Sources tab. Ensure that the correct field is selected from the COBOL copybook for the event field that you are editing.

8.4.7 Defining the adapter

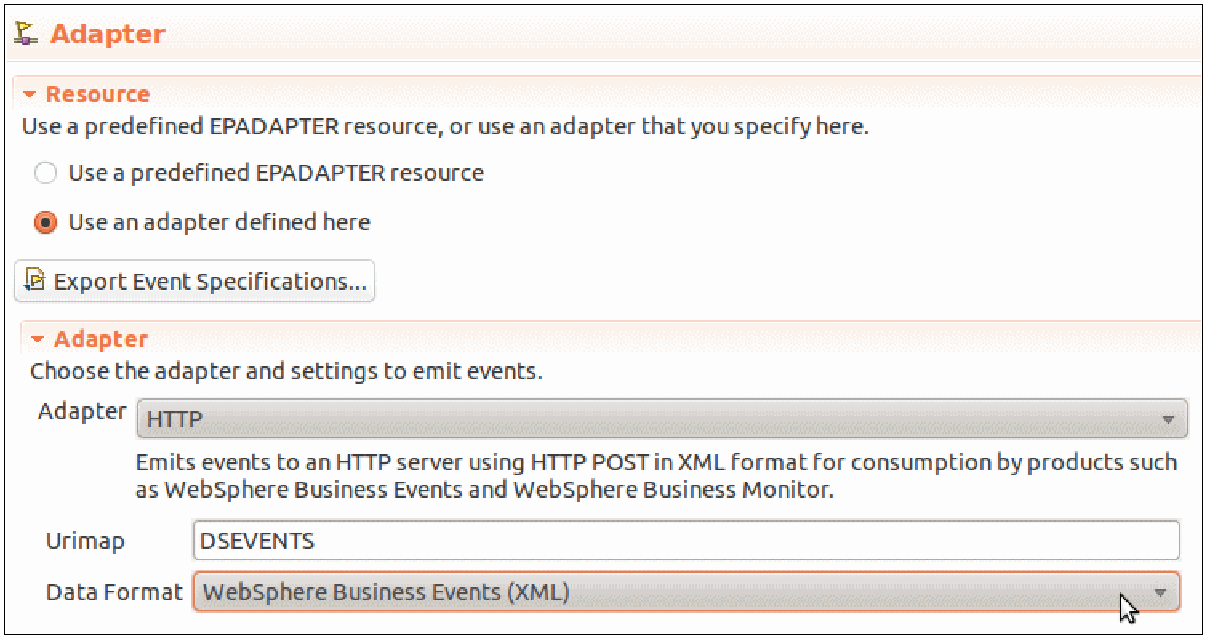
The adapter defines the protocol that CICS uses to send the event to the Decision Server Events run time. In this scenario, the HTTP adapter is used to send the event.

To configure the adapter, follow these steps:

1. Select the **Adapter** tab of the Event binding editor.
2. Enter the following information, as shown in Figure 8-20 on page 164:
   a. Select **Use an adapter defined here**.
   b. Select **HTTP** from the list of adapters.
   c. Enter the name `DSEVENTS` as the Urimap. This name is defined to CICS in a later step.
   d. For the Data Format, select **WebSphere Business Events (XML)** from the drop-down list.

   Note: The adapter is defined in the event bundle here for a quick deployment. In an actual scenario, it is suggested that the you define the adapter as an EPADAPTER in CICS. This approach separates the creation and management of the adapter from the event binding file.

3. Save the event binding, and close the CICS Event Binding Editor.


Adapter

Resource

Use a predefined EPADAPTER resource, or use an adapter that you specify here.

Use a predefined EPADAPTER resource [✓ Use an adapter defined here]

Export Event Specifications...

Adapter

Choose the adapter and settings to emit events.

Adapter: HTTP

Emits events to an HTTP server using HTTP POST in XML format for consumption by products such as WebSphere Business Events and WebSphere Business Monitor.

Urimap: DSEVENTS

Data Format: WebSphere Business Events (XML)

Figure 8-20 The completed adapter specification that CICS uses to transmit the events

8.4.8 Deploying the bundle to CICS

The RequestForQuote CICS bundle now contains the completed Request event binding. The *event binding* describes the fields that are emitted in the Request event, how the event is triggered, and how data from the COBOL application is mapped into the event fields.

The next step is to configure CICS to emit the Request event to the Decision Server Events run time each time that a client requests a quotation.

Enabling CICS for events

First, CICS must be enabled for event processing. Event processing is enabled, by default, when a `START=INITIAL` or `START=COLD` parameter is used during the startup of your CICS TS V4.1 or V4.2 system. When a `START=WARM` or `START=EMERGENCY` parameter is used, the settings from the previous run of CICS are used. So, unless the setting changed, event processing is enabled.

You can inquire on EVENTPROCESS to check whether event processing is enabled, for example, by using CICS Explorer. You can stop and start event processing from the IBM CICS Explorer, the CICSPlex® SM Web User Interface, or the CICS SPI or API command. You might want to stop event processing for an upgrade or system maintenance, and then start event processing again.

Deploying the bundle

The CICS bundle must be transferred to a directory in the z/OS Distributed File Service (zFS) for CICS to use it. CICS Explorer provides the required functionality to transfer the CICS bundle to the zFS.

To transfer the bundle, follow these steps:

1. Ensure that you configure an FTP connection under **Window** → **Preferences** → **CICS Explorer** → **Connections**.

2. Next, right-click the **RequestForQuote** bundle, and select **Export Bundle Project to z/OS UNIX File System**.
3. Navigate to a directory to which you have write access, typically in your home directory, for example, `/u/hiscm/RequestForQuote`, and click **Finish**.

CICS Explorer deploys the bundle to the specified directory.

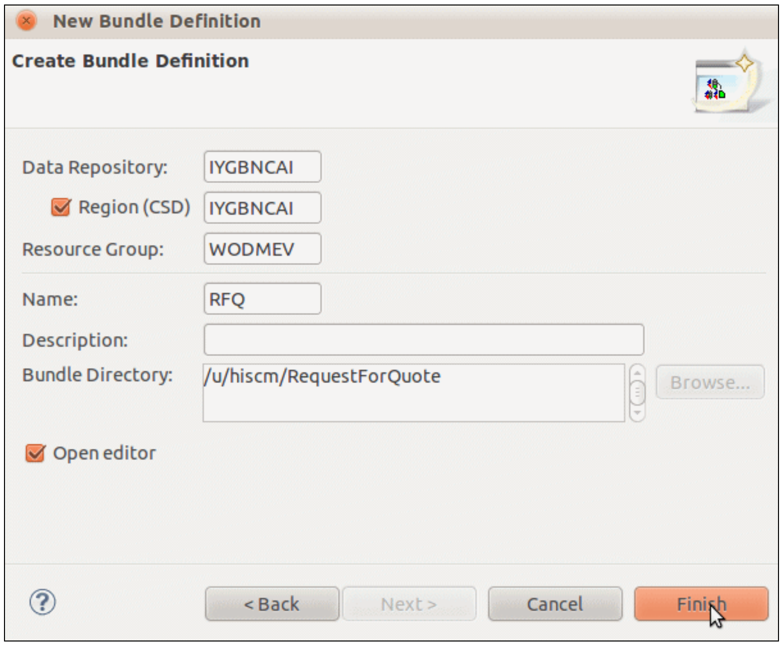
Creating a bundle definition

After the bundle is deployed to the zFS, its location must be defined to CICS by using a bundle definition (BUNDEF).

To create the bundle definition, follow these steps:

1. Select **Explorer** → **New Wizards** → **Other**.
2. Expand the **CICS Definitions** folder, select **Bundle Definition**, and click **Next**.
3. On the New Bundle Definition dialog, enter the following information as shown in Figure 8-21:
   a. Enter the group as `WODMEV`.
   b. Enter the name of the bundle definition as `RFQ`.
   c. Enter the Bundle Directory as the same location that you specified previously, for example, `/u/hiscm/RequestForQuote`.

   Click **Finish** to complete the creation of the bundle definition.



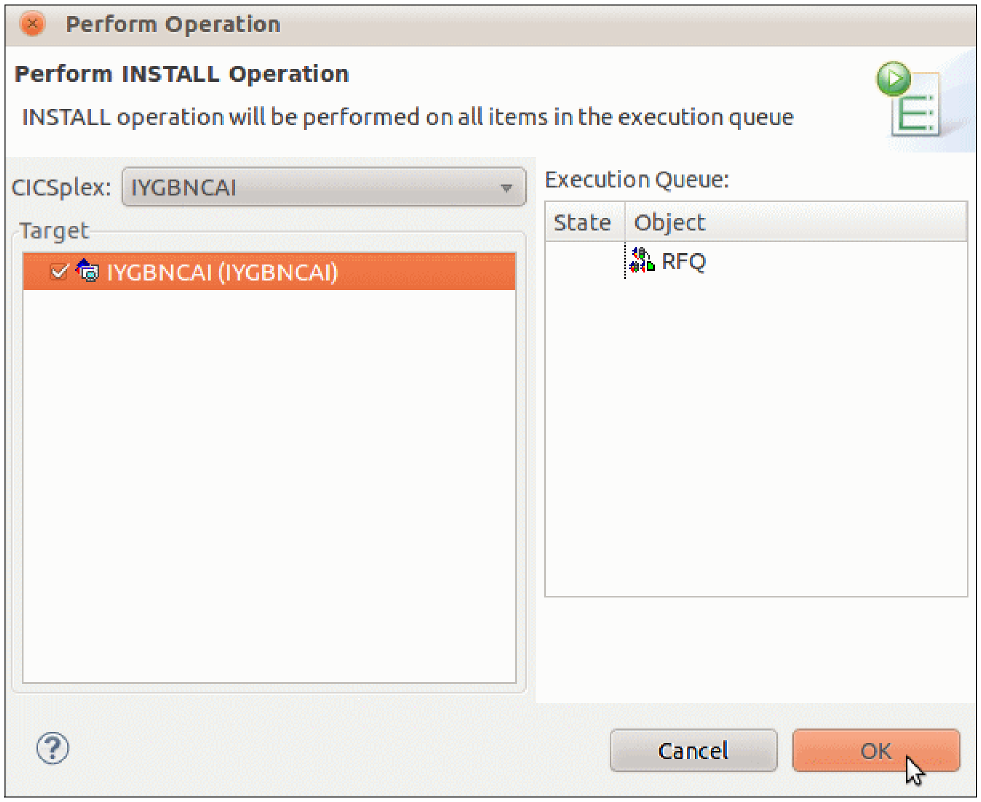
Data Repository: IYGBNCAI
☑ Region (CSD) IYGBNCAI
Resource Group: WODMEV
Name: RFQ
Description:
Bundle Directory: /u/hiscm/RequestForQuote
☑ Open editor

Figure 8-21 The completed bundle definition

Now that the bundle definition is created, it must be installed into CICS. Follow these steps to install the new bundle definitions:

1. Open the Bundle Definitions view by selecting **Window** → **Show View** and select **Bundle Definitions**.
2. Right-click the newly created **RFQ** bundle.
3. Select **install**.

4. Then, select **IYGBNCAI**, which is the CICS into which to install the bundle, and click **OK**, as shown in Figure 8-22.


Perform Operation

Perform INSTALL Operation

INSTALL operation will be performed on all items in the execution queue

CICSplex: IYGBNCAI

Target

☑ IYGBNCAI (IYGBNCAI)

Execution Queue:

State	Object
	RFQ

Figure 8-22 Installing the RFQ resource bundle

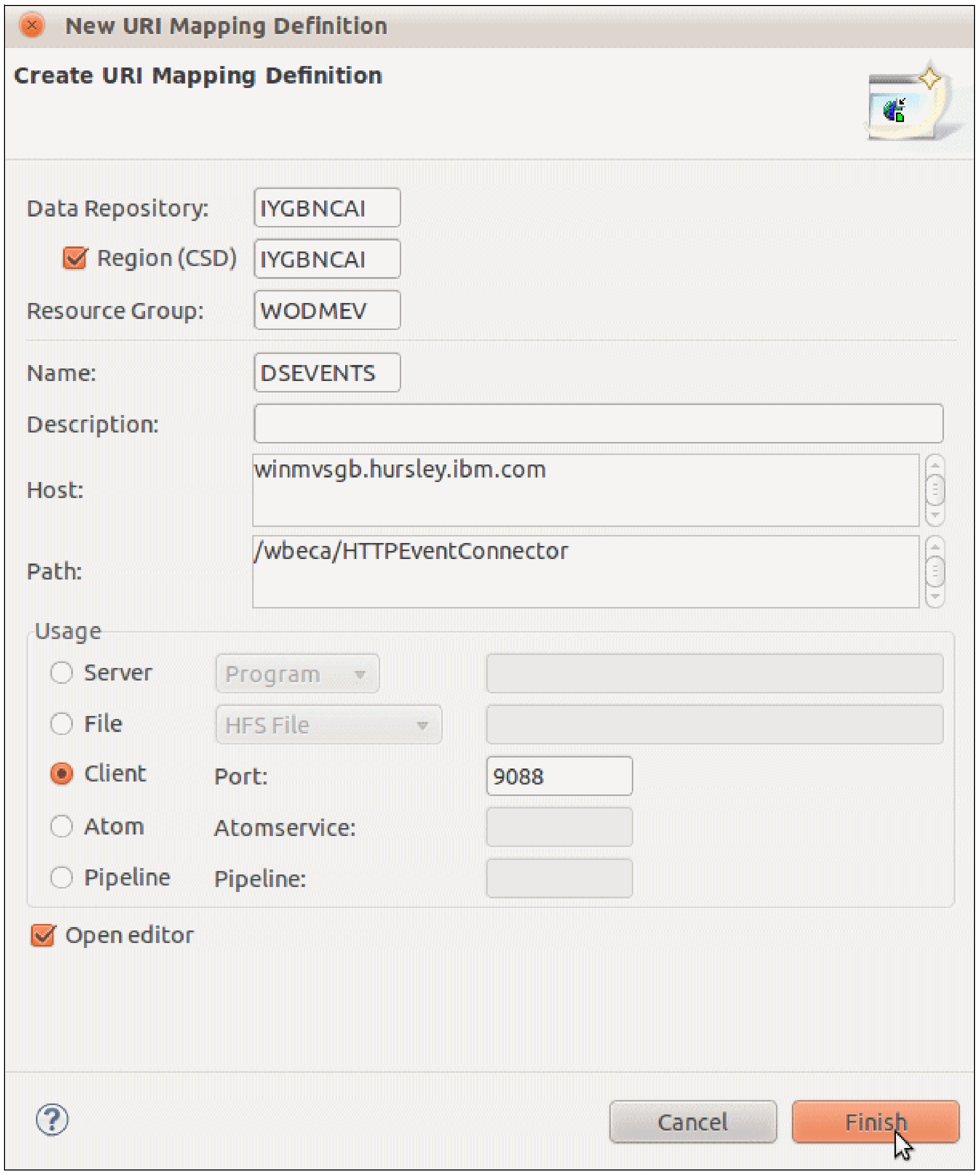
User ID access: Ensure that the user ID under which CICS is running has the appropriate access to the UNIX System Services directory to which the event binding is transferred.

Creating the URI mapping definition

The URI mapping definition DSEVENTS was used on the event adapter to tell CICS how to send the event to the Decision Server Events run time. Create this URI mapping definition:

1. Open the URI Mapping Definition view by selecting **Window** → **Show View** and select **URI Mapping Definition**.
2. Right-click in the new view and select **New**.
3. On the New URI Mapping Definition dialog, enter the following information, as shown in Figure 8-23 on page 167:
   a. Enter the resource group as `WODMEV`.
   b. Enter the name as `DSEVENTS`.
   c. Enter the host where the Decision Server Events run time is located.
   d. Enter the path as `/wbeca/HTTPEventConnector`.
   e. Click **Client** and enter the port on which the Decision Server Events run time is listening.

   Click **Finish**.



Data Repository: IYGBNCAI
[✓] Region (CSD) IYGBNCAI
Resource Group: WODMEV
Name: DSEVENTS
Description:
Host: winmvsgb.hursley.ibm.com
Path: /wbeca/HTTPEventConnector
Usage
Server Program
File HFS File
[✓ Client] Port: 9088
Atom Atomservice:
Pipeline Pipeline:
[✓] Open editor

Figure 8-23 Creating the URI mapping definition

4. Right-click the newly created URI mapping definition, and select **Install**.
5. Select the CICS to which to install the definition, and click **OK**.

Creating the TS Queue

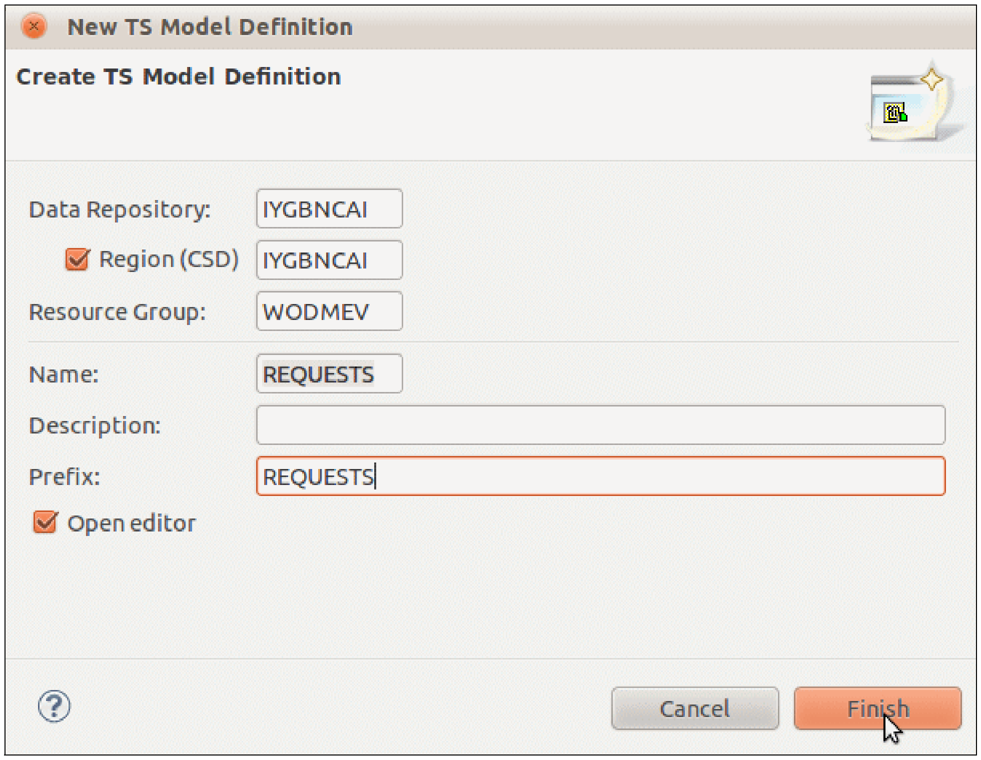
You can create the TS Queue by using CICS Explorer:

1. Open the **TS Model** view in CICS Explorer.
2. Right-click in the white space in the view, and right-click **New**.
3. Fill in the following information, as shown in Figure 8-24 on page 168:
   a. For Data Repository, type `IYGBNCAI`.
   b. Select **Region (CSD)** and **IYGBNCAI**.
   c. For Resource Group, type `WODMEV`.
   d. For Name, type `REQUESTS`.

e. For Prefix, type `REQUESTS`.

f. Select **Open editor**.

Click **Finish** to create the TS model definition.



Data Repository: IYGBNCAI
Region (CSD) ✓: IYGBNCAI
Resource Group: WODMEV
Name: REQUESTS
Description:
Prefix: REQUESTS
✓ Open editor

Figure 8-24 Creating the TS model definition

4. Right-click the newly created definition, and click **Install**.
5. Select your CICS, and click **OK**.

8.5 Running the scenario

CICS is now configured to emit the Request event when any CICS program writes the Request COBOL structure to the REQUESTS TS Queue.

When this event is emitted, it is received by the Decision Server Events run time and processed using the Event rule.

8.5.1 Enabling history in the Decision Server Event run time

Before running the sample COBOL application, configure the Decision Server Event run time to record history. This function allows the run time to remember the received events and the actions that are sent.

Important: The use of this function affects the performance of the Decision Server. It should be carefully considered whether this is required when running in a system in which high performance is required.

Enabling this feature allows you to monitor the events sent from CICS. To enable history in the Decision Server Events run time, see the *Configuring the event runtime to record history* topic in the IBM Operational Decision Manager Version 8.0.1 Information Center:

http://pic.dhe.ibm.com/infocenter/dmanager/v8r0m1/index.jsp?topic=%2Fcom.ibm.wodm.family.config.was%2Ftopics%2Ftsk_dse_config_runtime_recordhistory.html

8.5.2 Sample COBOL application to emit the Request event

To emit the Request event from CICS, the COBOL application that is shown in Example 8-2 is provided. The application imports the Request copybook and fills out the necessary data fields. It then writes the Request data structure to the REQUESTS TS queue to trigger the event emission.

Example 8-2 COBOL application that emits the Request event

```
CBL XOPTS(APOST SP COBOL3)
       PROCESS APOST
       PROCESS TRUNC(BIN) LIB RENT LIST
       PROCESS MAP XREF(FULL) NONAME

       IDENTIFICATION DIVISION.
       PROGRAM-ID. HBRRFQ.
       ENVIRONMENT DIVISION.
       DATA DIVISION.
       WORKING-STORAGE SECTION.
       COPY REQUEST.
       PROCEDURE DIVISION.
           Move "First" TO FIRST-NAME
           Move "Driver" TO LAST-NAME
           Move "00000" TO ZIPCODE
           Move 3 TO HOUSE-NUM
           MOVE 30 TO AGE
           MOVE "19000101" TO LIC-DATE
           MOVE "E" TO LIC-STATUS
           MOVE 0 TO NUMBER-ACCIDENTS
           Move "ABC123" TO VEC-ID
           MOVE "FASTCAR" TO MAKE
           MOVE "FASTEST" TO MODEL
           MOVE 1000 To VEC-VALUE
           MOVE "SU" TO VEC-TYPE

           EXEC CICS WRITEQ TS QUEUE('REQUESTS') FROM (REQUEST)
           END-EXEC.
           GOBACK.
```

Compile this COBOL program by using the Enterprise COBOL compiler for z/OS and install it into the same CICS into which the bundle definition was installed. Finally, define a transaction in CICS to run the HBRRFQ program.

This application is a stand-alone example of emitting an event from CICS by using a COBOL application. You can combine this example with the rule application in Chapter 3, “Getting started with business rules” on page 33. That way, the Request event is only emitted when a successful rule execution is completed.

For example, Example 8-3 shows a rule invocation that emits the Request event when a successful rule invocation is made.

Example 8-3 Combining a rule execution with an event emission

```
CALL 'HBRRULE' USING HBRA-CONN-AREA
           IF HBRA-CONN-COMPLETION-CODE NOT EQUAL HBR-CC-OK
               DISPLAY "invoke request rule failed"
           ELSE
               DISPLAY 'invoke request rule successful'
               EXEC CICS WRITEQ TS QUEUE('REQUESTS') FROM (REQUEST)
           END-IF
```

8.5.3 Emitting the event and firing the FollowUp action

This section describes how to emit and receive an event and then check the FollowUp action.

Emitting the event

Ensure that the Decision Server Events run time starts and runs the transaction in CICS, which causes the HBRRFQ program to run. This program causes the Request event to be emitted with the data, as shown in the sample COBOL application in 8.5.2, “Sample COBOL application to emit the Request event” on page 169.

The transaction must be run four times in quick succession to emulate a client requesting more than three quotations within the same hour. These transactions then cause the Decision Server Events run time to send the FollowUp action.

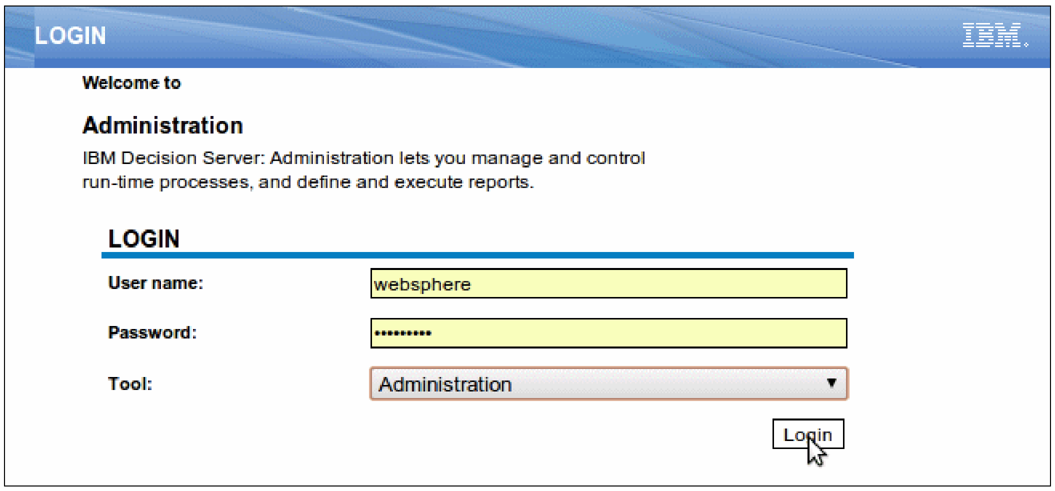
Receiving the event

Verify that the Decision Server Events run time received the Request event by logging in to the events administration console at this address:

`http://[hostname]:9088/wbe/common/login.jsp`

Then, follow these steps:

1. Select the Administration view, as shown in Figure 8-25.



Welcome to

Administration

IBM Decision Server: Administration lets you manage and control run-time processes, and define and execute reports.

LOGIN

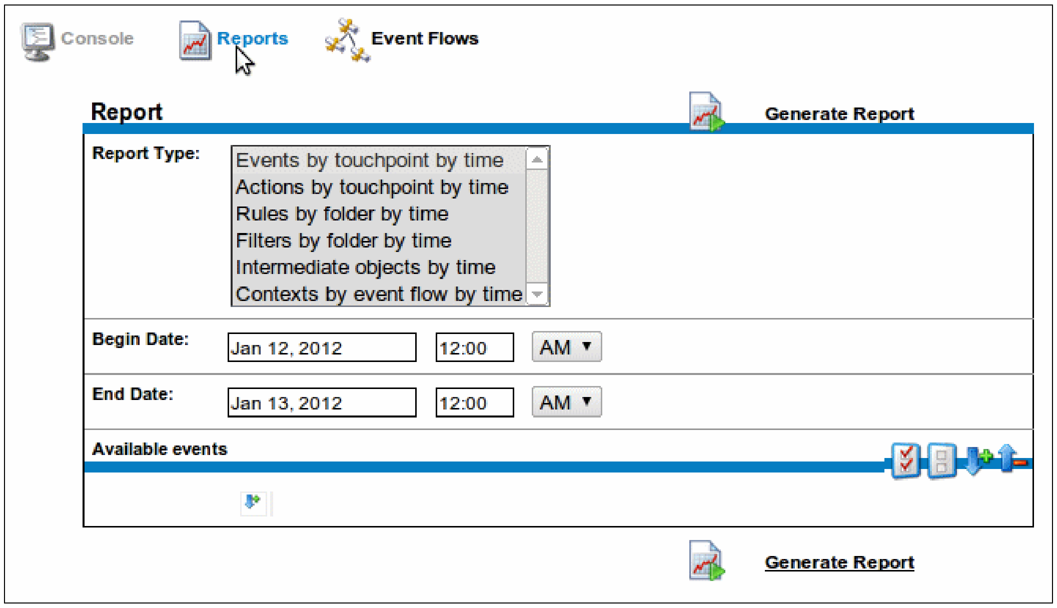
User name: websphere

Password: •••••••••

Tool: Administration

Figure 8-25 Logging in to the Administration console for events

2. When logged in, select the **Reports** tool, as shown in Figure 8-26.

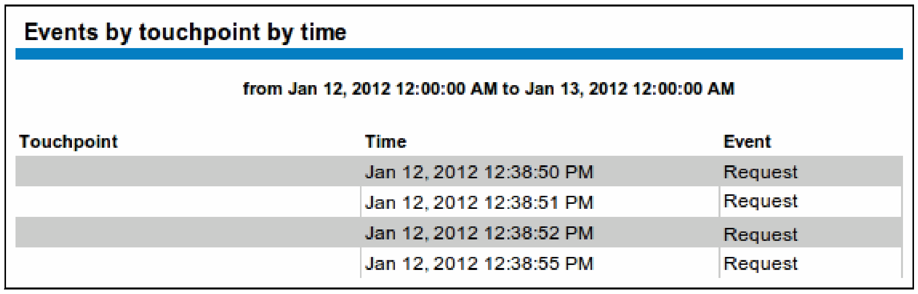


Report

Report Type:	Events by touchpoint by time Actions by touchpoint by time Rules by folder by time Filters by folder by time Intermediate objects by time Contexts by event flow by time
Begin Date:	Jan 12, 2012 12:00 AM
End Date:	Jan 13, 2012 12:00 AM

Figure 8-26 Running the generate reports function for the events run time

3. Click **Events by touch point by time** and click **Generate Report**. This selection shows all events that are received by the Decision Server Events run time, as shown in Figure 8-27.



Events by touchpoint by time

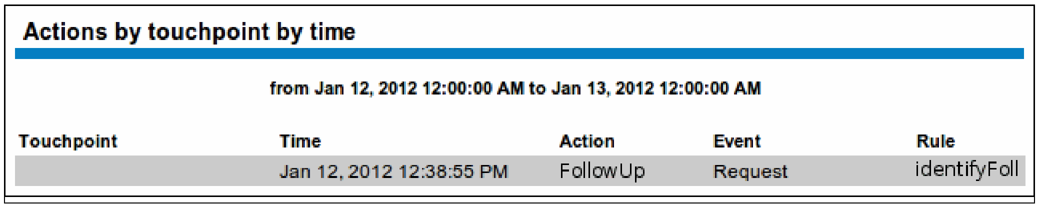
from Jan 12, 2012 12:00:00 AM to Jan 13, 2012 12:00:00 AM

Touchpoint	Time	Event
	Jan 12, 2012 12:38:50 PM	Request
	Jan 12, 2012 12:38:51 PM	Request
	Jan 12, 2012 12:38:52 PM	Request
	Jan 12, 2012 12:38:55 PM	Request

Figure 8-27 The events run time receives the Request event

Checking the action

Because four Request events were received within one hour, a FollowUp action was generated. To check the actions that fired, return to the Reports Tool. Click **Actions by touchpoint by time** and click **Generate Report**. This selection shows all actions that are fired by the events run time, as shown in Figure 8-28.



Actions by touchpoint by time

from Jan 12, 2012 12:00:00 AM to Jan 13, 2012 12:00:00 AM

Touchpoint	Time	Action	Event	Rule
	Jan 12, 2012 12:38:55 PM	FollowUp	Request	identifyFoll

Figure 8-28 The FollowUp action fired

This step completes the scenario for sending an event from CICS to the Decision Server Events run time. The next section, 8.6, "Using connectors to receive events from various z/OS sources" on page 172, describes various considerations when running the event technology connectors on z/OS.

8.6 Using connectors to receive events from various z/OS sources

Technology connectors are components that provide codeless connections for events and actions by various protocols, including SMTP, HTTP, and FTP. There are two categories of technology connectors: event connectors and action connectors. An *event connector* is used to receive events into the event run time. An *action connector* is used to send an action from the event run time to a target system.

A technology connector is associated with an individual event or action. All the specifications are defined as properties of that event or action in the Event Designer. When a technology connector is defined for an event or action, the icon of the event or action changes to help identify the type of technology connector. Enabling a technology connector requires no coding.

Section 8.2.6, "Configuring the technology connectors" on page 152, describes the full list of connectors that are available on z/OS. Considerations are listed when running these connectors on z/OS. If a connector was not described in this section, it has no special considerations for z/OS. For example, the Email connector uses SMTP to send or receive an email. The Email connector is configured in the same way for z/OS as for any other platform.

8.6.1 Connectors running in WebSphere Application Server

The event run time on Operational Decision Manager introduces a new type of connector that runs inside WebSphere Application Server. The HTTP, JMS, REST, SOAP, JDBC, and File connectors are available as WebSphere applications and are installed to the server where the event run time is installed. The connector applications are installed and updated automatically as required in response to the event run time reloading.

For information about the WebSphere connectors, see the *Running connector applications within the WebSphere Application Server environment* topic in the Decision Server V8.0.1 Information Center:

http://pic.dhe.ibm.com/infocenter/dmanager/v8r0m1/index.jsp?topic=%2Fcom.ibm.wodm.dserver.events.admin%2Ftopics%2Ftsk_dse_runningconnectorapps_wasenvt.html

The JMS queue connector runs as a WebSphere application and can be used to receive events and send actions by using WebSphere MQ on z/OS.

In this chapter, the HTTP connector was used to send an event from CICS to the Decision Server event run time. CICS also supports the JMS queue connector, which can be used to send the same event by using WebSphere MQ. Using WebSphere MQ increases the reliability of the transport, ensuring that the event is received despite any planned or unplanned interruptions.

8.6.2 Connectors running as a stand-alone batch job

The traditional method for running the technology connectors on z/OS is by using a stand-alone batch job that launches the Java process in which the connectors run. Use the WebSphere Application Server connectors where possible because they build on the WebSphere framework. This framework provides high availability and increased manageability of the connectors.

For more information, see the *Running the stand-alone technology connector application on z/OS* topic in the Decision Server V8.0.1Information Center:

http://pic.dhe.ibm.com/infocenter/dmanager/v8r0m1/index.jsp?topic=%2Fcom.ibm.wodm.dserver.events.admin%2Ftopics%2Ftsk_dse_runningstandaloneconnectorapp_zos.html

Part 2

System configuration

Multiple runtime environments are possible with Operational Decision Manager for z/OS. This part describes these environments and how to configure them. It contains the following chapters:

© Copyright IBM Corp. 2013. All rights reserved.

9

Prerequisites and considerations before you start

This chapter provides an overview of the teams that are needed to complete a configuration of Operational Decision Manager on z/OS. This chapter also includes a checklist to be completed with the client values before embarking on a configuration.

The following topics are covered in this chapter:

© Copyright IBM Corp. 2013. All rights reserved.

9.1 Runtime environments on z/OS

The runtime environments on z/OS all support the COBOL execution object model (COBOL XOM), including the Rule Execution Server running on WebSphere Application Server on z/OS.

Important: However, the COBOL XOM capability is not supported by the rule engine running on WebSphere Application Server on a distributed platform.

Figure 9-1 shows the runtime environments that can be configured on z/OS:

- zRule Execution Server for z/OS (zRES) hosted on CICS
- Stand-alone zRule Execution Server (zRES) for z/OS
- Rule Execution Server (RES) hosted on WebSphere Application Server

All of these run times support the COBOL XOM capability.

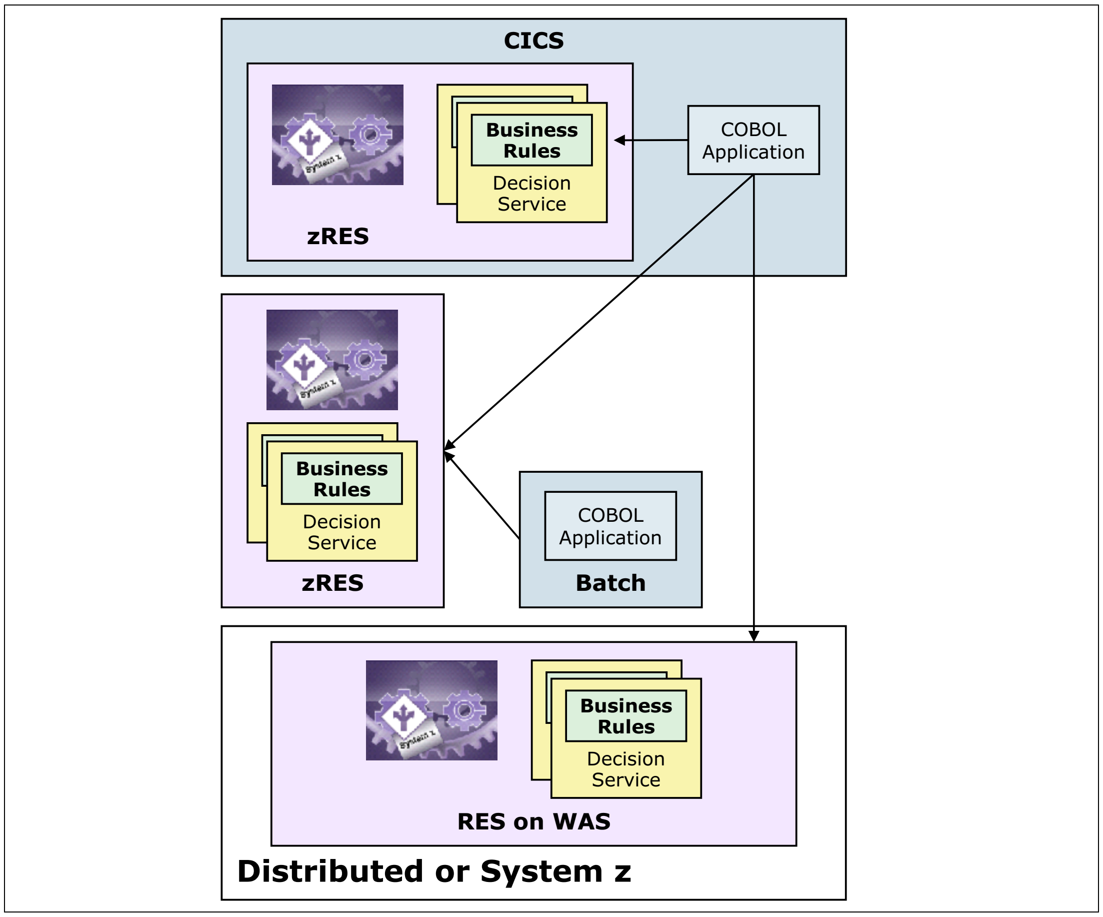


Figure 9-1 Runtime configurations on z/OS

9.1.1 Configuring the run times

You set up each runtime configuration by changing the related parameter values that are grouped into a number of partitioned data set (PDS) members as listed in Table 9-1 on page 179.

Table 9-1 Configuration parameters

Member name	Description
HBRBATCH	Used by the client to connect to the server.
HBRCICSD	DB2 on CICS.
HBRCICSJ	Used for CICS setup.
HBRCICSZ	Execution server with which CICS connects.
HBRCMMN	Common parameters between zRule Execution Server for z/OS and console.
HBRCNSL	Console parameters.
HBRPSIST	Member that defines the type of persistence used by the zRule Execution Server instance. This can be either DB2 or file system-based that is located in UNIX System Services.
HBRINST	Custom configuration values for the zRES instance.
HBRSCEN	Input for the Miniloan sample.
HBRWOLA	The member that defines the WebSphere Optimized Local Adapters (WOLA) connection for Operational Decision Manager to connect a COBOL program to a WebSphere Application Server instance of the Rule Execution Server.

For more information about the descriptions of the parameters in these members, see the *z/OS configuration and runtime variables* topic in the IBM Operational Decision Manager Version 8.0.1 Information Center:

http://pic.dhe.ibm.com/infocenter/dmanager/v8r0m1/index.jsp?topic=%2Fcom.ibm.wodm.family.config.zos%2Ftopics%2Fcon_ds_jcl_and_runtime_vars.html

Each configuration parameter is updated with the site-specific values when the HBRUUPTI job is used to configure a zRule Execution Server for z/OS instance. Use the job HBRUUPTI to help configure an Operational Decision Manager z/OS instance. This job takes the client values from the member HBRINST and stamps them into the related configuration parameter members and configuration JCL jobs for that runtime instance.

See Appendix B, "Configuring runtime values by using variables defined in HBRINST" on page 327 for the client values that need to be gathered before you configure Operational Decision Manager for z/OS.

9.1.2 Prerequisite checklist

Use Table 9-2 on page 180 to check that the z/OS system is at the correct prerequisite level. For the current information, see the IBM Decision Center for z/OS 8.0.1 on z/OS website:

http://pic.dhe.ibm.com/infocenter/prodguid/v1r0/clarity-reports/report/html/softwareReqsForProduct?deliverableId=1312191818820&osPlatform=z/OS#

Table 9-2 Prerequisites

Item	Value
z/OS level	z/11
Java level + service	6.0.1 (64-bit support)
DB2 + service	9.1
CICS + service	► CICS 4.1 + UK57632, UK69637, UK69654, and UK9655 or CICS 4.2. However, you can use CICS 3.*x* to run programs that connect to a stand-alone zRule Execution Server for z/OS. ► CICS 5.1
Logical partition (LPAR) environment	
zaap/ziip java/db2 workload	
WebSphere Application Server level	► V7 with Fix Pack 17 ► V8 with Fix Pack 4 ► V8.5
Operational Decision Manager	► HDM8010-ODM Base z/OS (mandatory) ► HDM8011-ODM and BR Common (mandatory) ► JDM8012-Events Component z/OS ► HDM8013-Rules Component z/OS ► JDM8014- Decision Center z/OS

9.2 Teams needed for installation and configuration

Before installing and configuring Operational Decision Manager on z/OS or Operational Decision Manager on WebSphere Application Server for z/OS, it is necessary to involve various groups that administer the products involved with Operational Decision Manager and to have the correct authorities to carry out the various initialization steps. Without these people, the configuration is likely to be unsuccessful. In particular, the jobs for creating the database can be precustomized by following the initial customization steps from the product documentation, but the customized jobs might not be applicable to every site. These JCL jobs are a suggested way of working and are examples that might require editing further for each location.

When considering an Operational Decision Manager environment, a number of infrastructure teams might need to be consulted. The following team functions can be grouped into one or many teams:

- Installation group: Responsible for installing products by using SMP/E for z/OS
- System programmers: Responsible for creating and starting tasks, z/OS file systems, datasets, and system configuration
- Security managers: Responsible for setting up IBM RACF® or equivalent groups and profiles
- Database administrators: Responsible for creating the rules and events DB2 repositories
- WebSphere Application Server administrators, if running on WebSphere Application Server
- CICS configurators, if running the rules engine in CICS
- IMS administrators, when running with IMS

9.3 Gathering the customizable information

After help is committed from the groups that are mentioned in 9.2, “Teams needed for installation and configuration” on page 180, gather all the information that is required to customize the JCL jobs. The task of customizing the JCL jobs is done by the job HBRUUPTI and uses the input that is supplied by the member HBRINST.

See Appendix B, “Configuring runtime values by using variables defined in HBRINST” on page 327 for details of the variables that are available in HBRINST.

9.4 Migration considerations

Operational Decision Manager is a collection of pieces that allow the business rules to be encapsulated away from the business application program. To migrate the rules on a different version of Operational Decision Manager requires the rules to be redeployed from one version of Operational Decision Manager to the new version of Operational Decision Manager. To “migrate to another version of Operational Decision Manager” is therefore a deployment task and a configuration of the new versioned server.

You use the following steps:

1. Back up the working directories for the server instances.
2. Take an image copy of your database.
3. Synchronize and export the rule projects.
4. Remove the instance directories.
5. Drop the database artifacts. The sequences need to be dropped separately if you run Decision Center because they are not related to the database.
6. Remove the PROC members from the SYSx.USERPROC or equivalent location.
7. Perform a full SMP/E for z/OS installation of the new version.
8. By using the same client values from the old version, place these values into the new HBRINST and update any new variables. Keep the name of the instances the same and run all the configuration steps.
9. Start the versioned instances and check the diagnostics.
10. Import and re-deploy the ruleapp.

10 zRule Execution Server for z/OS stand-alone server

This chapter describes the process of setting up a stand-alone zRule Execution Server for z/OS (zRES) server. This chapter also describes how the different types of database connections are set up from the zRES server.

The following topics are covered in this chapter:

10.1 Running on z/OS stand-alone

This section describes how to run the rule engine on z/OS to consume batch work. The following components make up the zRES server:

- A stand-alone address space that hosts the rules engine
- A zRES console to administer the zRES server
- A database to hold the rules

Figure 10-1 shows the runtime environment of a stand-alone zRule Execution Server for z/OS.

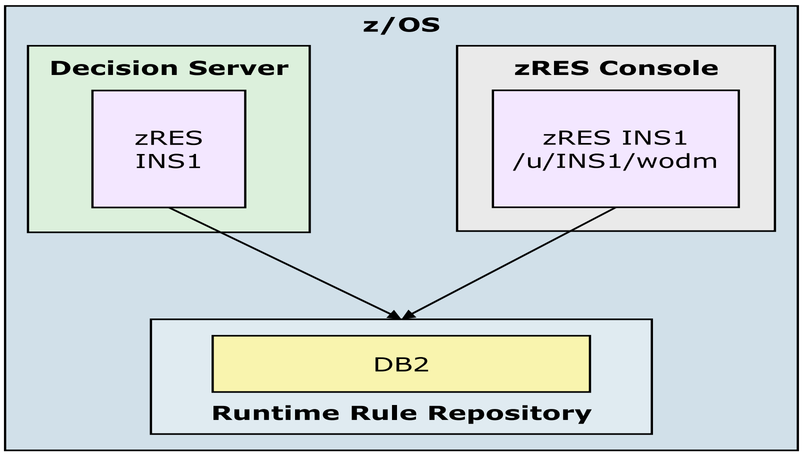


Figure 10-1 zRes stand-alone server

10.1.1 Configuring the stand-alone zRule Execution Server for z/OS

To configure the stand-alone zRule Execution Server for z/OS server, you must edit the values in the HBRINST member and run the HBRUUPTI job using the HBRINST member as input to this job. This action creates a set of data sets that are configured for this zRES instance.

For example, the output from HBRUUPTI can produce the runtime configuration data sets that are shown in Table 10-1 and Example 10-1 for one zRES instance.

Table 10-1 zRES instance configuration data sets

Data set	Description
…..SHBRJCL	Instance configuration jobs
….SHBRPARM	Instance configuration values
…..SHBRPROC	Configured zRule Execution Server for z/OS started tasks
…..SHBRWASC	WebSphere Application Server configuration

Example 10-1 Example zRES runtime configuration data sets

```
++HBRWORKDS++.HBR1.SHBRJCL
++HBRWORKDS++.HBR1.SHBRPARM
++HBRWORKDS++.HBR1.SHBRPROC
++HBRWORKDS++.HBR1.SHBRWASC
```

This step is repeated for each zRule Execution Server for z/OS instance that you want to create. Each zRule Execution Server for z/OS has a unique identifier that is given by the HBRSSID value.

Defining the zRule Execution Server for z/OS instance working directory

In addition to a set of data sets for each zRule Execution Server for z/OS instance, each instance also has a file system location. The run time uses the file system location, which is called a *working directory*.

Running the HBRCRTI job from the instance data set SHBRJCL sets up the working directory in z/OS UNIX System Services for the zRule Execution Server for z/OS instance.

10.1.2 Creating data sets for the zRule Execution Server for z/OS instance

This section describes how to create the data sets that are changed for the zRule Execution Server for z/OS instance.

Customizing the HBRINST member of SHBRPARM

This section details which values must be updated. For more information, see Appendix B, “Configuring runtime values by using variables defined in HBRINST” on page 327. This appendix contains tables that explain how to customize the values in SHBRPARM(HBRINST) for your system environment.

Preferred practice: A preferred practice is to copy the target library SHBRPARM partitioned data set (PDS) member HBRINST as a new member within the PDS, for example, HBR1INST. The HBRUUPTI job is modified to point to this instance of HBR1INST.

The INLINES data definition (DD) card must point to this new HBRINST member:

```
//INLINES  DD DISP=SHR,DSN=++HBRHLQ++.SHBRPARM(HBR1INST)
```

When this job completes, the output to the HBRWORKDS puts this member as HBRINST instead of HBR1INST (or your member name) within the customized SHBRPARM data set.

When creating a new instance of zRES, a copy of the HBRINST member is created to track all server changes. For example, the first data set that is created can be called HBR1INST, which covers HBR1 (and its corresponding list of subsystem identifiers (SSIDs)).

Setting up database persistence

When setting up the first zRES server, there are several values within the HBRINST member that need to be updated. Example 10-2 on page 186 lists the variables that are updated in the first instance using database persistence.

For more information about each of these variables, see Appendix B, “Configuring runtime values by using variables defined in HBRINST” on page 327.

Type 2 connection: By default, zRES establishes a type 4 connection. For details about how to use a type 2 connection, see 10.4, “Setting up the database connection” on page 200.

Example 10-2 List of variables that are used for database persistence

```
DB2HLQ
DB2RUNLIB
DB2SUBSYSTEM
DB2LOCATION
DB2VCAT
DB2ADMIN
DB2CURRSQLID
RESDATABASE
RTSDATABASE
EVDATABASE
RESSTOGROUP
RTSSTOGROUP
EVSTOGROUP
DB2TABLEBP
DB2INDEXBP
DB2LOBBP
DB2SAMPLEPROGRAM
DB2SAMPLEPROGRAMPLAN
DB2BP4K
DB2BP8K
DB2BP32K
DB2USER
DB2PSWD
DB2CONSTR
DB2JARLOCN
DB2NATIVELOC
```

10.1.3 Creating the working datasets using HBRUUPTI

The HBRUUPTI member that is within the ++HBRHLQ++.SHBRJCL data set uses the values in the HBRINST member to populate the SHBRJCL, SHBRPARM, SHBRPROC, and SHBRWASC data sets that are changed to your system environment.

Changing HBRUUPTI

You must perform the following steps to change HBRUUPTI to create the new working data sets for the zRule Execution Server for z/OS server. Customize the HBRINST data set to your system environment by using the tables in Appendix B, “Configuring runtime values by using variables defined in HBRINST” on page 327.

The preferred practice is to copy the target library SHBRPARM member HBRINST as a new member within the SHBRPARM PDS and to create a new PDS with a name similar to HBRINST (for example, HBR1INST).

Preferred practice: If using the preferred practice for the installation, update the INLINES DD card as described with the member for the current configuration. For example, if the first instance is HBR1, the member that is created is HBR1INST.

When running the customization job, SHBRJCL(HBRUUPTI), the HBRINST field update to the INLINES DD card must be changed to reflect this new location of the HBRINST file:

```
//INLINES DD DISP=SHR,DSN=++HBRHLQ++.SHBRPARM(HBR1INST)
```

When you run this job with the new HBRINST, the output working data set member does not show the updated values in the member HBRINST, but all the other members are modified to the user’s values.

To change HBRUUPTI, follow these steps:

1. Update the following line in HBRUUPTI that shows the target library high-level qualifier (HLQ) that is set to the value HBRHLQ. Update this line with the value of your HBRHLQ from Table B-1 on page 328. In this example, it is set to ODM.V8R0M1.TLIB:

   ```
   SET HBRHLQ=ODM.V8R0M1.TLIB
   ```

2. Update the INLINES line, as shown in Figure 10-2, to match where the customization member is created, which, by default, is in HBRHLQ.SHBRPARM(HBRINST). This points to the instance of HBRINST that you use for the customization. If you use the preferred practice, ensure that you update this to the correct value you have set.

```
//          SET HBRHLQ=WODM.V8R0M1.TLIB
//HBRUUPTI EXEC PGM=IKJEFT01,REGION=2M,DYNAMNBR=99
//SYSPROC  DD DISP=SHR,DSN=&HBRHLQ..SHBREXEC
//INLINES  DD DISP=SHR,DSN=&HBRHLQ..SHBRPARM(HBR1INST)
//SYSTSIN  DD *
```

Figure 10-2 Changing the INLINES line

3. Submit the job to create the working data sets for the zRule Execution Server instance. This job creates the following data sets:

   – ++HBRWORKDS++.++HBRSSIDLIST++.SHBRJCL
   – ++HBRWORKDS++.++HBRSSIDLIST++.SHBRPARM
   – ++HBRWORKDS++.++HBRSSIDLIST++.SHBRPROC
   – ++HBRWORKDS++.++HBRSSIDLIST++.SHBRWASC

   For this example, if the HBRSSIDLIST were HBR1 and HBR2, and the HBRWORKDS was WODM.V8R0M1, the following datasets are created:

   – WODM.V8R0M1.HBR1.SHBRJCL
   – WODM.V8R0M1.HBR1.SHBRPARM
   – WODM.V8R0M1.HBR1.SHBRPROC
   – WODM.V8R0M1.HBR1.SHBRWASC
   – WODM.V8R0M1.HBR2.SHBRJCL
   – WODM.V8R0M1.HBR2.SHBRPARM
   – WODM.V8R0M1.HBR2.SHBRPROC
   – WODM.V8R0M1.HBR2.SHBRWASC

   ++HBRSSIDLIST++ : From this point, the ++HBRSSIDLIST++ is simplified to ++SSID++, because the first member in the SSID is used where this is referenced. So, if the HBRSSIDLIST is HBR1 and HBR2, SSID refers to HBR1.

10.1.4 Creating the working directories in UNIX System Services

After submitting the HBRUUPTI job, navigate to the following PDS, and open the job HBRCRTI:

```
++HBRWORKDS++.++SSID++.SHBRJCL
```

This job runs an **`hbrcrtin.sh`** script that is in the ++HBRINSTPATH++, which is set in Table B-1 on page 328. This job creates the ++HBRWORKPATH++ directory in the UNIX System Services, which contains the following directories:

- `config`
- `logs`
- `res_data`

- `res_xom`
- `work`

The `config` directory contains the XML files that are required for the zRule Execution Server to start, including the run time and the console.

10.2 Configuring the stand-alone zRule Execution Server for z/OS

This section covers configuring the stand-alone zRule Execution Server for z/OS or one zRule Execution Server group with one console.

For every type of setup, you must initially configure a group of stand-alone zRule Execution servers (if you choose, this group can have only one member, although no failover for execution is present) and one console that connects these. This is required when setting up the other configurations that are described later in this document.

10.2.1 Defining a new subsystem for zRule Execution Server for z/OS

The first step for the configuration of the stand-alone zRule Execution Server for z/OS is to define the subsystem in which the new instances run. The systems programmer must perform this task.

The following **SETSSI** command must be run, where *++HBRSSID++* is the subsystem ID that was set in Table B-1 on page 328 under ++HBRSSIDLIST++. Run this command for each instance:

```
SETSSI ADD,SUBNAME=++HBRSSID++
```

For example, if HBR1 and HBR2 are in the list, this command is run for each instance. Using the example of HBR1 and HBR2 as the SSIDs, the following commands are correct:

```
SETSSI ADD,SUBNAME=HBR1
SETSSI ADD,SUBNAME=HBR2
```

10.2.2 Creating the started tasks (HBRXCNSL and HBRXMSTR)

The next task in the configuration is to copy HBRXCNSL and HBRXMSTR to the system PROCLIB. The HBRXCNSL and HBRXMSTR members are for the started tasks. These started tasks are attached to three PARMLIB members that are used for the definitions of various parameters for the zRule Execution Server. If you use the HBRUUPTI job, you do not need to modify these attached PARMLIB members unless your environment requires the change.

Production usage: You can use the file system setup in proofs of concept (POC) and the initial configuration. However, for production, the configuration must have a DB2 persistence layer. This is set by the member HBRPSIST and was set in Table B-1 on page 328 with the value for ++HBRPERSISTENCETYPE++.

Adding HBRXMSTR and HBRXCNSL to SYS1.PROCLIB

The next task is to copy HBRXMSTR and HBRXCNSL to `SYS1.PROCLIB` (or a similar PROCLIB on your environment). When copying over the data set members, you must change the names from HBRXMSTR to ++HBRSSID++MSTR and from HBRXCNSL to ++HBRSSID++CNSL. Follow these steps:

1. Copy the ++HBRWORKDS++.++SSID++.SHBRPROC(HBRXMSTR) to `SYS1.PROCLIB`(++SSID++MSTR).
2. Copy the ++HBRWORKDS++.++SSID++.SHBRPROC(HBRXCNSL) to `SYS1.PROCLIB`(++SSID++CNSL).
3. Repeat the process for HBRXMSTR for all members of the HBRSSIDLIST.

SYS1.PROC: `SYS1.PROCLIB` is the default. You need to change it to match your environment. ++HBRSSID++ was set in Table B-1 on page 328.

Authorizing the server instance as a started task

Authorize both ++HBRSSID++MSTR and ++HBRSSID++CNSL as started task procedures executing with the same user ID. Use the following commands:

```
RDEFINE STARTED ++HBRSSID++MSTR.* STDATA(USER(<HBRSSID_USER>)
GROUP(<HBRSSID_GROUP>)

RDEFINE STARTED ++HBRSSID++CNSL.* STDATA(USER(<HBRSSID_USER>)
GROUP(<HBRSSID_GROUP>)
```

HBRSSID_USER is the server user ID and *HBRSSID_GROUP* is the RACF security group name that is provided to you by your security administrator.

The started task definitions

Whether you are starting the stand-alone zRule Execution Server for z/OS started task or the CICS zRule Execution Server for z/OS started task, both started tasks require that the configuration parameters are provided to the job. The parameters are provided by the DD card HBRENVPR on each started task. The DD card HBRENVPR specifies the input parameter members.

10.2.3 Securing the zRule Execution Server for z/OS for z/OS resources

With Operational Decision Manager, you can secure the resources, files, and functions with RACF. This section describes how to create this security for the server using RACF.

Security options

If running the zRule Execution Server for z/OS in production, you might want to secure all or part of the zRule Execution Server for z/OS resources. However, if you plan to run the server in a testing environment, you might want security disabled. You can use the following options for security.

Within the file system, you can secure the following resources:

- The working directory so that only the authorized user IDs can access internal data
- The installation directory so that only the authorized user IDs can access the files that are needed to run the server

Using RACF, you can secure the following resources:

- You can secure the server resources that you use to perform the following tasks:
  - Issue zRule Execution Server for z/OS commands from the z/OS console (or equivalent).
  - Sign on to the Rule Execution Server console.
  - Connect to the zRule Execution Server for z/OS to execute rulesets.
- You can secure a subset of server resources. For example, you can secure access to the Rule Execution Server console only.

Securing access to the working directory and installation directory

The working directory contains data that includes logs from the zRule Execution Server for z/OS, component details, and trace output. The installation directory contains the necessary files to run the zRule Execution Server for z/OS server. The server user ID needs to read and execute access for ++HBRWORKPATH++, ++HBRINSTPATH++, the zRule Execution Server for z/OS work path, and the Operational Decision Manager installation directory, if the permissions are to be changed on these directories.

Creating the RACF classes for securing server resources

You can manage zRule Execution Server for z/OS by using RACF classes. You must create the three RACF classes through the use of the ++HBRWORKDS++.SHBRJCL(HBRCRECL) job. To secure the resources for the zRule Execution Server for z/OS instance, your RACF administrator must run the HBRCRECL job. This job can be run from any of the members within the ++HBRSSIDLIST++.

Creating the RACF classes

Using RACF, you can secure the following information:

- Ask the RACF administrator to run the HBRCRECL job or extract the code to use the preferred execution methods to perform the following tasks:
  - Issue zRule Execution Server for z/OS commands from the z/OS console (or equivalent).
  - Sign on to the Rule Execution Server console.
  - Connect to the zRule Execution Server for z/OS to execute rulesets.
- You can secure a subset of server resources. For example, you can secure access to the Rule Execution Server console only.

When your RACF administrator runs the HBRCRECL job, the job creates three RACF classes: HBRADMIN, HBRCONN, and HBRCMD. Table 10-2 on page 191 explains the characteristics of these classes.

Table 10-2 RACF classes created by ++HBRWORKDS++.SHBRJCL(HBRCRECL)

Class	Description
HBRADMIN	This class controls whether server security and security for specific server resources are enabled or disabled.
HBRCONN	This class specifies the user IDs that are authorized to connect to the zRule Execution Server for z/OS and to execute rulesets. This class is ignored if server security is disabled.
HBRCMD	This class specifies the user IDs that are authorized to issue zRule Execution Server for z/OS commands, such as **START**, **STOP**, **PAUSE**, or **RESUME** from the z/OS console (or equivalent). This class is ignored if server security is disabled.

POSIT: The supplied JCL in HBRCRECL gives a POSIT value of 128. Change POSIT, as required, to match your security environment requirements.

After running the HBRCRECL job, give the server user ID read access to the class profile by using the following commands:

```
PERMIT BPX.SERVER CLASS(FACILITY) ID(<HBRSSID_USER>) ACCESS(READ)
SETROPTS RACLIST(FACILITY) REFRESH
```

In this example, <*HBRSSID_USER*> represents the server user ID, which is the ID under which the server runs.

Disabling types of security

In Operational Decision Manager, you can optionally disable all types of security or parts of the security. When the HBRADMIN class is created, security is enabled on all zRule Execution Server for z/OS instances. Security can be enabled and disabled, as required. On a test system, you might want no security on the instance so that you can test more freely, but you do not have to disable the security to all instances that are used.

To disable levels of security, you must apply separate profiles to the HBRADMIN class. Table 10-3 lists the profiles that can be added to the HBRADMIN class by using the following commands:

```
RDEFINE HBRADMIN <RESOURCE_PROFILE> UACC(NONE)
SETROPTS RACLIST(HBRADMIN) REFRESH
```

Table 10-3 Resource profiles to disable parts of security on the zRule Execution Server

Resource profile	Description
++HBRSSID++.NO.SUBSYS.SECURITY	This profile disables all security for a particular server instance. If server security is disabled, HBRCONN and HBRCMD classes are not used.
++HBRSSID++.NO.CONNECT.SECURITY	This profile disables connection security for a particular server instance, but it maintains other types of security.
++HBRSSID++.NO.RESCONSOLE.SECURITY	This profile disables console security for a particular server instance, but it maintains other types of security.
++HBRSSID++.NO.COMMAND.SECURITY	This profile disables command security for a particular server instance, but it maintains other types of security. If you disable command security, any user can issue a zRule Execution Server for z/OS command from the z/OS console.

Replace ++HBRSSID++ with a value from the *++HBRSSIDLIST++* variable. Repeat for each server listed in the *++HBRSSIDLIST++* variable for which you want to disable security. Table B-1 on page 328 has details about the *++HBRSSIDLIST++* variable.

For more information, see the *Managing server security* topic in the IBM Operational Decision Manager Version 8.0.1 Information Center:

http://pic.dhe.ibm.com/infocenter/dmanager/v8r0m1/index.jsp?topic=%2Fcom.ibm.wodm.family.config.zos%2Ftopics%2Ftpc_ds_manage_zres_security.html

Continue to one of the following sections, depending on the profile that you plan to use:

- If the CONNECT profile is used, go to "Managing connection security" on page 192.
- If the RESCONSOLE profile is used, go to "Managing console security" on page 193.
- If the COMMAND profile is used, to go "Managing command security" on page 194.

Managing connection security

You set up connection security to ensure that only authorized user IDs can connect to the zRule Execution Server for z/OS instance to execute rulesets. Connection security uses the HBRCONN RACF class to authorize user IDs to connect to the server instance.

If the profile ++HBRSSID++.NO.SUBSYS.SECURITY or ++HBRSSID++.NO.CONNECT.SECURITY is used, the HBRCONN class is ignored.

To implement connection security, you must authorize the user ID under which the server runs and the user IDs of any applications that execute rulesets. The following steps are required for authorizing user IDs to the HBRCONN class:

1. The resource profile needs the server instance defined to the HBRCONN class. Execute the following command first to create the resource profile:

   ```
   RDEFINE HBRCONN ++HBRSSID++ UACC(NONE)
   ```

2. Give the server user ID UPDATE access to the ++HBRSSID++ resource profile by using the following command:

   ```
   PERMIT ++HBRSSID++ CLASS(HBRCONN) ID(<HBRSSID_USER>) ACCESS(UPDATE)
   ```

   UPDATE access: The server instance fails to initialize if the HBRCONN class does not have UPDATE access. This requirement does not affect a server instance with ++HBRSSID++.NO.SUBSYS.SECURITY or ++HBRSSID++.NO.CONNECT.SECURITY.

3. Refresh the ++HBRSSID++ resource profile by using the following command:

   ```
   SETROPTS RACLIST CLASS(HBRCONN) REFRESH
   ```

Next, authorize the applications by running the following steps:

1. Give READ access to the ++HBRSSID++ resource profile to each user that you want to authorize. Use the following command:

   ```
   PERMIT ++HBRSSID++ CLASS(HBRCONN) ID(<USER_ID>) ACCESS(READ)
   ```

   User IDs: For batch jobs, <*USER_ID*> is the RACF user ID that is used by the batch job. For CICS transactions, <*USER_ID*> is the user ID that is assigned to the CICS address space.

2. Refresh the ++HBRSSID++ resource profile by using the following command:

   ```
   SETROPTS RACLIST(HBRCONN) REFRESH
   ```

Managing console security

You use console security to ensure that there is control on the users that can access the zRule Execution Server for z/OS console. The zRule Execution Server for z/OS console security controls the ability to sign on to the zRule Execution Server for z/OS console. If security is enabled, users must enter a user ID and password to sign on.

If the profile ++HBRSSID++.NO.SUBSYS.SECURITY or the profile ++HBRSSID++.NO.RESCONSOLE.SECURITY is used, the HBRADMIN class is ignored.

A standard set of roles exists within the zRule Execution Server for z/OS that gives access rights to users. Enable console security by assigning user IDs to roles and then authorizing the roles to access the console.

Table 10-4 shows the profiles and the roles that they represent. The roles are listed in order of increasing authority. RESMON is the lowest authority, and RESADMIN is the highest authority. ++HBRSSID++ is the ID of the subsystem where the server runs.

Table 10-4 zRule Execution Server for z/OS console security profiles

Resource profile	**Role description**
++HBRSSID++.ROLE.RESMON	Users with monitoring rights are only allowed to view and explore RuleApps, rulesets, decision services, Execution Units (XUs), and statistics. These users are not allowed to modify these entities. They can also select a trace configuration and view and filter trace information in Decision Warehouse. This authority applies only to Rule Execution Server on WebSphere Application Server for z/OS.
++HBRSSID++.ROLE.RESDEP	In addition to monitoring rights, users with deploying rights are allowed to deploy RuleApp archives, to edit and remove entities (RuleApps, rulesets, decision services, Java execution object module (XOM) resources, and libraries), and to run diagnostics.
++HBRSSID++.ROLE.RESADMIN	Users with administrator rights have full control over the deployed resources and access to information about the server. They can perform the following actions: ► Deploy, browse, and modify RuleApps, Java XOM resources, and libraries ► Monitor the decision history, purge the history, and back up the history. ► Select a trace configuration, view and filter trace information, and clear trace information in Decision Warehouse. ► Run diagnostics and view server information.

Perform the following steps to enable console security:

1. Define each resource profile, as shown in Table 10-4 to the HBRADMIN class. Use the following commands to define all three roles:

   ```
   RDEFINE HBRADMIN ++HBRSSID++.ROLE.RESMON UACC(NONE)
   RDEFINE HBRADMIN ++HBRSSID++.ROLE.RESDEP UACC(NONE)
   RDEFINE HBRADMIN ++HBRSSID++.ROLE.RESADMIN UACC(NONE)
   ```

2. Assign each user ID to one of the resource profiles by using the following commands for the three roles:

   ```
   PERMIT ++HBRSSID++.ROLE.RESMON UACC(NONE)
   PERMIT ++HBRSSID++.ROLE.RESDEP UACC(NONE)
   PERMIT ++HBRSSID++.ROLE.RESADMIN UACC(NONE)
   ```

3. Refresh the HBRADMIN class using the following command:

   ```
   SETROPTS RACLIST(HBRADMIN) REFRESH
   ```

Managing command security

You use command security to ensure that only authorized users can issue zRule Execution Server for z/OS commands on the zRule Execution Server for z/OS console. Command security uses the HBRCMD RACF class to authorize user IDs to issue zRule Execution Server for z/OS commands.

If the profile ++HBRSSID++.NO.SUBSYS.SECURITY or the profile ++HBRSSID++.NO.COMMAND.SECURITY is used, the HBRCMD class is ignored.

When enabling command security on the zRule Execution Server for z/OS console, you must define a resource profile to the HBRCMD class for each command that you want to secure. Use the commands that are listed in Table 10-5 to secure the zRule Execution Server for z/OS console commands. ++HBRSSID++ is the ID of the subsystem where the server runs.

Table 10-5 zRule Execution Server for z/OS command security profiles

Resource profile	**Command**	**Command description**
++HBRSSID++.SET.TRACE	SET TRACE	Ability to turn trace on or off
++HBRSSID++.SET.RESCONSOLE	SET RES-CONSOLE	Start or stop the zRule Execution Server for z/OS console
++HBRSSID++.SET.PAUSE	PAUSE	Pause a job on the zRule Execution Server for z/OS
++HBRSSID++.SET.RESUME	RESUME	Resume a job on the zRule Execution Server for z/OS

To authorize users to issue zRule Execution Server for z/OS commands, perform the following steps:

1. If you want to limit any of the commands in Table 10-5 to authorized user IDs, you must define the resource profiles to the following HBRCMD class commands:

```
RDEFINE HBRCMD ++HBRSSID++.SET.TRACE UACC(NONE)
RDEFINE HBRCMD ++HBRSSID++.SET.RESCONSOLE UACC(NONE)
RDEFINE HBRCMD ++HBRSSID++.SET.PAUSE UACC(NONE)
RDEFINE HBRCMD ++HBRSSID++.SET.RESUME
```

2. If you want to limit any of the commands in Table 10-5 to authorized user IDs, you must permit the resource profiles to the following HBRCMD class commands:

```
PERMIT ++HBRSSID++.SET.TRACE CLASS(HBRCMD) ID(<USER_ID>) ACCESS(UPDATE)
PERMIT ++HBRSSID++.SET.RESCONSOLE CLASS(HBRCMD) ID(<USER_ID>) ACCESS(UPDATE)
PERMIT ++HBRSSID++.SET.PAUSE CLASS(HBRCMD) ID(<USER_ID>) ACCESS(UPDATE)
PERMIT ++HBRSSID++.SET.RESUME CLASS(HBRCMD) ID(<USER_ID>) ACCESS(UPDATE)
```

User IDs: For batch jobs, <*USER_ID*> is the RACF User ID that is used by the batch job. For CICS transactions, <*USER_ID*> is the user ID that is assigned to the CICS address space.

3. Refresh the HBRCMD class by using the following command:

```
SETROPTS RACLIST(HBRCMD) REFRESH
```

10.2.4 Starting the new instance

After completing the security setup and configurations, start the new server instance through the z/OS console.

Authorizing the load library

If you are setting up your first instance on the logical partition (LPAR), you must authorize the load library. You must perform the following steps:

1. Add the ++HBRHLQ++.SHBRAUTH load library to the authorized program facility (APF)-authorized libraries by using the following command:

   ```
   SETPROG APF,ADD,DSNAME=++HBRHLQ++.SHBRAUTH,SMS
   ```

2. Provide the RACF authorization for the ++HBRHLQ++.SHBRAUTH load library by using the following two commands:

   ```
   RALTER PROGRAM * ADDMEM('++HBRHLQ++.SHBRAUTH'//NOPADCHK)
   SETROPTS WHEN(PROGRAM) REFRESH
   ```

   Note that ++HBRHLQ++ is the product installation target library HLQ for the SHBRAUTH PDS.

Starting a server instance

To start a new server instance, issue the following command:

```
START ++HBRSSID++MSTR
```

Replace ++HBRSSID++ with each member of the ++HBRSSIDLIST++ to start all of the MASTER address spaces.

If the server does not start, look at the output of the HBRMSTR job to see why it did not start. Typically, the server does not start for following reasons:

- The load library was not APF-authorized.
- ++HBRINSTPATH++ does not point to the correct UNIX System Services directory (if you go to the location that you put in the ++HBRINSTPATH++, it shows directories that include `IBM0`, `IBM1`, `IBM2`, `IBM3`, `IBM4`, `lib`, `shared`, and so on).
- If you use symbolic links on ++HBRINSTPATH++ or ++HBRWORKPATH++, these links might not link correctly. Therefore, you must verify the link.
- You did not execute the RACF security commands. Verify whether the RACF security commands were run by using the resource profile setup. Ensure that the commands executed and ensure that the user that started the server is authorized to start the server.
- The ports that were used for ++HBRCONSOLEPORT++ and ++HBRCONSOLECOMHOST++ were already in use by another application.
- The same port is used for ++HBRCONSOLEPORT++ and ++HBRCONSOLECOMHOST++.

10.2.5 Logging on and performing diagnostics

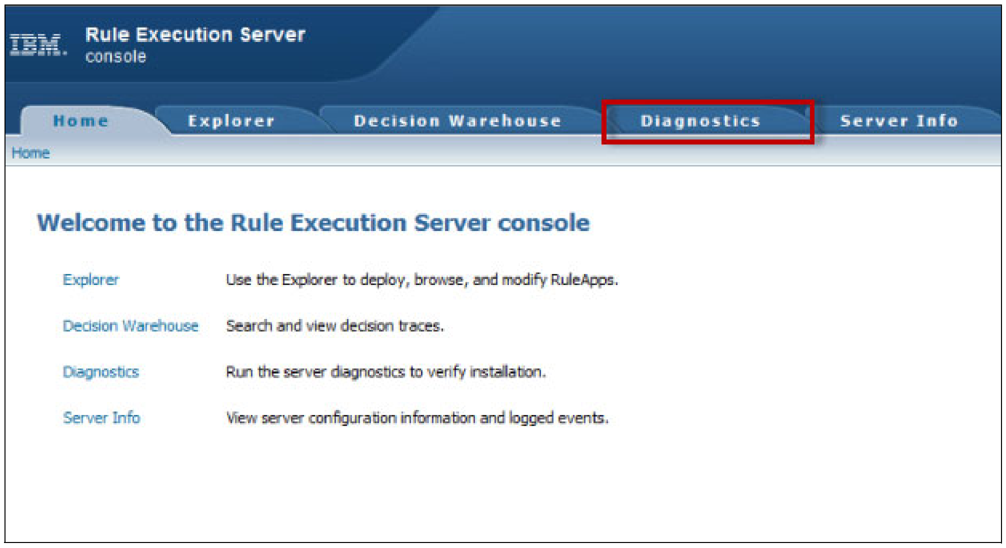
Now that the stand-alone zRule Execution Server for z/OS console is up and running, you must run diagnostics on it. You must use a RACF ID that is part of the RESADMIN group, which can run the diagnostics on the Rule Execution Server. This section describes logging on to the zRule Execution Server for z/OS console and performing diagnostics.

Perform the following steps:

1. Go to the following URL and log in with a RACF user ID and password combination that is part of the RESADMIN group:

   ```
   http:// ++HBRCONSOLECOMHOST++:++HBRCONSOLEPORT++/res
   ```

2. When you are logged in, you see a console similar to the console that is shown in Figure 10-3. Click the **Diagnostics** tab.



Welcome to the Rule Execution Server console

Explorer	Use the Explorer to deploy, browse, and modify RuleApps.
Decision Warehouse	Search and view decision traces.
Diagnostics	Run the server diagnostics to verify installation.
Server Info	View server configuration information and logged events.

Figure 10-3 Rule Execution Server console Welcome panel

3. The Diagnostics view is displayed. Click **Run Diagnostics**, as shown in Figure 10-4.

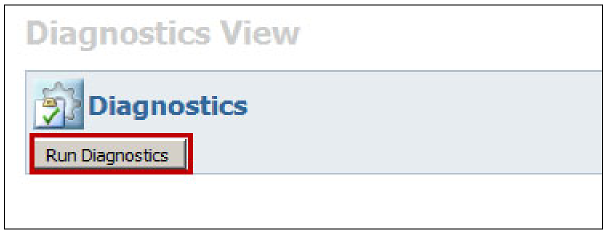
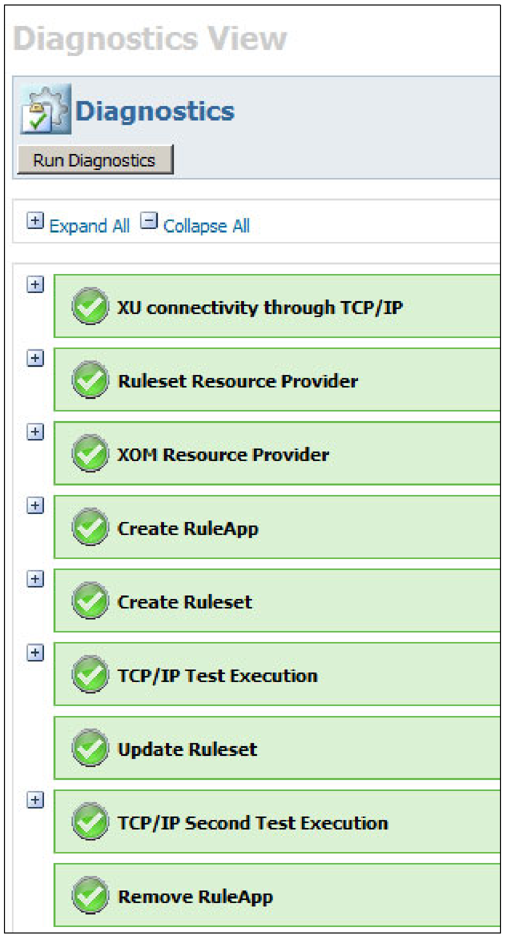


Figure 10-4 Rule Execution Server Diagnostics view

4. When you run the diagnostics, successful diagnostics show results with green check marks. If there are any issues, the diagnostics tool helps you troubleshoot the problem. Figure 10-5 shows a successful deployment.


Diagnostics View

Diagnostics

Run Diagnostics

Expand All Collapse All

✓ XU connectivity through TCP/IP

✓ Ruleset Resource Provider

✓ XOM Resource Provider

✓ Create RuleApp

✓ Create Ruleset

✓ TCP/IP Test Execution

✓ Update Ruleset

✓ TCP/IP Second Test Execution

✓ Remove RuleApp

Figure 10-5 Rule Execution Server Diagnostics successful test

The Rule Execution Server is now available for use. You can start using this instance of the zRule Execution Server for z/OS in your environment for rules execution.

10.3 Managing multiple zRule Execution Server for z/OS servers on one LPAR

Figure 10-6 on page 198 shows one LPAR with two zRule Execution Server for z/OS batch servers that are administered by one console. The second zRule Execution Server for z/OS (INS2) is started with the parameter **`HBRCONSOLE=NO`**, so that INS2 does not launch another console. By default, when you set up the HBRSSIDLIST, INS2 is already set up this way. However, after you set up the initial zRule Execution Servers, if you require more servers, follow the steps that are described in this section to add more servers to the same console.

The same HBRPSIST member is used by both zRule Execution Server for z/OS instances so that they connect to the same repository. The same HBRCMMN member can be used by each zRule Execution Server for z/OS instance. The value HBRCONSOLECOMPORT must be the same for both zRule Execution Server for z/OS instances, because both zRule

Execution Server for z/OS instances communicate with the same zRule Execution Server for z/OS console.

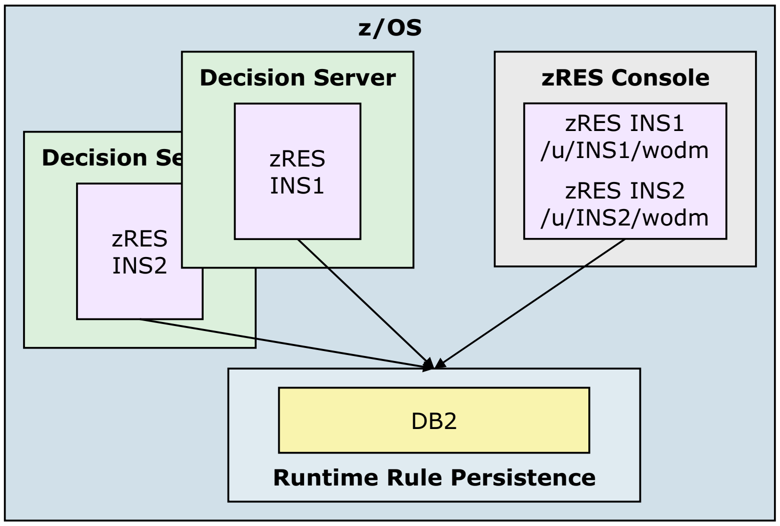


Figure 10-6 One console managing multiple stand-alone zRule Execution Servers for z/OS

10.3.1 Adding a zRule Execution Server for z/OS to a running console

This section describes an Operational Decision Manager topology of adding a zRule Execution Server for z/OS server to a zRule Execution Server for z/OS console that is already running. This topology is created so that you can have multiple zRule Execution Server for z/OS servers running within a single LPAR.

When you set up a new zRule Execution Server for z/OS, the process is similar to the process to set up the first zRule Execution Server for z/OS.This section explains the necessary changes that you make to the HBRINST job. With these changes, you can quickly and easily set up a new zRule Execution Server for z/OS that operates under the same console as an existing zRule Execution Server for z/OS.

Changing HBRINST

When you add a stand-alone zRule Execution Server for z/OS to an existing zRule Execution Server for z/OS console, you make a few changes (see Table 10-6 on page 199):

- Create a new ++HBRSSIDLIST++ to start a new group of zRule Execution Server for z/OS instances.
- Create a ++HBRWORKPATH++ so that the logs, configuration data, res_data, res_xom, and work paths exist for this group. If you are using file-based persistence, share this directory with your previous WORKPATH. Your logs will show the subsystem ID (SSID) on them. However, if you are using file persistence, this work path can be split into separate UNIX System Services paths so that you can keep the tasks separate.

Table 10-6 HBRINST customization values for rules on z/OS

Column one value	Your chosen value	Example value	Reason to update, change, or leave the default
++HBRSSIDLIST++		HBR3,HBR4	Every time that a new zRule Execution Server for z/OS is set up, you must modify this value. Setting a naming convention that can scale with your system is important. This value must be four characters or fewer.
++HBRWORKPATH++		/u/HBR3	This value is the work path for the specific instance of the server. You must update this value for each new zRule Execution Server for z/OS instance that you create. A preferred practice is to create a common directory for the work path directories. Then, create a directory within this common directory for each new instance.

Reusing a copy of the zRule Execution Server for z/OS HBRINST: If you use the preferred practice, you can reuse and update a copy of the existing zRule Execution Server for z/OS HBRINST if you deploy to the same zRule Execution Server for z/OS console. Using this approach, you make only the necessary updates.

Updating HBRUUPTI

Similar to the first configuration, you must update HBRUUPTI so that the new ++HBRWORKDS++ sets are created for the new instance of the stand-alone zRule Execution Server for z/OS. See 10.1.3, "Creating the working datasets using HBRUUPTI" on page 186 for the procedure to update the corresponding lines in the JCL to create the new zRule Execution Server data sets for the new instances. This procedure creates four new PDSs similar to the first instance, but using the ++HBRSSIDLIST++ to build the new data sets.

10.3.2 Creating the working directory

Next, you run the job HBRCRTI in the ++HBRWORKDS++ to create the working directory in the path that was set by ++HBRWORKPATH++. This job creates the UNIX System Services directory for the specific server instance.

10.3.3 DB2 persistence

The DB2 persistence layer that was set with the first group of stand-alone zRule Execution Servers remains the same, so do not repeat this step.

10.3.4 Defining the subsystem for the new instance

The next step for the configuration of a new instance of the stand-alone zRule Execution Server is to define the subsystem in which the new instance runs. The systems programmer must perform this task. Execute the following command, ++HBRSSID++, for each SSID from the ++HBRSSIDLIST++ set in Table 10-6:

```
SETSSI ADD, SUBNAME=++HBRSSID++
```

10.3.5 Modifying and adding the started tasks to the PROCLIB

When you add an instance to an existing console, you create only one started task for this new instance, ++HBRSSID++MSTR. You copy HBRXMSTR as ++HBRSSID++MSTR to `SYS1.PROCLIB` or the equivalent in your environment.

At this time, you must change the data member called HBRMSTR, which is in the ++HBRWORKDS++.SHBRPARM(HBRMSTR). In this data set member, change the **`HBRCONSOLE`** parameter:

```
HBRCONSOLE = NO
```

If you do not change this data member to `NO`, when the instance is started later, it tries to start its own console. If the other instance is already running, this instance fails because the ports are already in use.

Copy ++HBRWORKDS++.SHBRPROC(HBRXMSTR) to `SYS1.PROCLIB` because ++HBRSSID++MSTR. ++HBRSSID++ is the SSID of the new server instance that was created for the specific instance.

SYS1.PROCLIB: `SYS1.PROCLIB` is the default. Change it to match the equivalent library in your environment. ++HBRSSID++ was set in Table 10-6 on page 199.

Authorizing the server instance as a started task

Authorize the ++HBRSSID++MSTR as a started task procedure that executes with the same user ID. Use the following command for each zRule Execution Server instance:

```
RDEFINE STARTED ++HBRSSID++MSTR.* STDATA(USER(<HBRSSID_USER>)
GROUP(<HBRSSID_GROUP>)
```

10.3.6 Security setup for the new stand-alone zRule Execution Server for z/OS

Complete the security setup for the new stand-alone zRule Execution Server for z/OS by using the same method that is described in 10.2.3, “Securing the zRule Execution Server for z/OS for z/OS resources” on page 189. See this section to perform the security setup for the zRule Execution Server instance using the ++HBRSSID++ value that was set in Table 10-6 on page 199.

10.3.7 Starting the new instance

Similar to the first zRule Execution Server for z/OS instance, enter the following command to start the new zRule Execution Server for z/OS instance:

```
START ++HBRSSID++MSTR
```

10.4 Setting up the database connection

Connection to the database can be either a type 4 connection or a type 2 connection. See the DB2 documentation to determine which type is preferable for your environment. For more information, see the *How JDBC applications connect to a data source* topic in the DB2 10 for z/OS Information Center:

http://publib.boulder.ibm.com/infocenter/dzichelp/v2r2/index.jsp?topic=%2Fcom.ibm.db2z10.doc.java%2Fsrc%2Ftpc%2Fimjcc_cjvjdcon.htm

Both the zRule engine and the zRule console make a connection to the database, so both need to be configured to connect to DB2.

The default database connection that is created by zRES is type 4, which is documented in the *Step 5: Configuring a DB2 persistence layer* topic of the IBM Operational Decision Manager Version 8.0.1 Information Center:

http://pic.dhe.ibm.com/infocenter/dmanager/v8r0m1/index.jsp?topic=%2Fcom.ibm.wodm.family.config.zos%2Ftopics%2Ftpc_ds_create_db2_persist.html

This section explains how to set up a type 2 database connection for the console and for zRES.

10.4.1 Setting up a type 2 configuration for the console

You need to include the DB2 library in the `STEPLIB` in the PROC member. Obtain these in ++HBRWORKDS++.++HBRSSID++.SHBRPROC (from where they are copied to the SYSTEM.PROCLIB data set, as described in 10.2.2, "Creating the started tasks (HBRXCNSL and HBRXMSTR)" on page 188).

For HBRXCNSL, and for each HBRXMSTR that accesses the database, add the DB2 libraries to the `STEPLIB`, for example:

```
//STEPLIB  DD DISP=SHR,DSN=&HBRHLQ..SHBRAUTH
//         DD DISP=SHR,DSN=SYS2.DB2.V910.SDSNLOAD
//         DD DISP=SHR,DSN=SYS2.DB2.V910.SDSNLOD2
//         DD DISP=SHR,DSN=SYS2.DB2.V910.SDSNEXIT
```

10.4.2 Updating the database parameters in HBRPSIST

The URL that specifies the location of the database is in ++HBRWORKDS++.++HBRSSID++.SHBRPARM(HBRPSIST). This needs to be updated, and the necessary JARs must be added in an **`HBRDBJARS`** parameter, for example:

```
* URL associated with the database.
HBRDBURL=jdbc:db2:DSN910GP+
              :currentSQLID=ZRES;

* Jars required by the database.
* HBRDBJARS= set to the DB2 jars
HBRDBJARS=/usr/lpp/db2910/classes/db2jcc.jar:+
/usr/lpp/db2910/classes/db2jcc_license_cisuz.jar:+
/usr/lpp/db2910/classes/db2jcc_javax.jar:+
/usr/lpp/db2910/classes/sqlj.zip:
```

10.4.3 Setting up the DB2 identifying file

You need to create an identifying file that indicates the SSID of the DB2 subsystem. This file can then be read by the relevant Java Message Services (JVMs). This file needs to be an EBCDIC file in the UNIX System Services, and it contains the SSID of the database. For example, for the DB2 SSID DHGP, this file contains the following line:

```
db2.jcc.ssid=DHGP
```

10.4.4 Updating the PARM members

You need to update the PARM members that are associated with the CONSOLE and each zRES that accesses the database. The members are HBRCNSL and HBRMSTR, and they are in ++HBRWORKDS++.++HBRSSID++.SHBRPARM. For each of these members, perform the following steps:

1. Include a LIBRARY_SUFFIX in the PARM member that points to the DB2 libraries on UNIX System Services. Add a line that indicates the LIBPATH_SUFFIX, for example:

   ```
   LIBPATH_SUFFIX=/usr/lpp/db2910/lib
   ```

2. Indicate the DB2 subsystem to the JVM by pointing the JVM at the file that is created in 10.4.3, "Setting up the DB2 identifying file" on page 201, using the parameter **`JAVA_OPTIONS`**. For example, if you use the identifying file `DB2_SSID` in the `/u/db2Id directory`, your **`JAVA_OPTIONS`** might read this way:

   ```
   JAVA_OPTIONS=-Ddb2.jcc.propertiesFile=/u/db2Id/DB2_SSID -Xmx256M -Xms256M
   ```

10.4.5 Using your own jobs

If you use your own jobs to configure the databases, ensure that CAPS ALL is turned off.

11

Configuring CICS to work with Operational Decision Manager

This chapter describes Rule Execution Server for z/OS when it is run within the CICS Java Message Service (JVM) server. It considers the configuration of CICS and Rule Execution Server for z/OS required for this environment. It also considers the use of multiple CICS JVM servers, each running an instance of Rule Execution Server for z/OS.

The following topics are covered in this chapter:

© Copyright IBM Corp. 2013. All rights reserved.

11.1 Configuring CICS to invoke a stand-alone Rule Execution Server for z/OS

A CICS region can be configured so that a CICS program can call the Rule Execution Server for z/OS (zRES). Currently, the supported versions of CICS are V3.2, V4.1, V4.2, and V5.1, as shown in Figure 11-1. This section describes the required configuration to enable this feature.

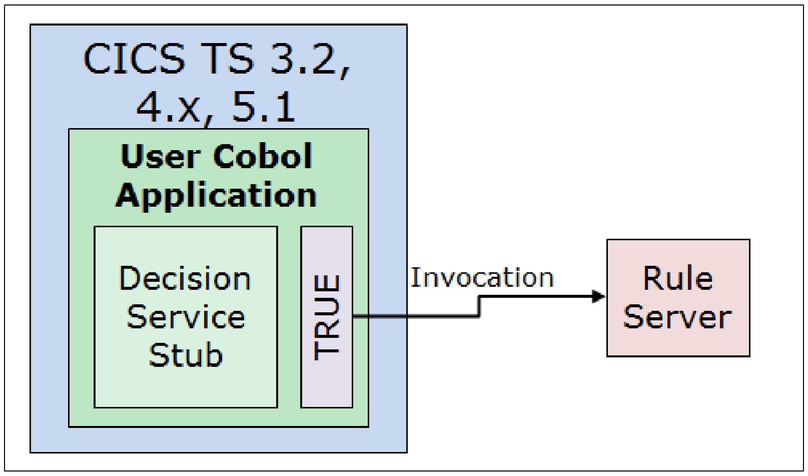


Figure 11-1 CICS COBOL application and stand-alone server

11.1.1 Setting the parameters for CICS

The parameters that are required for zRES to run with CICS are in HBRINST and are described in detail in Appendix B, “Configuring runtime values by using variables defined in HBRINST” on page 327.

The following parameters are required to configure CICS to call rules in zRES:

- `CICSHLQ`
- `CICSCSDDSN`
- `CICSLIST`

Set these parameters in `++HBRHLQ++.SHBRPARM(HBRINST)`.

After you set these parameters, run the following job to apply them to the working data set jobs:

```
++HBRHLQ++.SHBRJCL(HBRUUPTI)
```

For more information about these parameters, see “CICS” on page 329 or the *z/OS configuration and runtime variables* topic in the IBM Operational Decision Manager Version 8.0.1 Information Center:

http://pic.dhe.ibm.com/infocenter/dmanager/v8r0m1/index.jsp?topic=%2Fcom.ibm.wodm.family.config.zos%2Ftopics%2Fcon_ds_jcl_and_runtime_vars.html

Note: Other CICS parameters are only necessary when you set up to run zRES in a CICS JVM server. These parameters are described in 11.2, “Configuring a CICS JVM server to run a Rule Execution Server” on page 206.

11.1.2 Defining the required resources

The resources required for CICS are defined by the JCL job:

```
++HBRWORKDS++.++HBRSSID++.SHBRJCL(HBRCSD)
```

Submit this job to create the resources.

11.1.3 Updating the GRPLIST parameter

After defining the resources, add HBRLIST to the **GRPLIST** parameter in the system initialization table:

```
GRPLIST=(CICSHTAP,HBRLIST)
```

11.1.4 Updating the CICS JCL

Modify the CICS region JCL to include the lines calling to the zRES on the CICS JVM server. You must add the following lines to the components of the CICS JCL.

DFHRPL

In the CICS program library, `DFHRPL`, add the SHBRCICS PDS to the DFHRPL section:

```
//         DD DSN=++HBRHLQ++.SHBRCICS,DISP=SHR
```

Passing the runtime variables to the CICS region

It is necessary to inform the application in which server to run the rules. Add the following runtime variables from the SHBRPARM PDS to the CICS region:

```
//HBRENVPR DD DISP=SHR,DSN=++HBRWORKDS++.++HBRSSID++.SHBRPARM(HBRCICSZ)
//          DD DISP=SHR,DSN=++HBRWORKDS++.++HBRSSID++.SHBRPARM(HBRCMMN)
```

11.1.5 Scenario for installation verification

If you plan to add the installation verification procedure (IVP) to test the zRES on the CICS JVM server, add the following line in the runtime variables section:

```
//SCENARIO DD DISP=SHR,DSN=++HBRWORKDS++.++HBRSSID++.SHBRPARM(HBRSCEN)
```

After the configuration is complete and the region is restarted later, this line enables the CICS MINI transaction (miniloan sample application). The miniloan sample application can be used to verify that the rule engine is connected and working.

If you want to run the miniloan sample application, you must submit the job for deploying the sample rule artifacts. Submit the following job:

```
++HBRWORKDS++.++HBRSSID++.SHBRJCL(HBRDPLOY)
```

This job deploys the rule artifacts to be used by the zRES and can be used directly in the persistence layer. Therefore, it can be used to deploy the sample application to any configuration.

11.1.6 Starting zRES and CICS

Start zRES and CICS now (or restart them if CICS is already running). To start zRES, use the following command:

```
START ++HBRSSID++MSTR
```

11.1.7 Installing HBRGROUP

Install the HBRGROUP resources to CICS by running the following command:

```
CEDA INSTALL GROUP(HBRGROUP)
```

11.1.8 Testing the configuration

The configuration can be tested by using the HBRC transaction. This transaction enables CICS to call zRES. The return code that is shown in Table 11-1 indicates the success of the transaction.

Table 11-1 Return codes

Code	Meaning
GBRZC9000	An error has occurred when executing the HBRC transaction.
GBRZC9001	CICS has connected to zRES.
GBRZC9002	CICS has disconnected from zRES.
GBRZC9003	The CICS region is already connected to zRES.

11.1.9 Automatically connecting CICS to a running zRES instance

This optional step means that it is not necessary to run the HBRC transaction to connect to a running zRES. The zRES must be started before the CICS region. Otherwise, you need to connect the CICS region by manually running HBRC.

There are two ways to automatically connect CICS to the running zRES instance:

- If you do not have a program list table defined, add the following parameter to the CICS system initialization table:

  ```
  PLTPI=HB
  ```

- If you have a program list table defined and specified in your CICS system initialization table, add the HBRCCONN program to the list by using this statement:

  ```
  DFHPLT TYPE=ENTRY,PROGRAM=HBRCCON
  ```

11.2 Configuring a CICS JVM server to run a Rule Execution Server

Another runtime feature of Operational Decision Manager is the addition of a Rule Execution Server for z/OS (zRES) that runs within the CICS JVM server on CICS V4.1, V4.2, and V5.1. The setup is similar for all versions of CICS, although the version of supported Java that is required to operate the CICS JVM servers within the region differs depending on the CICS version. This section describes the setup of this server instance. The configuration that is created for running zRES on a CICS JVM server is shown in Figure 11-2.

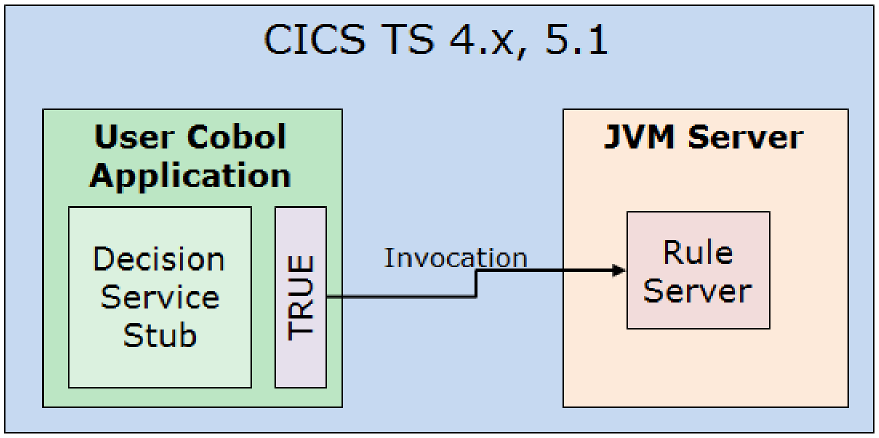


Figure 11-2 CICS COBOL application and CICS JVM server

11.2.1 HBRINST changes

You must update the HBRINST member for the CICS zRES to create a zRES running on a CICS JVM server. The values that you need to update vary on the configuration setup that you want to create. The following sections outline the different versions of CICS and the variables within HBRINST that are important for these versions of CICS.

Building a zRES with CICS JVM server attached to an existing zRES stand-alone instance

If there is an existing zRES stand-alone instance configured and started, the updates in the HBRINST member are limited to the CICS values within the member.

When configuring a CICS JVM server, the values that are listed in Figure 11-3 need to be updated to build the necessary environment for the first CICS region. These parameters are described in detail in Appendix B, "Configuring runtime values by using variables defined in HBRINST" on page 327.

```
CICSWORKPATH
CICSHLQ
CICSCSDDSN
CICSINSTPATH
CICSLIST
HBRJAVA31HOME (CICS V4.1 only)
HBRJAVAHOME (CICS V4.2 and above)
JDBCPLAN
```

Figure 11-3 HBRINST members to consider when you are setting up zRES within a CICS environment

CICS V4.1 requires HBRJAVA31HOME because it only supports Java to 31 bits. It is used to help build a zRES on a CICS JVM that operates with a 31-bit JVM server. CICS V4.2 and V5.1 use HBRJAVAHOME because they support Java 64.

Tip: CICSWORKPATH and HBRWORKPATH (from the zRES stand-alone configuration) must differ for the configuration to succeed. A preferred practice is to append /CICS to the HBRWORKPATH to build the new CICSWORKPATH. So if the HBRWORKPATH is /u/HBR1, the CICSWORKPATH is /u/HBR1/CICS. This practice prevents the use of any common configuration files from the two setups.

The following jobs are updated:

- HBRCSD, HBRCSD41, and HBRCJS41 for CICS V4.1
- HBRCSD, HBRCSDJ, and HBRCJVMP for CICS V4.2 and V5.1

The following Java levels are supported:

- The CICS 4.1 environment supports Java 6.0.1-32 bit only.
- The CICS 4.2 environment supports Java 6.0.1-64 bit only.
- The CICS 5.1 environment supports Java 7.0-64 bit only.

If your zRES uses a different level of Java from the level that is required by the CICS environment, ensure that this is changed in the HBRINST member to account for the required level of Java. Otherwise, the CICS region JVM in which the zRES is installed does not start.

Producing the working datasets

After you complete the update of the HBRINST member, the HBRUUPTI job needs to be submitted to incorporate the new values. The following datasets are updated:

```
++HBRWORKDS++.++HBRSSID++.SHBRPARM
++HBRWORKDS++.++HBRSSID++.SHBRJCL
```

If these datasets exist from a previous installation, you can replace the existing instances with the new ones by deleting and rerunning HBRUUPTI. This gives the members the values from both the previous installation and the newly updated CICS pieces.

The other option is to manually update the necessary members. The members that require updating for CICS are listed in Table 11-2. These members are directly affected by CICS. Members that are not directly affected by CICS modifications, such as for DB2 changes, are not included.

Table 11-2 Members requiring modification if manual update is used

	++HBRWORKDS++. ++HBRSSID++.SHBRJCL	**++HBRWORKDS++. ++HBRSSID++.SHBRPARM**
CICS V4.1	HBRCRTCI HBRCSD HBRCSD41 HBRCJS41 HBRCWOLA	HBRCICSD HBRCICSJ
CICS V4.2 CICS V5.1	HBRCRTCI HBRCSD HBRCSDJ HBRCJVMP HBRCWOLA	HBRCICSD HBRCICSJ

For the original versions of the members that must be updated, look at the original target libraries of the members that are in ++HBRHLQ++.SHBRJCL and ++HBRHLQ++.SHBRPARM.

11.2.2 Creating the working directories

For CICS zRES for z/OS, you need to create a workpath that is dedicated to the CICS system. This workpath cannot be the same workpath as the HBRWORKPATH.

Submit the job HBRCRCTI. This job creates the CICS working path directories within UNIX System Services with the required configuration pieces, as well as a separate directory for the CICS JVM server logs.

If there is not an existing zRES stand-alone server, see 11.3, “Configuring a zRES dedicated to a CICS region with HBRMODE set to NORULE” on page 212 for instructions to set up a dedicated console to CICS regions with no other execution.

11.2.3 Creating the JVM profile

Create the CICS JVM server profile for the correct version of CICS. This step depends on the version of CICS JVM server, either 31 bit on CICS V4.1 or 64 bit on CICS V4.2 and CICS V5.1.

To create the correct profile, submit one of these jobs:

- CICS V4.2/5.1: Run job HBRCJVMP to create the JVM profile for a CICS V4.2/5.1 region.
- CICS V4.1: Run job HBRCJS41 to create the JVM profile for a CICS V4.1 region.

This job creates the profile within the `++HBRCICSWORKPATH++` directory, which is created in 11.2.2, “Creating the working directories” on page 209.

Important: Within CICS, ensure that the level of Java that is used is supported by the CICS version that you are running. For CICS v4.1 and v4.2, this level is Java V6.0.1. For CICS v5.2, this level is Java V7.0.

11.2.4 Defining the CICS resources

Next, you define the CICS resources that are required by the server. Submit the two jobs that relate to the resources, which are both in the following data set:

```
++HBRWORKDS++.++HBRSSID++.SHBRJCL
```

The first job is the same job for both CICS V4.1, CICS V4.2, and CICS V5.1:

- HBRCSD: This job defines the resources that are required by CICS V4.1, V4.2, and V5.1.

The next step depends on the CICS version:

- HBRCSDJ: This job defines the necessary resources for a CICS V4.2 region or CICS V5.1 region.
- HBRCSD41: This job defines the necessary resources for a CICS V4.1 region.

11.2.5 Adding HBRLIST to the system initialization table

After you define the resources, add HBRLIST to the system initialization table that is specified by the **GRPLIST** parameter:

```
GRPLIST=(CICSHTAP,HBRLIST)
```

11.2.6 Setting the JVMPROFILEDIR

If you do not already have a defined JVM profile directory, you need to set the default JVM profile to point at the working directory that you created for CICS in “Building a zRES with CICS JVM server attached to an existing zRES stand-alone instance” on page 207.

In the CICS system initialization table, create the variable *JVMPROFILEDIR*. It needs to point to the CICS working directory:

```
JVMPROFILEDIR=++CICSWORKPATH++
```

If you use a JVMPROFILEDIR that cannot correspond to the ++CICSWORKPATH++, you must copy the profile that is created 11.2.3, “Creating the JVM profile” on page 209 to the correct JVMPROFILEDIR directory.

11.2.7 Changing the CICS region JCL

The CICS region JCL must be modified to include the lines that call the zRES on the CICS JVM server.

DFHRPL

In the CICS program library, DFHRPL, add the SHBRCICS PDS to the DFHRPL section:

```
//         DD DSN=++HBRHLQ++.SHBRCICS,DISP=SHR
```

Passing the runtime variables to the CICS region

Add the following runtime variables from the ++HBRWORKDS++.++HBRSSID++.SHBRPARM PDS to the CICS region. The following data definition (DD) statements are required:

```
//HBRENVPR DD DISP=SHR,DSN=++HBRWORKDS++.++HBRSSID++.SHBRPARM(HBRCICSJ)
//          DD DISP=SHR,DSN=++HBRWORKDS++.++HBRSSID++.SHBRPARM(HBRCMMN)
//          DD DISP=SHR,DSN=++HBRWORKDS++.++HBRSSID++.SHBRPARM(HBRCICSD)
```

The HBRCICSD member in ++HBRWORKDS++.++HBRSSID++.SHBRPARM sets up the database environment.

STEPLIB

Add the ++HBRHLQ++.SHBRAUTH library to the STEPLIB concatenation. This has the requisite APF Authorization members required for the CICS region, for example:

```
// DD DSN=++HBRHLQ++.SHBRAUTH,DISP=SHR
```

The STEPLIB of the region must also contain the members SDSNLOAD and SDSNLOD2.

11.2.8 Scenario for installation verification

If you plan to add the installation verification procedure (IVP) to test the zRES on the CICS JVM server, add the following line in the runtime variables section:

```
//SCENARIO DD DISP=SHR,DSN=++HBRWORKDS++.++HBRSSID++.SHBRPARM(HBRSCEN)
```

After the configuration is complete and the region is restarted later, this line enables the CICS MINI transaction (miniloan sample application). The miniloan sample application can be used to verify that the rule engine is connected and works.

Before you run the IVP, you must submit the job for deploying the rule artifacts by submitting the following job:

```
++HBRWORKDS++.++HBRSSID++.SHBRJCL(HBRDPLOY)
```

This job deploys the rule artifacts directly to the persistence layer. Therefore, it can be used to deploy the sample application to any configuration.

11.2.9 Security for the zRES on CICS JVM server

For the security setup for the zRES on the CICS JVM server, perform the steps in 10.2.3, "Securing the zRule Execution Server for z/OS for z/OS resources" on page 189. You perform the same steps for all zRES servers. The CICS region's user for the started task must be granted access to the Connect security so that it can connect to the zRES instance for rule execution.

11.2.10 CEDA installation of HBRGROUP resources

After you start the CICS region, you must install the resources that were defined earlier. Run the following command in CICS:

```
CEDA INSTALL GROUP(HBRGROUP)
```

11.2.11 Database connect for the CICS region

If you use the database configuration, run the following command to connect the database to the CICS region of the zRES:

```
CEMT INQUIRE DB2CONN
```

Then, change the CONNECTST property from `Notconnected` to `Connected`.

Tip: You can also use the **`DB2START SIT`** parameter to perform this connection.

11.2.12 Connecting the zRES to the CICS JVM server

After the started task is up and the resources are connected, connect the zRES to the CICS JVM server by using the CICS transaction HBRC. This task sets up storage in the CICS JVM for connection to zRES and checks and initializes the JVM. If successful, this transaction returns the following message:

```
GBRZC9001I RC=0000
```

If the connection is unsuccessful, it returns `GBRZC9001E RC=XXXX`, where *XXXX* is the return code message. If it is a `3006` message, you did not start the ++HBRSSID++MSTR task yet. If it is a `3014` message, you did not enable the CICS JVM server yet.

For more information about these return codes, see the *Completion codes* topic in the Decision Server for z/OS V8.0.1 product documentation:

http://pic.dhe.ibm.com/infocenter/dmanager/v8r0m1/index.jsp?topic=%2Fcom.ibm.wodm.zos.dserver.ref%2Fhtml%2Freasoncodes%2Fhtml%2Fcodes_zres.html

11.2.13 Deploying the installation verification program

If you set up the CICS region to have HBRSCEN, perform the following steps to run the IVP:

1. Deploy the RuleApp for database persistence by running the job HBRDPLOY.
2. After you deploy the RuleApp, go back to the CICS region and run the CICS transaction MINI. Your region then displays the output that is shown in Example 11-1.

Example 11-1 MINI output

```
MINICICS--msg-The yearly income is lower than the basic request
MINICICS --Loan customer 0000000006
MINICICS --about to call # Execution Server
MINICICS -- Rule executed in-JVMS
MINICICS--name-Michelle            loan amount-0001000100-approved-F
MINICICS--msg-The loan cannot exceed 1000000
MINICICS --Disconnect from zRule Execution Server
MINICICS --SUCCESSFUL COMPLETION of demo
MINICICS--name-John                loan amount-0000250000-approved-F
MINICICS--msg-The age exceeds the maximum.
MINICICS --Loan customer 0000000003
MINICICS --about to call zRule Execution Server
MINICICS -- Rule executed in-JVMS
MINICICS--name-Sarah               loan amount-0000500000-approved-F
MINICICS--msg-Credit score below 200
MINICICS --Loan customer 0000000004
MINICICS --about to call zRule Execution Server
MINICICS -- Rule executed in-JVMS
MINICICS--name-Andy                loan amount-0000500000-approved-F
MINICICS--msg-Too big Debt-To-Income ratio
MINICICS --Loan customer 00000000 5
MINICICS --about to call zRule Execution Server
MINICICS -- Rule executed in-JVMS
MINICICS--name-David               loan
amount-0000250000-approved-F
```

The setup of the zRES on the CICS JVM server is complete.

11.3 Configuring a zRES dedicated to a CICS region with HBRMODE set to NORULE

There are instances where batch and CICS environments are not allowed to coexist due to rules within an enterprise. In this case, you need to change the configuration to run a console, which is similar to setting up an address space and database in the configuration in Chapter 10, “zRule Execution Server for z/OS stand-alone server” on page 183.

This section describes how to set up a console address space that is dedicated to a CICS region on a logical partition (LPAR), but without a ZRES engine and, therefore, not allowing any batch rule executions.

11.3.1 Why a NORULE zRES address space is needed

There are cases where it is not appropriate to have a batch environment running along with a CICS environment, therefore disallowing the standard method of running a console address space. This situation might occur because the enterprise wants to maintain a difference between the batch and online systems, the rules are different, the organizations that maintain them are different, and so on.

In these situations, it might be necessary to create a dedicated zRES address space on an LPAR that handles the runtime repository only for CICS. This configuration is explained in this section. The primary purpose is to start the console without the zRES engine, therefore reducing the memory footprint and not creating the zRES address space.

11.3.2 Configuration parameter

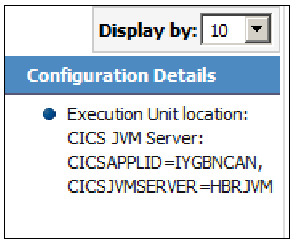
NORULE is one of the settings for the **`++HBRMODE++`** parameter, the details of which are in Table B-1 on page 328.

The parameter is present in the SHBRPARM(HBRINST) and SHBRPARM(HBRMSTR) members. Setting this in HBRINST before running HBRUUPTI to create the HBRWORKDS data set sets it as the default in HBRMSTR. However, it can be set at a later stage in HBRMSTR.

After this variable is set, the MSTR is started with no rules, which the job log indicates with the following message:

```
HBRMODE is NORULE. The server does not execute rules locally.
```

When HBRC is next run from CICS (which might be on the initialization of the JVMs), it connects to the console. When you start the console, look on the Server Info tab. On the far right of the Execution Units grid, you see an indication that CICS JVM is being used, as shown in Figure 11-4.



- Execution Unit location: CICS JVM Server: CICSAPPLID=IYGBNCAN, CICSJVMSERVER=HBRJVM

Figure 11-4 Console showing that the rules run under CICS JVM

11.4 Working with an IBM CICSPlex

Rule execution can be run in multiple CICS servers at a time, all controlled by the same console and running from the same DB2 database. This allows multiple CICS systems within a CICSPlex to access zRES, as shown in Figure 11-5.

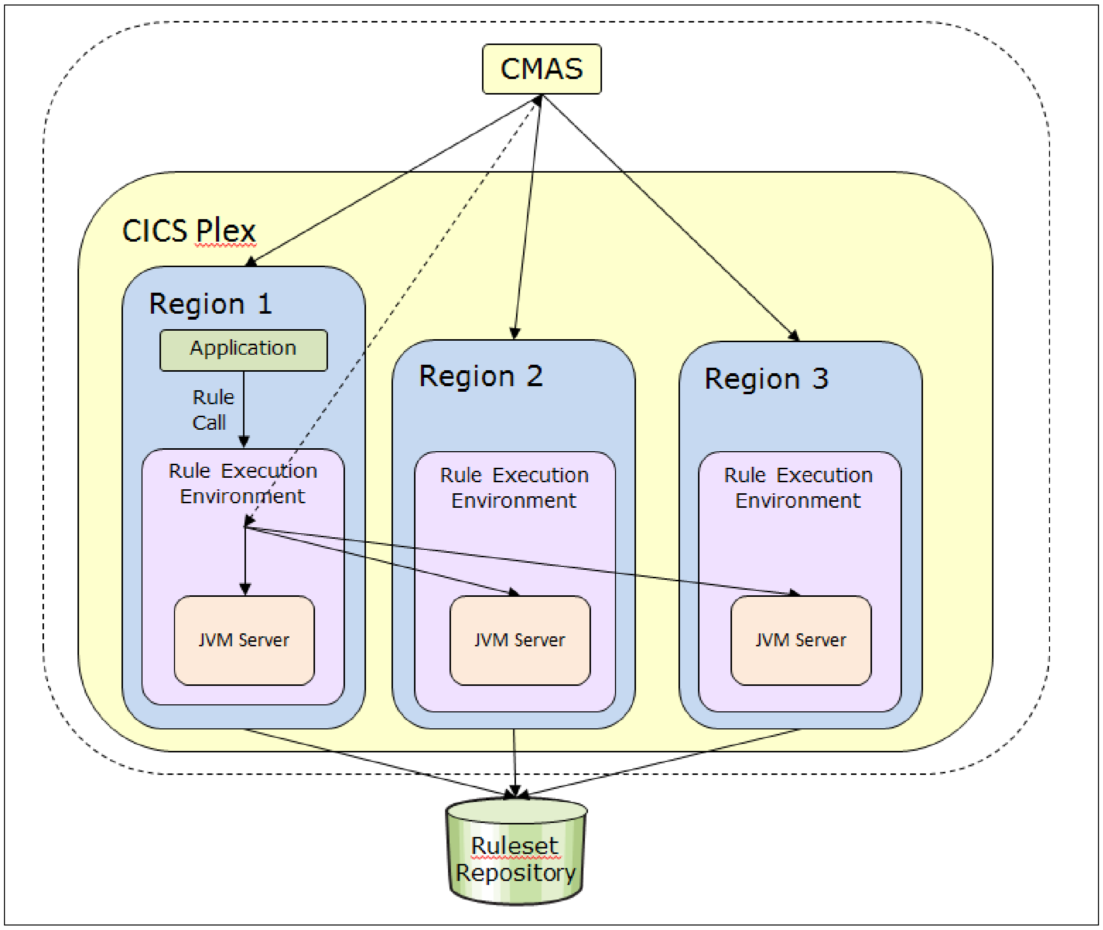


Figure 11-5 Multiple CICS JVM servers

The CICS application in a region (for example, Region 1 in Figure 11-5) makes a rule call into the Rule Execution Environment. The environment then uses the CICSPlex System Manager address space (CMAS) that manages this CICSPlex to route the call through to run on the most appropriate CICS JVM server to run the application, based on the current workload constraints. Although the rule might be run on any of the CICS JVM servers, this selection is invisible to the calling application.

11.4.1 Using a CICSPlex for zRES

The benefits of using IBM CICSPlex are described in the *Introduction to CICSPlex SM* topic in the CICS Transaction Server for z/OS, Version 4 Release 2 Information Center:

http://pic.dhe.ibm.com/infocenter/cicsts/v4r2/index.jsp?topic=%2Fcom.ibm.cics.ts.cpsmconcepts.doc%2Ftopics%2Feyue3am.html

For zRES, using a CICSPlex in the layout displayed in Figure 11-5 on page 214 allows work to be routed to the most appropriate CICS JVM server for workload managing that matches the workload management goals. Using a CICSPlex helps to ensure the optimum performance for zRES.

11.4.2 Configuring the use of a CICSPlex with zRES

To set up multiple CICS JVM servers, first set up a single connection to the calling CICS JVM server by using the method that is described in 11.2, “Configuring a CICS JVM server to run a Rule Execution Server” on page 206. You can also see the *Configuring zRule Execution Server for z/OS to execute rules on a CICS JVM server* topic in the IBM Operational Decision Manager Version 8.0.1 Information Center:

http://pic.dhe.ibm.com/infocenter/dmanager/v8r0m1/index.jsp?topic=%2Fcom.ibm.wodm.family.config.zos%2Ftopics%2Ftsk_ds_config_cics_jvm.html

The working directory that is created as part of this process can be used by all servers in the CICSPlex. All servers can also use the default CICS JVM profile in that working directory.

Note: This section assumes that you use the HBRCSD job. Depending on your configuration, it might be appropriate to use HBRCSD41 or HBRCSDJ instead.

Changing the HBRCSD job

Each of the CICS regions needs to be set up using the HBRCSD job. At the moment, this job is pointing to the calling CICS JVM server.

In the job ++HBRWORKDS++.++HBRSSID++.SHBRPARM(SHBRJCL), the following two lines need to be changed:

```
//STEPLIB  DD DISP=SHR,DSN=<CICS INSTALL>.SDFHLOAD
//DFHCSD   DD DISP=SHR,DSN=<CICS REGION>.DFHCSD
```

Change these lines so that they are appropriate for the first CICS region to be included. For example, they now look like the following lines:

```
//STEPLIB  DD DISP=SHR,DSN=CTS510.CICS680.SDFHLOAD
//DFHCSD   DD DISP=SHR,DSN=CTS510GP.IYGDNCAZ.DFHCSD
```

Rerun the job for this new CICS region.

Repeat for each CICS region.

Note: You might want to save the original HBRCSD for future use or reference. You might want to make copies of this job for each of the regions, for example, HBRCSD01, HBRCSD02, and so on.

Completing the configuration for each region

For each region, run the following steps:

1. Update the CICS region startup JCL by using the procedure that is defined in 11.2.7, “Changing the CICS region JCL” on page 210 for each region.
2. Configure security, as described in 11.2.9, “Security for the zRES on CICS JVM server” on page 211.
3. Start the zRES on the CICS JVM server by using the procedure that is described in 11.2.10, “CEDA installation of HBRGROUP resources” on page 211.

4. Perform the CEDA installation of HBRGROUP by using the procedure that is described in 11.2.10, “CEDA installation of HBRGROUP resources” on page 211.
5. If you use a database, connect the database by using the procedure that is described in 11.2.11, “Database connect for the CICS region” on page 211.
6. Connect to the zRES by using the procedure that is described in 11.2.12, “Connecting the zRES to the CICS JVM server” on page 211.
7. If required, you can test the CICS JVM server by using the procedure that is described in 11.2.13, “Deploying the installation verification program” on page 212.

11.5 Working with multiple CICS JVM servers

Rules execution can be run through multiple separate CICS systems on the same LPAR, with the execution units (XU) being deployed to each CICS system. This topology is displayed in Figure 11-6.

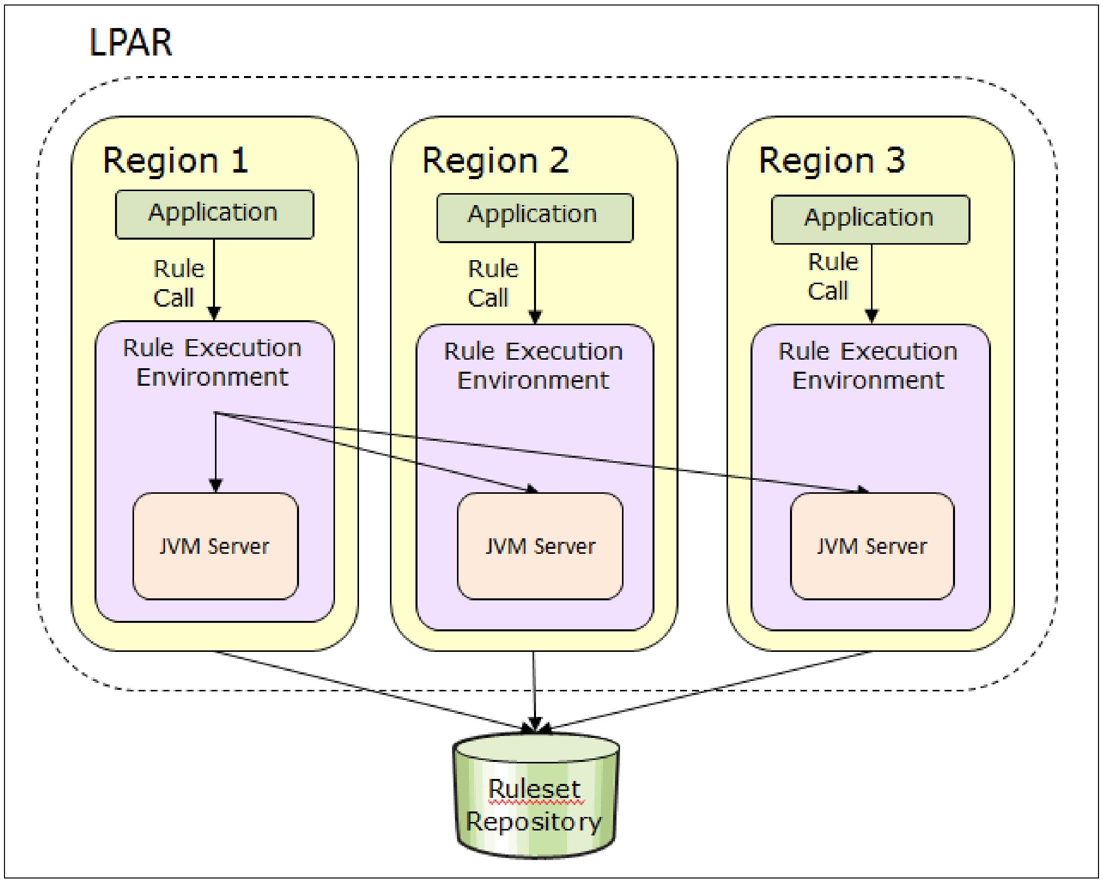


Figure 11-6 Multiple separate CICS JVM servers

Using this method allows different CICS systems to simultaneously access the same set of rules. They might be running different applications or have different uses that require them to be separate from each other.

To set up multiple CICS JVM servers, first set up a single server by using the method that is described in 11.2, “Configuring a CICS JVM server to run a Rule Execution Server” on page 206.

Additional information is in the *Configuring zRule Execution Server for z/OS to execute rules on a CICS JVM server* topic of the IBM Operational Decision Manager Version 8.0.1Information Center:

http://pic.dhe.ibm.com/infocenter/dmanager/v8r0m1/index.jsp?topic=%2Fcom.ibm.wodm.family.config.zos%2Ftopics%2Ftsk_ds_config_cics_jvm.html

The working directory that is created as part of this process can be used by all servers in the CICSPlex. All servers can also use the default CICS JVM profile in that working directory.

Note: This section assumes that you use the HBRCSD job. Depending on your configuration, it might be appropriate to use HBRCSD41 or HBRCSDJ instead.

Changing the jobs

If you want to use the same working directory for all the CICS servers that use the same zRES, you need to change the HBRCSD job for each server. Follow the procedure that is described in "Changing the HBRCSD job" on page 215. Then, correct the CICS region startup JCL, as described in 11.2.7, "Changing the CICS region JCL" on page 210.

If you want to use a different JVM profile for each server, you also need to create new working directories and JVM profiles for each of the servers. Each of the CICS regions needs modifications to the jobs in HBRWORKDS, and then those jobs need to be run again.

TIP: Although the following steps indicate that the jobs must be modified, you might want to make copies of the jobs, therefore retaining the original jobs for future reference and use.

For each of the new regions, you need to perform the following actions.

Creating the new working directories

If you want to use different directories, you can use the job HBRCRTCI. You need to modify the name of the working directory so that it points to the directory that is required for the new CICS region.

Modify the SYSTSIN of the job to indicate the new area to use. For example, see the code extract that is shown in Figure 11-7.

```
//SYSTSIN  DD  *
  BPXBATCH SH . +
  /usr/lpp/zDM/V8R0M1/zexecutionserver+
  /bin/hbrcrtin.sh +
  /usr/lpp/zDM/V8R0M1 +
  /u/HBR1/HBR1CICS42A
/*
```

Figure 11-7 HBRCRTCI prior to code change

This code then becomes the code extract that is shown in Figure 11-8.

```
//SYSTSIN  DD  *
  BPXBATCH SH . +
  /usr/lpp/zDM/V8R0M1/zexecutionserver+
  /bin/hbrcrtin.sh +
  /usr/lpp/zDM/V8R0M1 +
  /u/HBR1/HBR1CICS42B
/*
```

Figure 11-8 HBRCRTCI after the code change

After you modify the code, submit the job to create the new directory.

Tip: If you extend the length of the directory to be created, ensure that you do not extend it beyond the allowable width for the JCL format.

Creating the new profile

Now that the new directory is created, you can optionally create a new JVM profile within this directory:

- For CICS V4.2 and later, use the job HBRCJVMP.
- For CICS V4.1, use the job HBRCJS41.

The following instructions assume that the job HBRCJVMP is used.

Modify the job HBRCJVMP to indicate the working directory for the new CICS region. This directory is within the SYSTSIN area of the JCL. For example, see the code extract that is shown in Figure 11-9.

```
//SYSTSIN  DD  *
  BPXBATCH SH . +
  /usr/lpp/zDM/V8R0M1/zexecutionserver+
  /bin/hbrcjvmp.sh +
  /usr/lpp/zDM/V8R0M1 +
  /u/HBR1/HBR1CICS42A +
  /java/java601_bit64_GA/J6.0.1_64 +
  /usr/lpp/db2910/classes +
  /usr/lpp/db2910/lib
 /*
```

Figure 11-9 HBRCJVMP prior to code change

This code becomes the code extract that is shown in Figure 11-10.

```
//SYSTSIN  DD  *
  BPXBATCH SH . +
  /usr/lpp/zDM/V8R0M1/zexecutionserver+
  /bin/hbrcjvmp.sh +
  /usr/lpp/zDM/V8R0M1 +
  /u/HBR1/HBR1CICS42B +
  /java/java601_bit64_GA/J6.0.1_64 +
  /usr/lpp/db2910/classes +
  /usr/lpp/db2910/lib
 /*
```

Figure 11-10 HBRCJVMP after code change

Updating HBRCSD

The new CICS region needs to be set up by using the HBRCSD job.

In the job ++HBRWORKDS++.++HBRSSID++.SHBRPARM(SHBRJCL), you need to change the following two lines:

```
//STEPLIB  DD DISP=SHR,DSN=<CICS INSTALL>.SDFHLOAD
//DFHCSD   DD DISP=SHR,DSN=<CICS REGION>.DFHCSD
```

Change these lines so that they are appropriate for the new CICS region to be included, for example:

```
//STEPLIB  DD DISP=SHR,DSN=CTS510.CICS680.SDFHLOAD
//DFHCSD   DD DISP=SHR,DSN=CTS510GP.IYGDNCAZ.DFHCSD
```

Rerun the job for this new CICS region.

Completing the configuration for each region

To complete the configuration for each region, follow the instructions that are described in “Completing the configuration for each region” on page 215.

12

Configuring IMS to work with Operational Decision Manager

This chapter describes the use of Operational Decision Manager on z/OS from IMS.

This chapter contains the following topics:

© Copyright IBM Corp. 2013. All rights reserved.

12.1 IMS and Operational Decision Manager

Using Operational Decision Manager from IMS allows the rules that influence the business logic to be kept outside the IMS environment. Encapsulating the rules from these decision points outside of the IMS application decouples the rules from the IMS applications. This allows the business rules to be more reactive to changes without having to modify the IMS applications. IMS can call Operational Decision Manager from programs running in the Message Processing Regions (MPR), Batch Message Processing (BMP), or Data Language/Interface (DL/I) programs. IMS and the Rule Execution Server for z/OS (zRES) must reside on the same logical partition (LPAR).

IMS uses the same API calls that are used by batch and CICS programs:

- HBRCONN: To connect to the server group
- HBRRULE: To run rules
- HBRDISC: to disconnect from the server group

However, these API calls use IMS-dedicated stubs that are contained in a separate library, as shown in Figure 12-1.

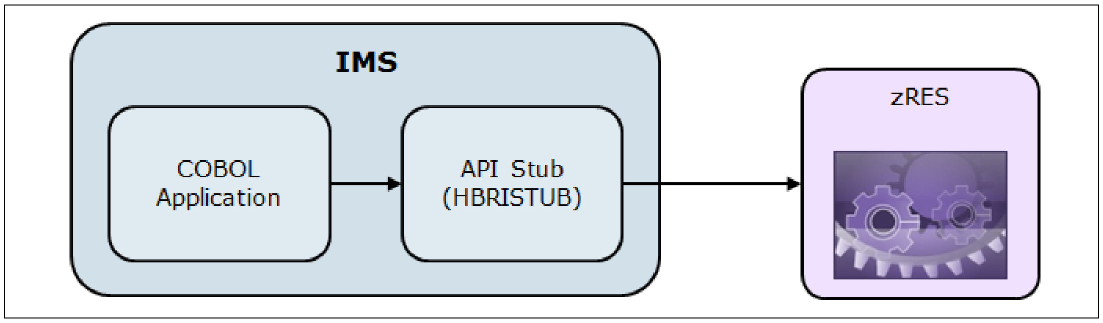


Figure 12-1 IMS calling into zRES

12.2 Configuration

This section describes the required configuration for the different types of programs within IMS.

For all types of programs, to resolve the API calls of HBRCONN, HBRRULE, and HBRDISC, the IMS program needs to be link-edited with the HBRISTUB module. Include the following link-edit step when binding the client program:

```
INCLUDE HBRLIB (HBRISTUB)
```

There are three configuration parameters that relate to IMS:

- IMSHLQ: The high-level qualifier (HLQ) of the IMS installation
- IMSREGID: The region ID of the IMS region to be used
- IMSREGHLQ: The HLQ of the IMS region to be used

These three configuration parameters are used in the creation of the provided sample programs (HBRMINI and HBRMINIT). Although it is useful to set them up, it is not necessary. zRES can be used without them being set up, provided that the other configuration steps are followed.

12.2.1 BMP and DLI

There is no additional setup required for BMP and DL/I programs to call into the zRES.

12.2.2 Message Processing Region (MPR)

Message Processing Region (MPR) programs require additional setup to access zRES. In the message processing JCL, you need to add the following information:

1. An HBRENVPR statement that gives the location of the data set member that contains the details about the location of the rules environment:

   ```
   //HBRENVPR DD DISP=SHR,
   // DSN=WODM.V80.CUSTMISE.HBR1.SHBRPARM(HBRBATCH)
   ```

2. Include the zRES load library in the STEPLIB and DFSESL statements:

   ```
   // DD DSN=WODM.V80.HBR1.BASE.SHBRLOAD,DISP=SHR
   ```

3. Restart your IMS Message Processing Region so that it uses the changes that you made to the JCL.

12.3 IMS and Rules Execution Server on WebSphere Application Server for z/OS

Further configuration steps are necessary to run rules on a Rules Execution Server on WebSphere Application Server for z/OS. These steps are described in 13.5, “IMS and Rule Execution Server using WOLA on z/OS” on page 239.

13

Configuring WebSphere Optimized Local Adapters support

WebSphere Optimized Local Adapters (WOLA) is a feature of WebSphere Application Server for z/OS that manages communication between WebSphere Application Server and an external address space, such as CICS, batch, or IMS, that resides in the same logical partition (LPAR).

For more information about WOLA, see the *Configuring WebSphere Optimized Local Adapters (WOLA)* topic in the IBM Operational Decision Manager Version 8.0.1 Information Center:

`http://pic.dhe.ibm.com/infocenter/dmanager/v8r0m1/topic/com.ibm.wodm.family.config.zos/topics/con_ds_install_config_wola.html`

The following topics are covered in this chapter:

13.1 Overview of WebSphere Operational Local Adapters

WebSphere Operational Local Adapters (WOLA) is a component of WebSphere Application Server for z/OS. It uses cross-memory mechanisms to provide a bidirectional, high volume exchange of messages between WebSphere Application Server for z/OS and the calling application.

When Operational Decision Manager is installed on WebSphere Application Server for z/OS, WOLA can be used as a way for COBOL applications to execute rules using the Rule Execution server (RES) within Operational Decision Manager on WebSphere Application Server for z/OS rather than running the z/OS native Rule Execution Server. COBOL applications do not need any changes to connect to Operational Decision Manager using WOLA because the redirection is achieved by runtime JCL variables. This allows Operational Decision Manager to benefit from this method of high volume message exchange.

13.1.1 Configuring WOLA

When calling to RES using WOLA, it is necessary for WOLA to know the correct WebSphere Application Server with which to connect. There might be more than one WebSphere Application Server running on the same system and you need to connect to the one running RES. This information is included in the required JCL variables.

13.1.2 JCL variables for using WOLA

Use the following JCL variables for WOLA:

- The first variable indicates that the target RES is accessed using WOLA. Set *++HBRTARGETRES++* to `WOLA`.
- RES also needs to know where the WOLA load library can be located. This is the load library that was created as part of the WOLA setup. The *++HBRWOLALOADLIB++* indicates the location of your WOLA load library. This load library is created as part of the configuration process that is described in step 1 on page 227.
- Indicate the details of the WebSphere Application Server to use by using the Cell, Node, and Server name, which lead to a unique WebSphere Application Server. The following variables give a unique identifier to the correct WebSphere Application Server:
  - *++HBRWOLACELL++*
  - *++HBRWOLANODE++*
  - *++HBRWOLASERVER++*

For more information about the WOLA-related variables that can be set in the HBRINST member, see “WebSphere Optimized Local Adapters (WOLA) script parameters” on page 333.

13.2 Sample configuration of WebSphere Application Server to use WOLA

The exact configuration of WOLA depends on the version of WebSphere Application Server for z/OS that is used.

For complete details, see the *Configuring WebSphere Optimized Local Adapters (WOLA)* topic in the IBM Operational Decision Manager Version 8.0.1 Information Center:

http://pic.dhe.ibm.com/infocenter/dmanager/v8r0m1/topic/com.ibm.wodm.family.config.zos/topics/tsk_ds_install_wola_was8.html

This section provides an example of a configuration that only uses the following steps to configure WOLA:

1. Create a load library that contains the modules that are required by WOLA from within the WebSphere Application Server installation directory in UNIX System Services. In this example, it is assumed the WebSphere Application Server is installed in the `/was/V8.0` directory. The library is created at the selected location. This example used `WODM.OLA.LOADLIB`:

   ```
   /was/v8.0/profiles/default/bin/copyZOS.sh OLAMODS WODM.OLA.LOADLIB
   ```

2. On the WebSphere Application Server console, set the environment variables that are required for WOLA. For more information about these variables for WebSphere Application Server, see the *Optimized local adapters environment variables* topic in the WebSphere Application Server, Network Deployment, Version 8.0 Information Center:

   http://pic.dhe.ibm.com/infocenter/wasinfo/v8r0/index.jsp?topic=%2Fcom.ibm.websphere.nd.multiplatform.doc%2Finfo%2Fae%2Fae%2Fcdat_olacustprop.html

   It is possible to set the environment variables automatically by using the script **`olaRar.py`** that is described in the *olaRar.py script file* topic in the WebSphere Application Server, Network Deployment (z/OS), Version 8.0 Information Center:

   http://pic.dhe.ibm.com/infocenter/wasinfo/v8r0/index.jsp?topic=%2Fcom.ibm.websphere.zseries.doc%2Finfo%2Fzseries%2Fae%2Fcdat_wsadminola.html

   However, if you want to set the environment variables manually, complete the following steps:

   a. Expand **Environment** and select **WebSphere variables**, as shown in Figure 13-1.

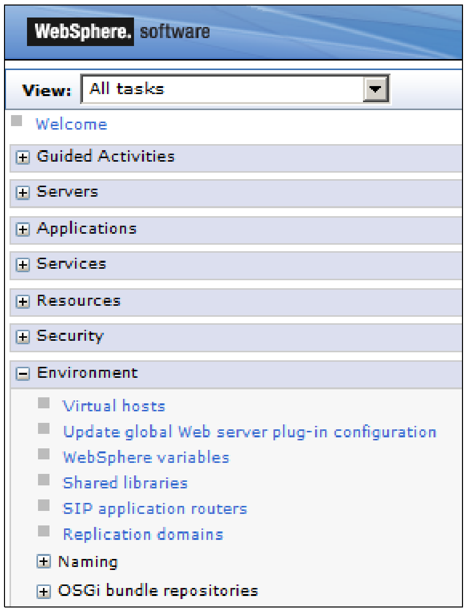
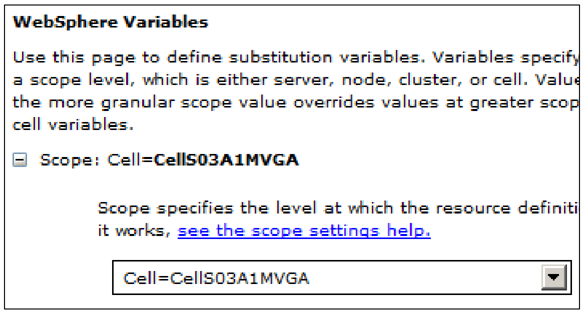


Figure 13-1 WebSphere Application Server console menu

b. Select the **Scope** cell in the drop-down menu, as shown in Figure 13-2.



WebSphere Variables

Use this page to define substitution variables. Variables specify
a scope level, which is either server, node, cluster, or cell. Value
the more granular scope value overrides values at greater scop
cell variables.

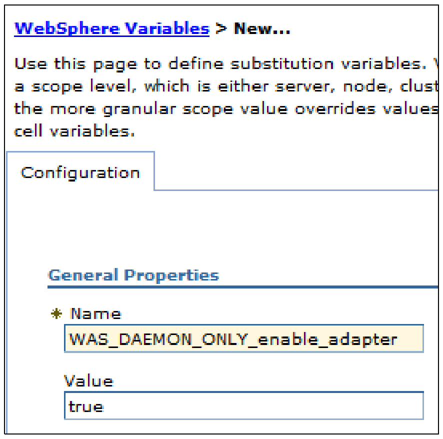
⊟ Scope: Cell=**CellS03A1MVGA**

Scope specifies the level at which the resource definiti
it works, see the scope settings help.

Cell=CellS03A1MVGA

Figure 13-2 Selecting the scope

c. Add the new variable `WAS_DAEMON_ONLY_enable_adapter` and set it to `true`, as shown in Figure 13-3.



WebSphere Variables > New...

Use this page to define substitution variables. V
a scope level, which is either server, node, clust
the more granular scope value overrides values
cell variables.

Configuration

General Properties

* Name
WAS_DAEMON_ONLY_enable_adapter

Value
true

Figure 13-3 Setting WAS_DAEMON_ONLY_enable_adapter

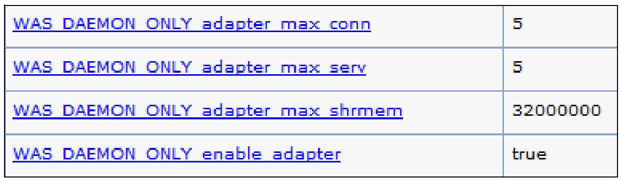
d. There are three other variables that can be added (or updated if already present) to the appropriate values for your system. The values can be used when setting up a sample server to ensure that WOLA is working. (You need different values for your development, test, and production systems.) These values (shown in Figure 13-4) are for tuning purposes and are not actually required for WOLA to work in the first instance:

- *WAS_DAEMON_ONLY_adapter_max_con*n

  The maximum number of connections needed for this WOLA
- *WAS_DAEMON_ONLY_adapter_max_serv*

  The maximum number of outbound services needed for this WOLA
- *WAS_DAEMON_ONLY_adapter_max_shrmem*

  The maximum amount of shared memory allocated to the WOLA



WAS_DAEMON_ONLY_adapter_max_conn	5
WAS_DAEMON_ONLY_adapter_max_serv	5
WAS_DAEMON_ONLY_adapter_max_shrmem	32000000
WAS_DAEMON_ONLY_enable_adapter	true

Figure 13-4 Typical values for different WOLA-related variables

e. Save the variables.

3. WOLA requires a resource adapter, which now needs to be installed from an archive, and also a connection factory. This resource archive is called `ola.rar`. To install this resource adapter, complete the following steps:

   a. In the WebSphere Application Server administrative console, select the **Resources** menu, expand **Resource Adapters**, and select **Resource Adapters**, as shown in Figure 13-5.

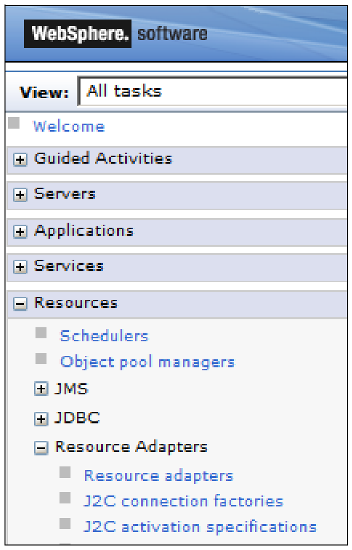
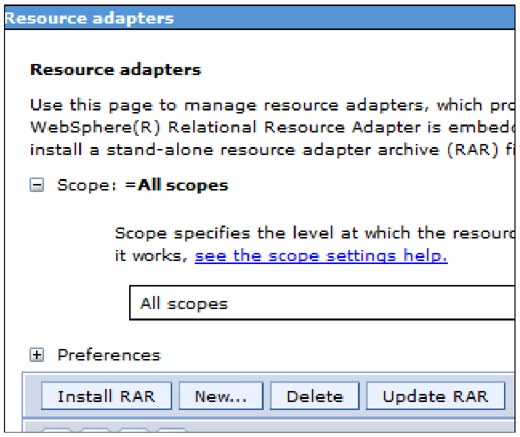


Figure 13-5 Resources menu in the administrative console

   b. On the Resource adapters page, click **Install RAR**, as shown in Figure 13-6.


Resource adapters

Resource adapters

Use this page to manage resource adapters, which pro
WebSphere(R) Relational Resource Adapter is embedd
install a stand-alone resource adapter archive (RAR) fi

Scope: =**All scopes**

Scope specifies the level at which the resourc
it works, see the scope settings help.

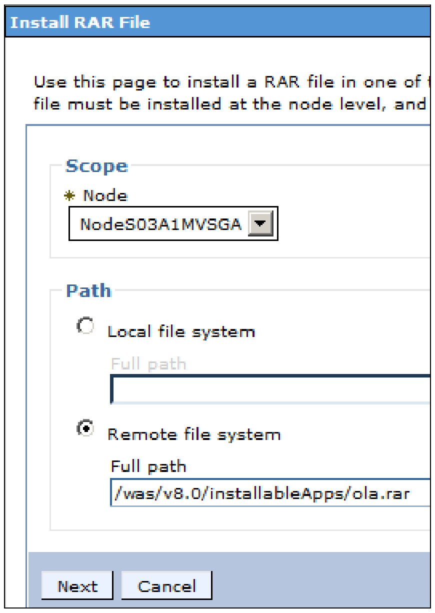
All scopes

Preferences

Install RAR | New... | Delete | Update RAR

Figure 13-6 Selecting Install RAR

c. Select **Remote file system** and select the **ola.rar** file from the `installableApps` directory within your WebSphere Application Server installation directory, as shown in Figure 13-7. Click **Next**.



Install RAR File

Use this page to install a RAR file in one of
file must be installed at the node level, and

Scope

* Node
NodeS03A1MVSGA

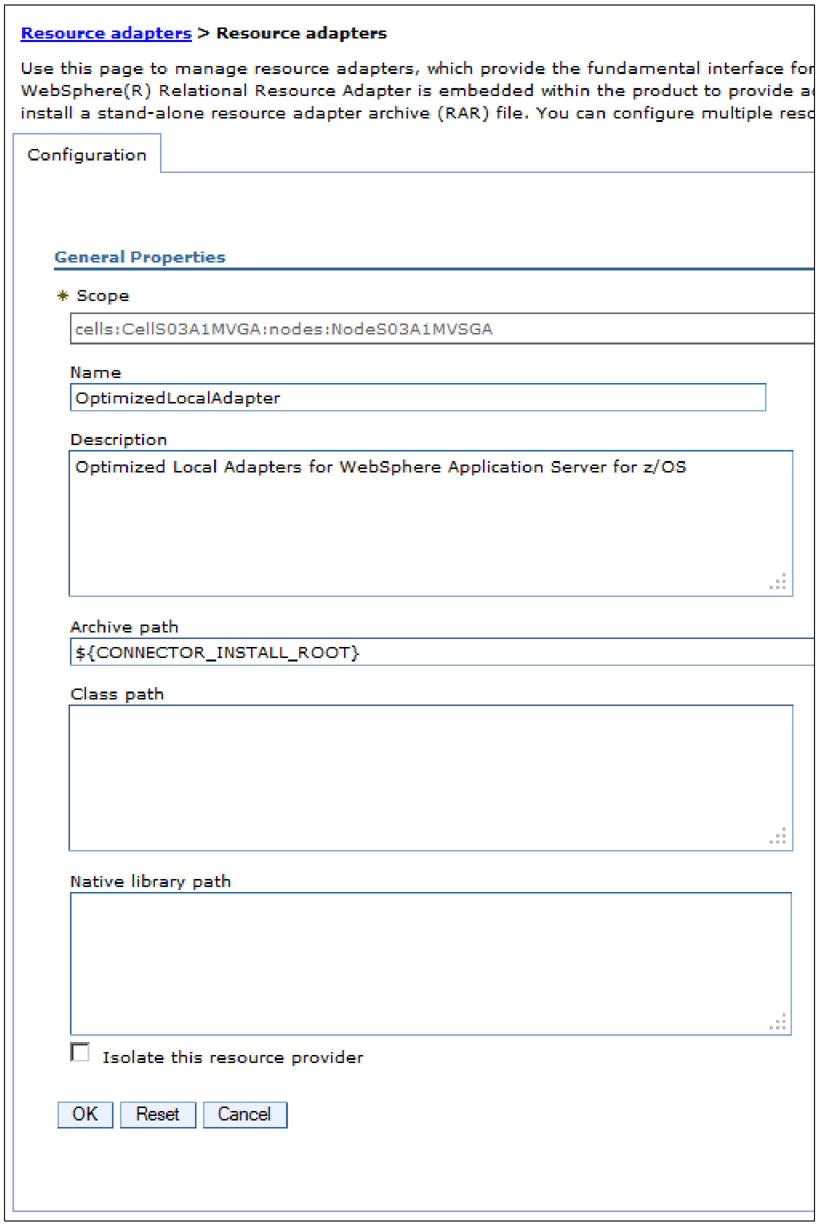
Path

Local file system
Full path

[✓ Remote file system]
Full path
/was/v8.0/installableApps/ola.rar

Figure 13-7 Installing the Resource Archive file

d. Accept the defaults on the Resource adapters page, as shown in Figure 13-8. Click **OK** to save.



Resource adapters > **Resource adapters**

Use this page to manage resource adapters, which provide the fundamental interface for
WebSphere(R) Relational Resource Adapter is embedded within the product to provide a
install a stand-alone resource adapter archive (RAR) file. You can configure multiple reso

Configuration

General Properties

* Scope: cells:CellS03A1MVGA:nodes:NodeS03A1MVSGA

Name: OptimizedLocalAdapter

Description: Optimized Local Adapters for WebSphere Application Server for z/OS

Archive path: ${CONNECTOR_INSTALL_ROOT}

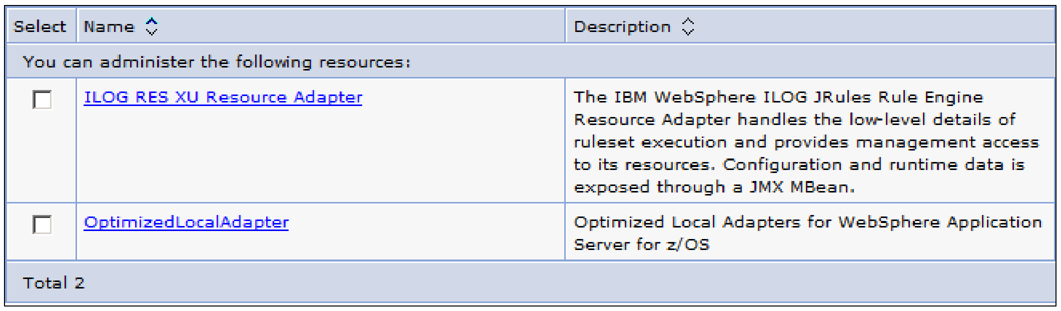
Class path:

Native library path:

☐ Isolate this resource provider

Figure 13-8 Installing the resource adapter

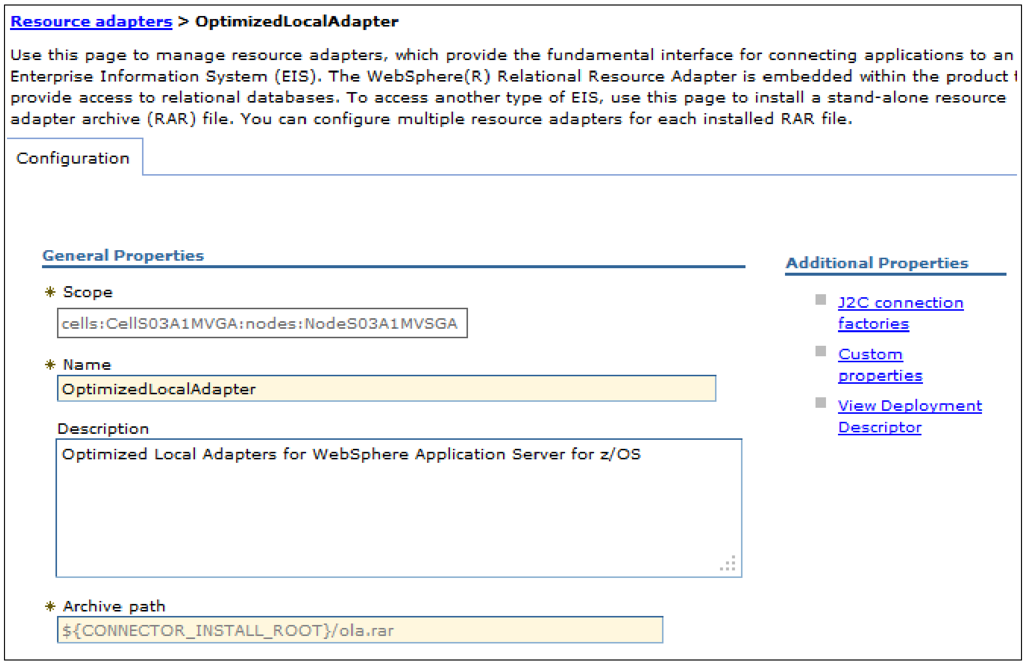
e. The RAR file is now in the list of installed resource adapters, as shown in Figure 13-9. Create a connection factory by clicking the **OptimizedLocalAdapter** resource.



Select	Name	Description
You can administer the following resources:		
☐	ILOG RES XU Resource Adapter	The IBM WebSphere ILOG JRules Rule Engine Resource Adapter handles the low-level details of ruleset execution and provides management access to its resources. Configuration and runtime data is exposed through a JMX MBean.
☐	OptimizedLocalAdapter	Optimized Local Adapters for WebSphere Application Server for z/OS
Total 2		

Figure 13-9 List of resource adapters

f. On the OptimizedLocalAdapter page, as shown in Figure 13-10, under Additional Properties, click **J2C connection factories**.



Resource adapters > **OptimizedLocalAdapter**

Use this page to manage resource adapters, which provide the fundamental interface for connecting applications to an Enterprise Information System (EIS). The WebSphere(R) Relational Resource Adapter is embedded within the product t provide access to relational databases. To access another type of EIS, use this page to install a stand-alone resource adapter archive (RAR) file. You can configure multiple resource adapters for each installed RAR file.

General Properties

* Scope
cells:CellS03A1MVGA:nodes:NodeS03A1MVSGA

* Name
OptimizedLocalAdapter

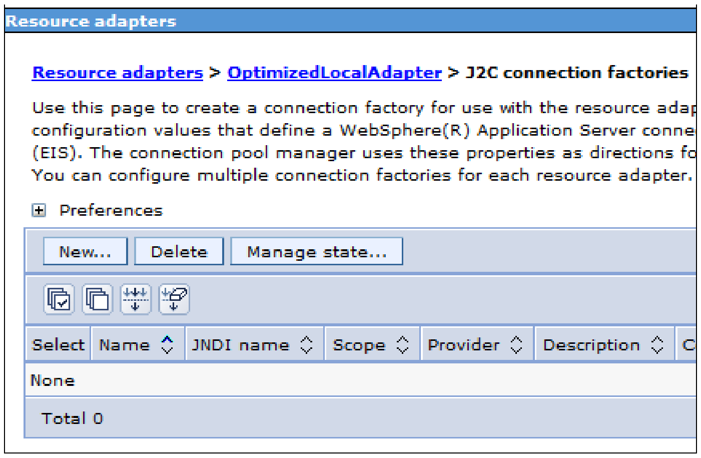
Description
Optimized Local Adapters for WebSphere Application Server for z/OS

* Archive path
${CONNECTOR_INSTALL_ROOT}/ola.rar

Additional Properties

Figure 13-10 Optimized Local Adapter details

g. On the J2C connection factories page (Figure 13-11), click **New**.


Resource adapters

Resource adapters > OptimizedLocalAdapter > J2C connection factories

Use this page to create a connection factory for use with the resource adap
configuration values that define a WebSphere(R) Application Server conne
(EIS). The connection pool manager uses these properties as directions fo
You can configure multiple connection factories for each resource adapter.

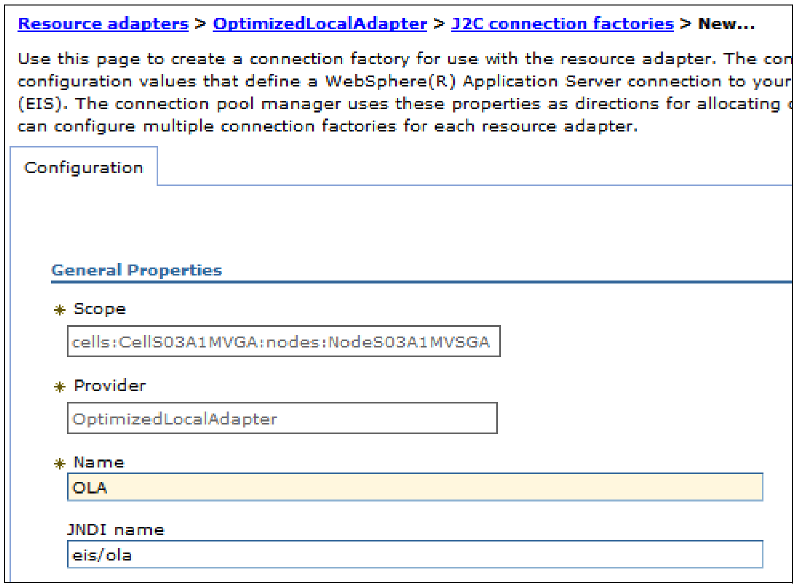
⊞ Preferences

New... | Delete | Manage state...

Select	Name	JNDI name	Scope	Provider	Description	C
None						
Total 0						

Figure 13-11 J2C connection factories page

h. Enter the details, as shown in Figure 13-12. Fields that are unavailable do not need to be populated. Save your changes.


Resource adapters > OptimizedLocalAdapter > J2C connection factories > New...

Use this page to create a connection factory for use with the resource adapter. The con
configuration values that define a WebSphere(R) Application Server connection to your
(EIS). The connection pool manager uses these properties as directions for allocating c
can configure multiple connection factories for each resource adapter.

Configuration

General Properties

* Scope
cells:CellS03A1MVGA:nodes:NodeS03A1MVSGA

* Provider
OptimizedLocalAdapter

* Name
OLA

JNDI name
eis/ola

Figure 13-12 Populating the fields for the J2C connection factory

4. Restart WebSphere Application Server to pick up the changes. You see messages in the WebSphere Application Server logs that indicate the WOLA status:

```
Support is activated: BBOM0001I enable_adapter:1
```

5. It is now necessary install the WOLA Enterprise JavaBeans (EJB) into WebSphere Application Server. This application is used to listen for the WOLA input.

Note: The following instructions are specific for WebSphere Application Server V8.0. If you are using a different version, other configuration might be necessary. For more information, see the *Enabling the server environment to use optimized local adapters* topic in the WebSphere Application Server, Network Deployment, Version 8.0 Information Center:

`http://pic.dhe.ibm.com/infocenter/wasinfo/v8r0/index.jsp?topic=%2Fcom.ibm.websphere.nd.multiplatform.doc%2Finfo%2Fae%2Fae%2Ftdat_enableconnector.html`

Complete the following steps:

a. In the WebSphere Application Server administrative console, select **Application** → **New Application** and select **New Enterprise Application** from the New Application panel, as shown in Figure 13-13.

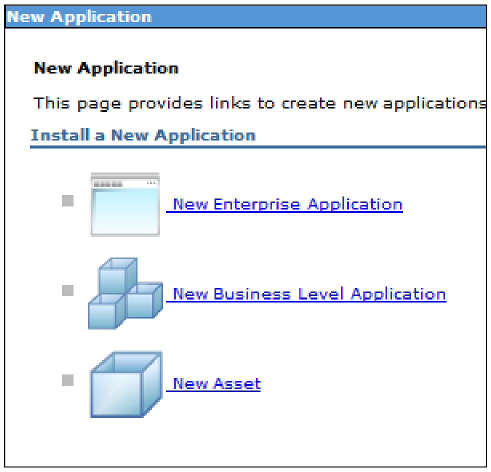
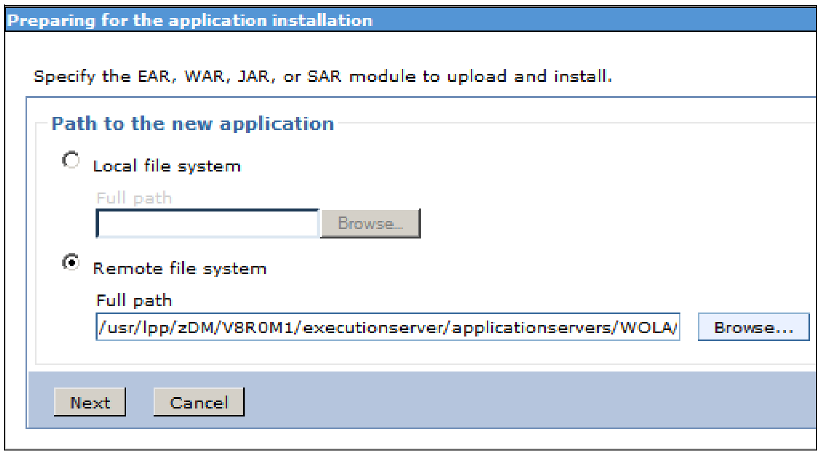

New Application
New Application
This page provides links to create new applications
Install a New Application
New Enterprise Application
New Business Level Application
New Asset

Figure 13-13 WebSphere Application Server New Application panel

b. On the Preparing for the application installation panel, select **Remote file system** and click **Browse** to help you select the location of the WOLA EAR file on the z/OS server. It is in the `/executionserver/applicationservers/WOLA` directory of your installation. An example is shown in Figure 13-14.


Preparing for the application installation
Path to the new application
Browse...
Browse...
Next
Cancel

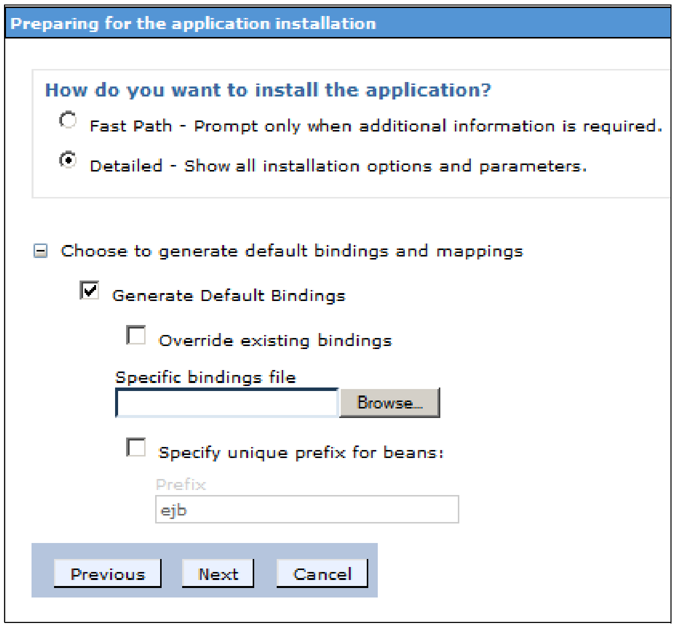
Specify the EAR, WAR, JAR, or SAR module to upload and install.

Local file system
Full path

[✓ Remote file system]
Full path
/usr/lpp/zDM/V8R0M1/executionserver/applicationservers/WOLA/

Figure 13-14 Preparing for the application installation

c. On the following window, select **Detailed** for the type of installation. Expand **Choose to generate default bindings and mappings** and check **Generate Default Bindings**, as shown in Figure 13-15. Click **Next**.


Preparing for the application installation

How do you want to install the application?

Fast Path - Prompt only when additional information is required. [✓ Detailed - Show all installation options and parameters.]

Choose to generate default bindings and mappings

[✓] Generate Default Bindings

[ ] Override existing bindings

Specific bindings file

[ ] Specify unique prefix for beans:

Prefix: ejb

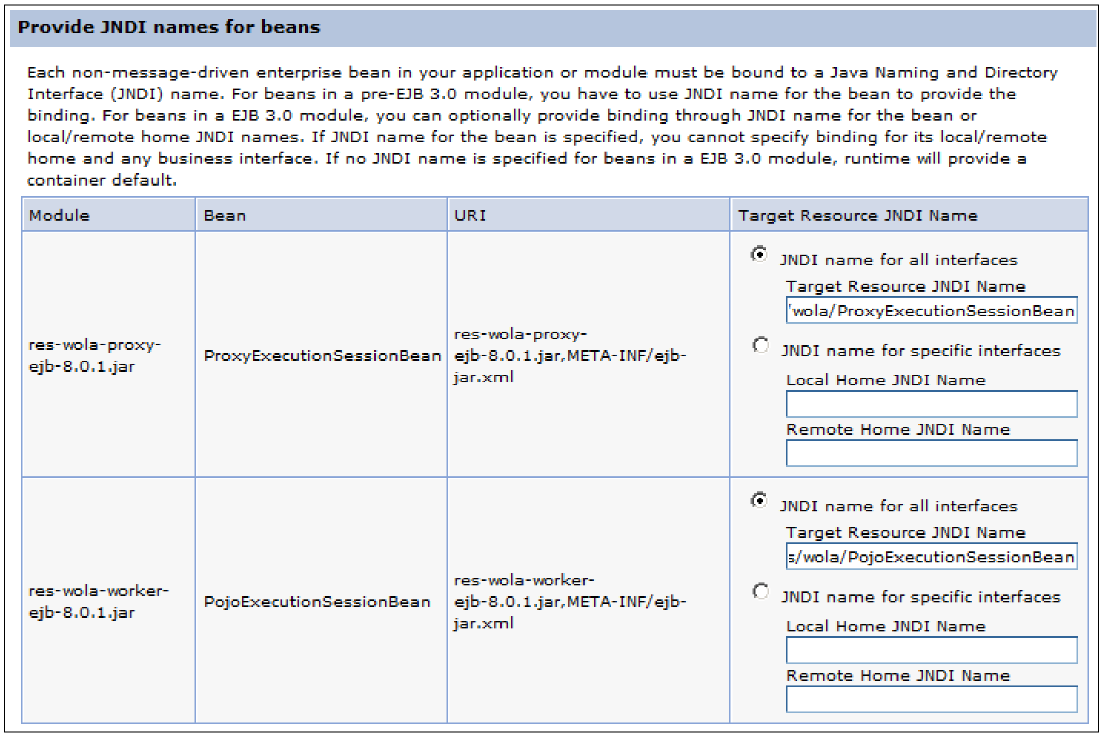
Figure 13-15 Selecting the options for installing the application

d. On the Install New Application window, click **Next** until you reach step 5.

e. For `res-wola-proxy-ejb-8.0.1.jar`, select **JNDI for all interfaces**. If it is not already entered, enter `ejb/com/ibm/rules/wola/ProxyExecutionSessionBean` for the Target Resource JNDI Name.

For `res-wola-worker-ejb-8.0.1.jar`, select **JNDI for all interfaces**. If it is not already entered, enter `ejb/com/ibm/rules/wola/PojoExecutionSessionBean` for the Target Resource JNDI Name.

Figure 13-16 shows the step to provide JNDI names for beans. Then, click **Next** until you reach step 7.



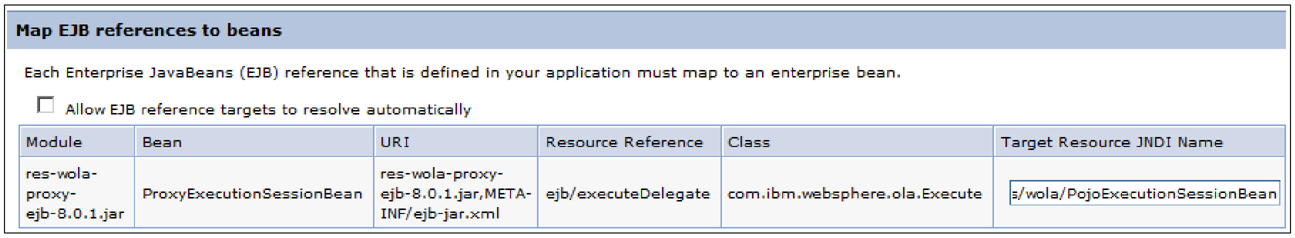
Provide JNDI names for beans

Each non-message-driven enterprise bean in your application or module must be bound to a Java Naming and Directory Interface (JNDI) name. For beans in a pre-EJB 3.0 module, you have to use JNDI name for the bean to provide the binding. For beans in a EJB 3.0 module, you can optionally provide binding through JNDI name for the bean or local/remote home JNDI names. If JNDI name for the bean is specified, you cannot specify binding for its local/remote home and any business interface. If no JNDI name is specified for beans in a EJB 3.0 module, runtime will provide a container default.

Module	Bean	URI	Target Resource JNDI Name
res-wola-proxy-ejb-8.0.1.jar	ProxyExecutionSessionBean	res-wola-proxy-ejb-8.0.1.jar,META-INF/ejb-jar.xml	[✓ JNDI name for all interfaces] Target Resource JNDI Name: 'wola/ProxyExecutionSessionBean; [ ] JNDI name for specific interfaces; Local Home JNDI Name: ; Remote Home JNDI Name:
res-wola-worker-ejb-8.0.1.jar	PojoExecutionSessionBean	res-wola-worker-ejb-8.0.1.jar,META-INF/ejb-jar.xml	[✓ JNDI name for all interfaces] Target Resource JNDI Name: s/wola/PojoExecutionSessionBean; [ ] JNDI name for specific interfaces; Local Home JNDI Name: ; Remote Home JNDI Name:

Figure 13-16 Step 5: Provide JNDI names for beans

f. Step 7 requires the mapping of EJB references to beans. For the Module column for `res-wola-proxy-ejb-8.0.1.jar`, if it is not already entered, enter the Target Resource JNDI Name of `ejb/com/ibm/rules/wola/PojoExecutionSessionBean`, as shown in Figure 13-17. Click **Next** until you reach the Summary window.



Map EJB references to beans

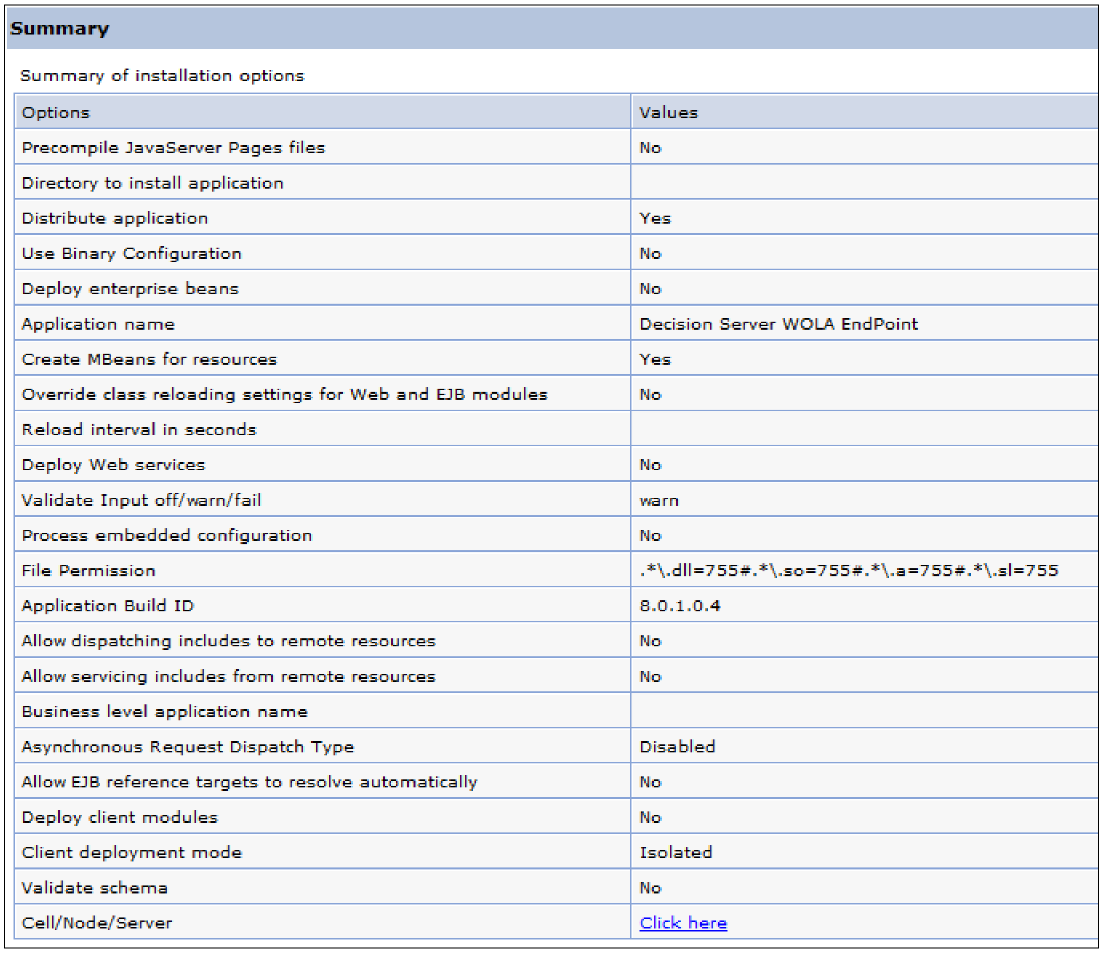
Each Enterprise JavaBeans (EJB) reference that is defined in your application must map to an enterprise bean.

☐ Allow EJB reference targets to resolve automatically

Module	Bean	URI	Resource Reference	Class	Target Resource JNDI Name
res-wola-proxy-ejb-8.0.1.jar	ProxyExecutionSessionBean	res-wola-proxy-ejb-8.0.1.jar,META-INF/ejb-jar.xml	ejb/executeDelegate	com.ibm.websphere.ola.Execute	s/wola/PojoExecutionSessionBean

Figure 13-17 Mapping the EJB reference to the bean

g. On the Summary window, as shown in Figure 13-18, click **Finish**. The application is installed. Click **Save** to save the changes.



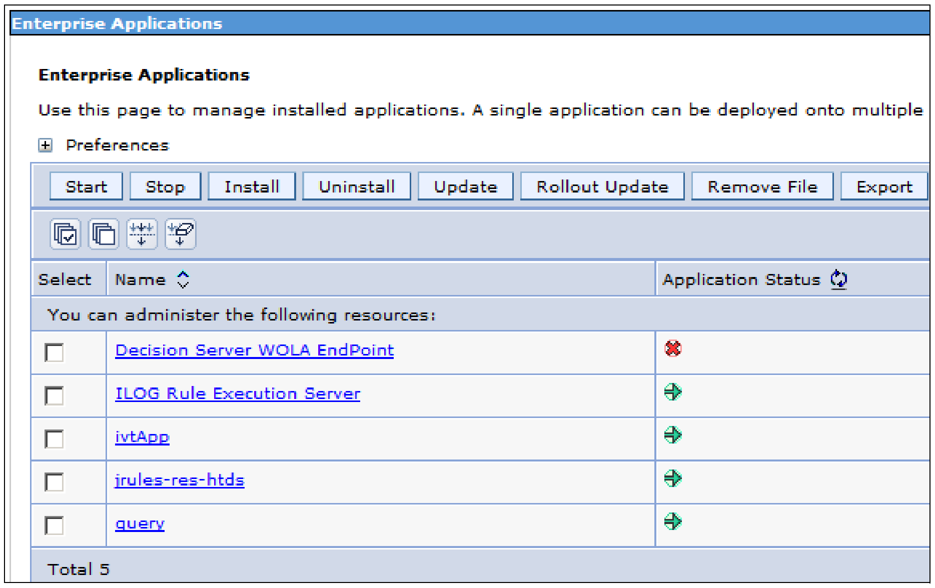
Summary

Summary of installation options

Options	Values
Precompile JavaServer Pages files	No
Directory to install application	
Distribute application	Yes
Use Binary Configuration	No
Deploy enterprise beans	No
Application name	Decision Server WOLA EndPoint
Create MBeans for resources	Yes
Override class reloading settings for Web and EJB modules	No
Reload interval in seconds	
Deploy Web services	No
Validate Input off/warn/fail	warn
Process embedded configuration	No
File Permission	.*\.dll=755#.*\.so=755#.*\.a=755#.*\.sl=755
Application Build ID	8.0.1.0.4
Allow dispatching includes to remote resources	No
Allow servicing includes from remote resources	No
Business level application name	
Asynchronous Request Dispatch Type	Disabled
Allow EJB reference targets to resolve automatically	No
Deploy client modules	No
Client deployment mode	Isolated
Validate schema	No
Cell/Node/Server	Click here

Figure 13-18 Summary of application installation

h. The application can be displayed by selecting **Applications** → **Application Types** → **WebSphere Enterprise Applications**, as shown in Figure 13-19, where the application is listed as Decision Server WOLA EndPoint. Start the application by selecting it and clicking **Start**.



Use this page to manage installed applications. A single application can be deployed onto multiple

Select	Name	Application Status
You can administer the following resources:		
☐	Decision Server WOLA EndPoint	✖
☐	ILOG Rule Execution Server	➔
☐	ivtApp	➔
☐	jrules-res-htds	➔
☐	query	➔
Total 5		

Figure 13-19 List of installed applications that include WOLA Endpoint

13.3 Batch programs and Rule Execution Server using WOLA on z/OS

With the previous configurations already in place (setting up the variables as described in 13.1, "Overview of WebSphere Operational Local Adapters" on page 226 and setting up the WebSphere Application Server as described in 13.2, "Sample configuration of WebSphere Application Server to use WOLA" on page 226), there are no further requirements to connect a batch program to a RES running in WebSphere Application Server for z/OS using WOLA.

13.4 CICS and Rule Execution Server using WOLA on z/OS

Several of the steps described in this section are the same steps that are required for setting up CICS to work with RES. These steps are indicated. However, there are some differences to ensure that CICS is aware of the location of WOLA and the details about which WebSphere Application Server for z/OS WOLA is accessing. Complete the following steps:

1. Check the value of the *ola_cicsuser_identity_propagate* variable. This variable is used to specify permissions for CICS application level identities to be used for authentication when calling the rule. By default, it is set to 0, which indicates undefined. More information about this variable is in the *Optimized local adapters environment variables* topic of the WebSphere Application Server, Network Deployment, Version 8.0 Information Center:

   `http://pic.dhe.ibm.com/infocenter/wasinfo/v8r0/index.jsp?topic=%2Fcom.ibm.websphere.nd.multiplatform.doc%2Finfo%2Fae%2Fae%2Fcdat_olacustprop.html`

2. Similar to setting up CICS for RES, submit the JCL job HBRCSD to define the resources that are required by CICS.

3. Submit the job HBRCWOLA. This job defines the resources that are required for WOLA under CICS.
4. Edit the CICS system initialization table. For setting up CICS for RES, add the value of the **CICSLIST** parameter to the list of resource definition groups specified by the **GRPLIST** parameter. To work with WOLA, also add the name BBOLIST to the **GRPLIST** parameter to ensure that CICS can locate the necessary transactions.
5. When setting up CICS for RES, the `SHBRAUTH` load libraries needed to be added to STEPLIB and DFHRPL concatenations in the CICS JCL. To work with WOLA, you also need to add the `SHBRWOLA` libraries, for example:

```
//STEPLIB DD DISP=SHR,DSN=WODM.V8R0M1.HBR1.CUSTMISE.HBR1.SHBRAUTH
// DD DISP=SHR,DSN=WODM.OLA.LOADLIB
//DFHRPL DD DISP=SHR,DSN=WODM.V8R0M1.HBR1.CUSTMISE.HBR1.SHBRCICS
// DD DISP=SHR,DSN=WODM.OLA.LOADLIB
```

6. Pass the necessary runtime variables to the server by adding the SHBRPARM(HBRWOLA) and SHBRPARM(HBRCMMN) data set members to the HBRENVPR data definition (DD) statement, as shown in the following DD statements:

```
//HBRENVPR DD DISP=SHR,DSN=WODM.V8R0M1.HBR1.CUSTMISE.HBR1.SHBRPARM(HBRWOLA)
//         DD DISP=SHR,DSN=WODM.V8R0M1.HBR1.CUSTMISE.HBR1.SHBRPARM(HBRCMMN)
```

7. Start the WebSphere Application Server and CICS.
8. RES requires the HBRGROUP resources to be installed. WOLA requires the BBOACSD group to be installed. Use the following commands:

```
CEDA INSTALL GROUP(HBRGROUP)
CEDA INSTALL GROUP(BBOACSD)
```

9. Activate the optimized local adapters TRUE program by using the following command:

```
BBOC START_TRUE
```

10. The configuration can be tested by running the HBRC transaction from CICS.

For more information, see the *Use the CICS environment* topic in the WebSphere Application Server, Network Deployment, Version 8.0 Information Center:

http://pic.dhe.ibm.com/infocenter/wasinfo/v8r0/index.jsp?topic=%2Fcom.ibm.websphere.nd.multiplatform.doc%2Finfo%2Fae%2Fae%2Ftdat_useola_in_step4.html

13.5 IMS and Rule Execution Server using WOLA on z/OS

To connect IMS to RES using a WOLA interface, it is necessary to provide IMS with details of the location of that WOLA. In each case, follow this process:

1. There needs to be an entry in the external subsystem member in the IMS PROCLIB to contain an entry to indicate that WOLA must be used. If you do not already have a member, you need to create one. Include this entry:

```
WOLA,BBOA,BBOAIEMT
```

2. Pass the **SSM** parameter into your IMS startup data.
3. The WOLA load library that is created during the WOLA setup needs to be included in the IMS control region startup JCL in both the `STEPLIB` and the `DFSESL` DDs:
   - BMP: You need to restart IMS in order to pick up the changes.
   - MPR: MPR requires additional steps for setup:

a. Similar to setting up for calling out to RES (as described in 12.2, “Configuration” on page 222), include an `HBRENVPR DD` in your message processing region JCL. In this case, ensure that it points to a member that contains the WOLA parameters rather than the RES group.

b. Similar to setting up for calling out to RES, add the WOLA load library to your `STEPLIBR` and `DFSESL DD` statements in the message processing JCL.

The message processing region needs to be restarted for IMS to pick up the changes:

– DL/I: DL/I programs are currently not supported in this environment.

For information, see the Use the *IMS environment* topic in the WebSphere Application Server, Network Deployment, Version 8.0 Information Center:

http://pic.dhe.ibm.com/infocenter/wasinfo/v8r0/index.jsp?topic=%2Fcom.ibm.websphere.nd.multiplatform.doc%2Finfo%2Fae%2Fae%2Ftdat_useola_in_step5.html

14

Configuring decision warehousing

This chapter describes the Decision Warehouse, which is part of the Rules Execution server. This chapter explains the configuration of the Decision Warehouse and describes possible uses.

The following topics are covered in this chapter:

- 14.1, “Introducing the Decision Warehouse” on page 242
- 14.2, “Configuring the Decision Warehouse” on page 242

© Copyright IBM Corp. 2013. All rights reserved.

14.1 Introducing the Decision Warehouse

The Decision Warehouse is used for monitoring and reporting on ruleset execution. With ruleset monitoring enabled, details of each ruleflow used, the path, and the rules fired are recorded. The purpose of the Decision Warehouse function is to help you understand what happened when a ruleset was executed. This data might be required for auditing or performance analysis. It can also be used when testing, as described in Chapter 15, “Configuring and running Decision Validation Services” on page 245.

The Decision Warehouse is accessed by using the Rules Execution Server for z/OS (zRES) console or the console for Rules Execution Server on WebSphere Application Server for z/OS (RES). The details of recorded statistics are stored in a database for future reporting.

Decision Warehouse is shipped in a default configuration so that it can be used immediately.

14.2 Configuring the Decision Warehouse

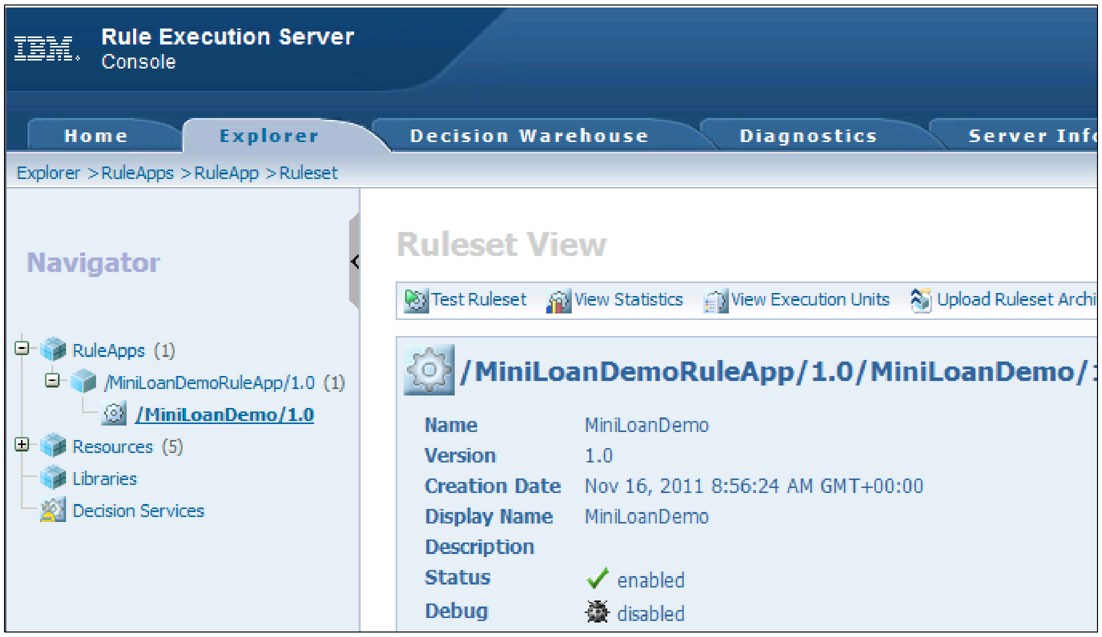
Complete the following steps to configure the Decision Warehouse:

1. Set up the database resources. During the normal Operational Decision Manager configuration, the Decision Warehouse Database definitions, customized to your environment for the Decision Warehouse function, are presented in the partitioned data set (PDS) member:

   ++*HBRWORKDS*++.SHBRJCL(HBRDSCTR)

   Run the HBRDSCTR job to create the database entities that are required to store the rules data that is traced during rule execution.

2. Enable ruleset monitoring. To monitor ruleset execution, you must set the monitoring options in the Ruleset View, as shown in Figure 14-2 on page 243, by using the Rule Execution Server console that is shown in Figure 14-1.



/MiniLoanDemoRuleApp/1.0/MiniLoanDemo/

Name	MiniLoanDemo
Version	1.0
Creation Date	Nov 16, 2011 8:56:24 AM GMT+00:00
Display Name	MiniLoanDemo
Description	
Status	✓ enabled
Debug	disabled

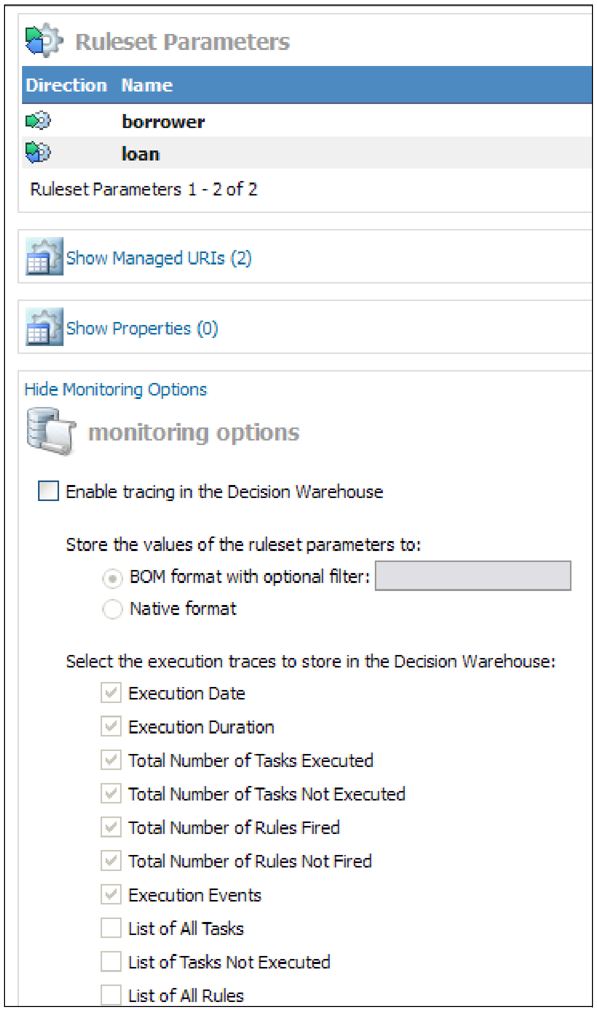
Figure 14-1 Ruleset view

Follow these steps:

a. Select the **Explorer** tab.

b. From the Navigator view, select **RuleApps**, and then the ruleset.

c. On the Ruleset Parameters page (Figure 14-2), under the monitoring options section, enable ruleset execution monitoring by selecting **Enable tracing in the Decision Warehouse**. Click **Save**.



Ruleset Parameters

Direction	Name
	borrower
	loan

Ruleset Parameters 1 - 2 of 2

Show Managed URIs (2)

Show Properties (0)

Hide Monitoring Options

monitoring options

- [ ] Enable tracing in the Decision Warehouse

Store the values of the ruleset parameters to:

- ● BOM format with optional filter:
- ○ Native format

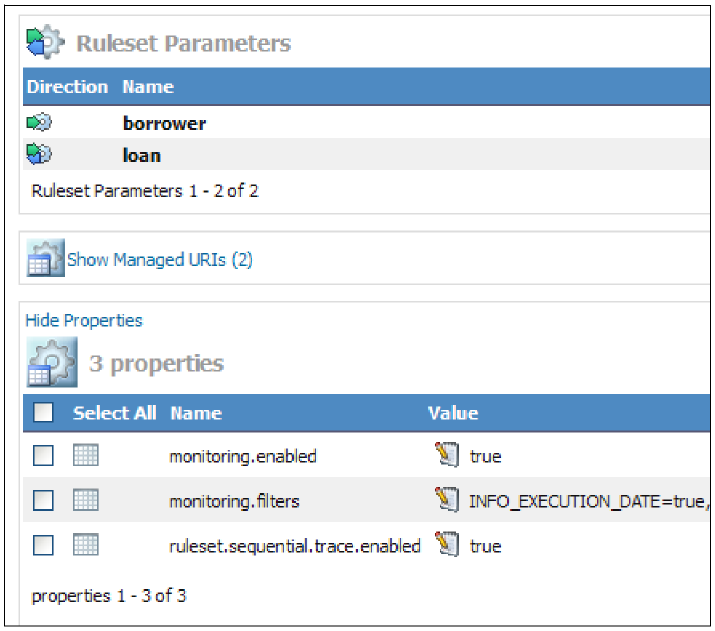
Select the execution traces to store in the Decision Warehouse:

- [✓] Execution Date
- [✓] Execution Duration
- [✓] Total Number of Tasks Executed
- [✓] Total Number of Tasks Not Executed
- [✓] Total Number of Rules Fired
- [✓] Total Number of Rules Not Fired
- [✓] Execution Events
- [ ] List of All Tasks
- [ ] List of Tasks Not Executed
- [ ] List of All Rules

Figure 14-2 Monitoring options

d. Check that the following ruleset properties are set to **true**, as shown in Figure 14-3:

- monitoring.enabled
- ruleset.sequential.trace.enabled (if the ruleset contains tasks that use the sequential or fast path execution mode)



Direction	Name
	borrower
	loan

Ruleset Parameters 1 - 2 of 2

Select All	Name	Value
☐	monitoring.enabled	true
☐	monitoring.filters	INFO_EXECUTION_DATE=true,
☐	ruleset.sequential.trace.enabled	true

properties 1 - 3 of 3

Figure 14-3 Monitoring properties

3. Execute the ruleset. The execution trace data is written to the default Decision Warehouse database.

4. View the results. From the Rule Execution Server console, select the **Decision Warehouse** tab. Use the decision identifier (Decision ID) to locate a specific transaction and view the executed rules. A decision identifier is automatically generated, by default, and is equal to the identifier of the execution unit (XU) connection.

15

Configuring and running Decision Validation Services

This chapter describes the configuration of Rules Execution Server for z/OS (zRES) and Rules Execution Server on WebSphere Application Server for z/OS (RES) so that Decision Validation Services (DVS) can be used. This chapter also describes the details of the Service Scenario Provider (SSP) and the Key Performance Indicator (KPI) architectures.

An example of how to configure and run a typical test using DVS is included in this chapter. This example can assist you with the configuration of DVS projects and demonstrate some of the uses of DVS.

The following topics are covered in this chapter:

- 15.1, “Decision Validation Service for stand-alone zRES” on page 246
- 15.2, “DVS using RES on WebSphere Application Server for z/OS” on page 258
- 15.3, “Service scenario provider and key performance indicator architecture” on page 270
- 15.4, “Example” on page 275

© Copyright IBM Corp. 2013. All rights reserved.

15.1 Decision Validation Service for stand-alone zRES

This section describes how to configure zRES on z/OS so that simulations and test suites can be run there. These steps are in addition to the standard steps in setting up DVS, as described in the *Configuring Decision Validation Services on WebSphere Application Server for z/OS* topic in the IBM Operational Decision Manager Version 8.0.1 Information Center:

http://pic.dhe.ibm.com/infocenter/dmanager/v8r0m1/index.jsp?topic=%2Fcom.ibm.wodm.family.config.zos%2Ftopics%2Ftpc_dc_add_dvs_config_steps.html

Important: The instructions in this section assume that your Rules Project has been developed in Rules Designer.

During the initial configuration of zRES, ensure that the following variables are set:

- The `++HBRMODE++` variable must be set to `TEST`, which indicates that the server will run in test mode.
- The `++HBRSSPPORT++` variable indicates the port on which the SSP is located.

For more information about these properties, see "Rules z/OS" on page 328.

Tip: By default, the zRES console and SSP on zRES run from different port numbers. To use the same port number, follow the instructions in the *Repackaging Decision Center to locate the SSP* topic in the IBM Operational Decision Manager Version 8.0.1 Information Center:

http://pic.dhe.ibm.com/infocenter/dmanager/v8r0m1/index.jsp?topic=%2Fcom.ibm.wodm.family.config.zos%2Ftopics%2Ftsk_ssp_port_location.html

15.1.1 Running from Rules Designer

After you set up zRES, complete the following steps in this section from the Rules Designer.

Deploying the XOM to zRES

Complete the following steps to deploy the execution object module (XOM) to zRES:

1. In Rules Designer, right-click the rules project, and select **Decision Validation Services → Deploy XOM for Remote DVS Testing**, as shown in Figure 15-1.

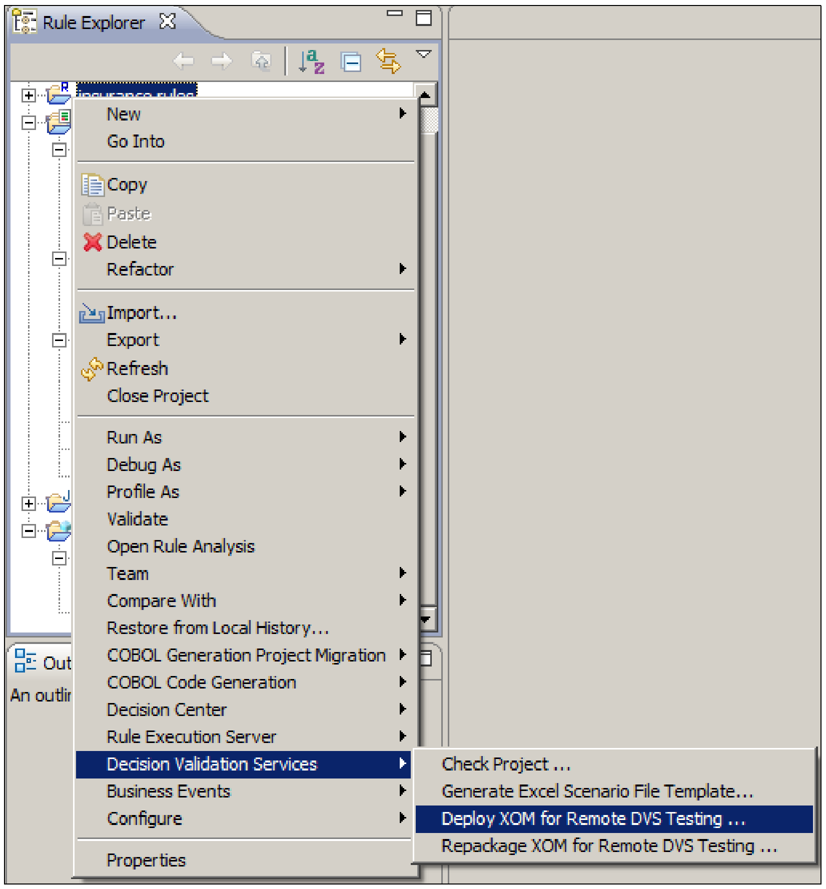
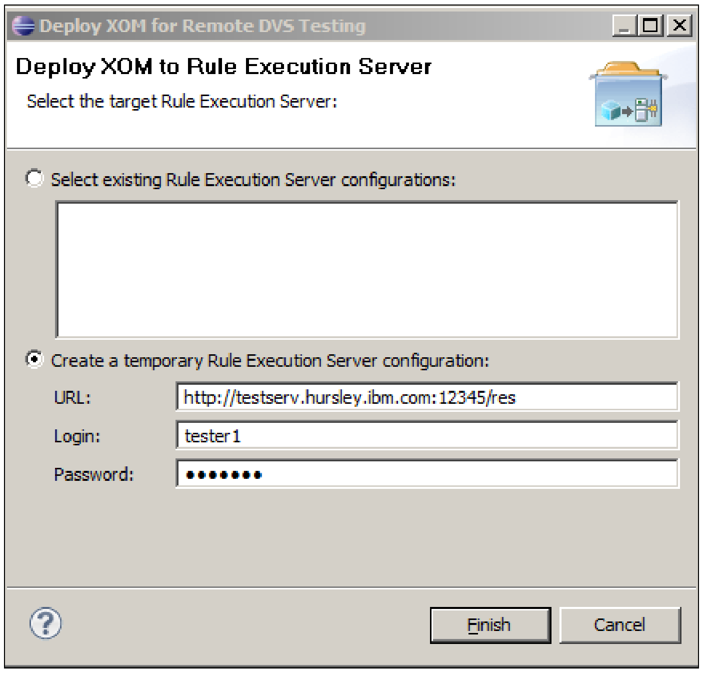


Figure 15-1 Selecting to deploy XOM for remote DVS testing

2. On the Deploy XOM for Remote DVS Testing window, populate the fields with the details of your zRES server, as indicated in Figure 15-2, and click **Finish** to deploy the XOM. On this window, you need to use your *console* port.


Deploy XOM for Remote DVS Testing
Deploy XOM to Rule Execution Server
Select the target Rule Execution Server:
Finish
Cancel

( ) Select existing Rule Execution Server configurations:

(●) Create a temporary Rule Execution Server configuration:

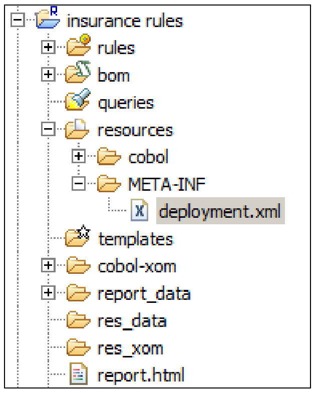
URL:	http://testserv.hursley.ibm.com:12345/res
Login:	tester1
Password:	•••••••

Figure 15-2 Deploying the XOM to the Rule Execution Server

Updating the deployment.xml file

Complete the following steps to correct the `deployment.xml` file to handle the zRES server:

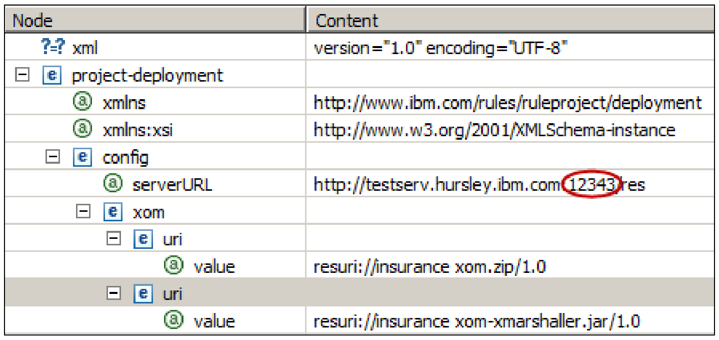
1. After you deploy the XOM to your zRES, the `deployment.xml` file is created in the `resources/META-INF` directory of your rule project, as shown in Figure 15-3. Open this file for editing.



```
insurance rules
  rules
  bom
  queries
  resources
    cobol
    META-INF
      deployment.xml
  templates
  cobol-xom
  report_data
  res_data
  res_xom
  report.html
```

Figure 15-3   Deployment.xml file

2. In the `deployment.xml` file, correct the port for the rules server so that it points to the *SSP* port. See Figure 15-4.



Node	Content
xml	version="1.0" encoding="UTF-8"
project-deployment	
xmlns	http://www.ibm.com/rules/ruleproject/deployment
xmlns:xsi	http://www.w3.org/2001/XMLSchema-instance
config	
serverURL	http://testserv.hursley.ibm.com:12343/res
xom	
uri	
value	resuri://insurance xom.zip/1.0
uri	
value	resuri://insurance xom-xmarshaller.jar/1.0

Figure 15-4   Changing the port to the SSP port

3. Save and close the `deployment.xml` file.

Tip: If you intend to redeploy the XOM, you need to restore the original port to deploy the `deployment.xml` file correctly. After you redeploy the XOM, restore the `deployment.xml` file to the SSP port. The system uses one port for *deployment* and another port for SSP.

Setting up the DVS project

Complete the following steps to set up the DVS project:

1. Right-click and select **New**. Select the **DVS Project** wizard, as shown in Figure 15-5. Click **Next** to open the wizard.

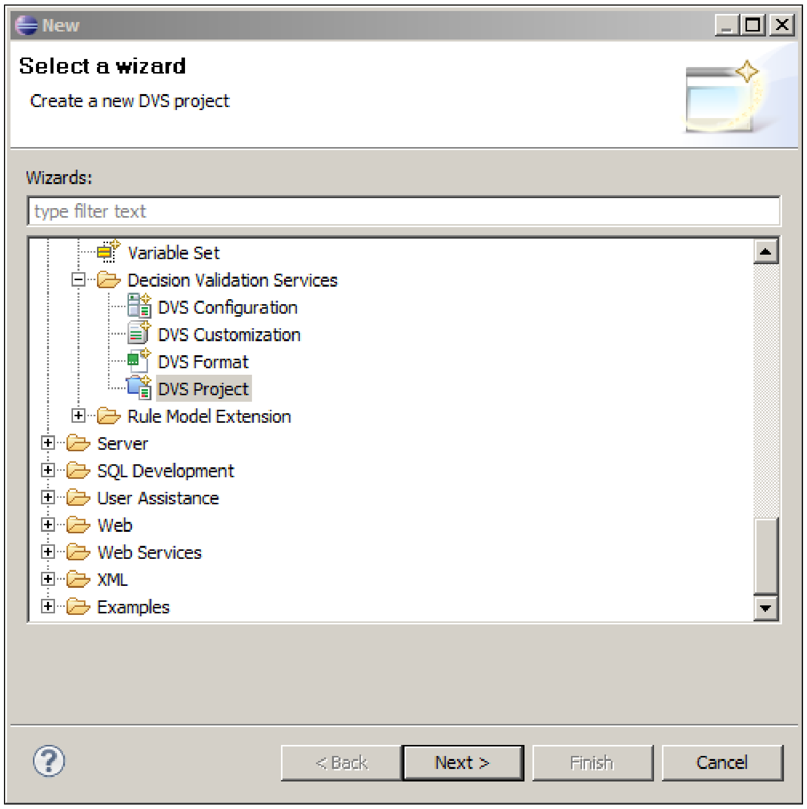
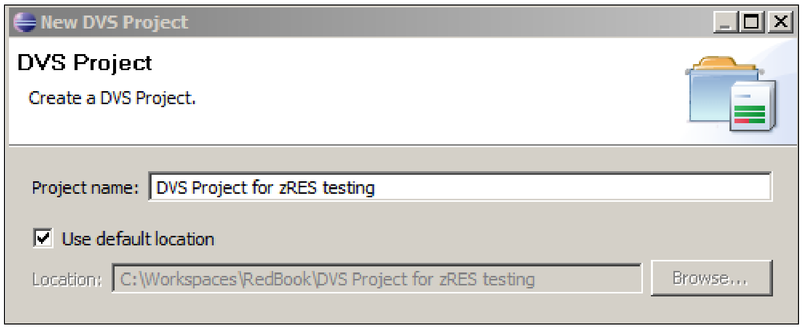


Figure 15-5 Selecting the DVS Project wizard

2. Enter your project name and click **Next**, as shown in Figure 15-6.



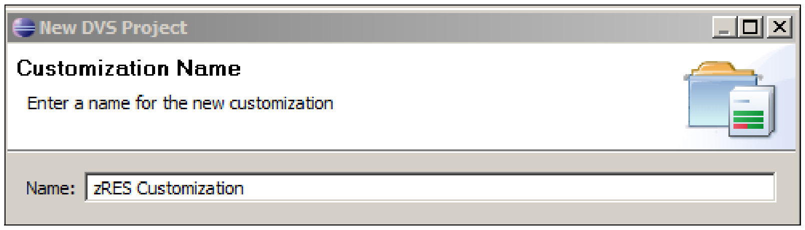
Project name: DVS Project for zRES testing

☑ Use default location

Location: C:\Workspaces\RedBook\DVS Project for zRES testing

Figure 15-6 Creating the DVS project and specifying the project name

3. Enter the name for the customization and click **Finish**, as shown in Figure 15-7.



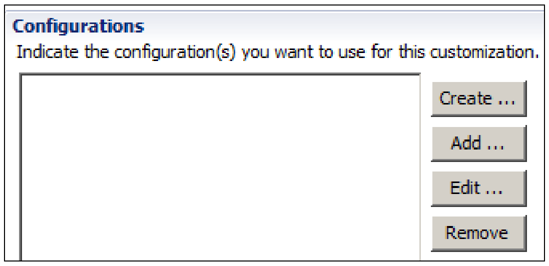
Customization Name

Enter a name for the new customization

Name: zRES Customization

Figure 15-7 Creating the DVS project and specifying the customization name

4. Create a new configuration by clicking **Create**, as shown in Figure 15-8.

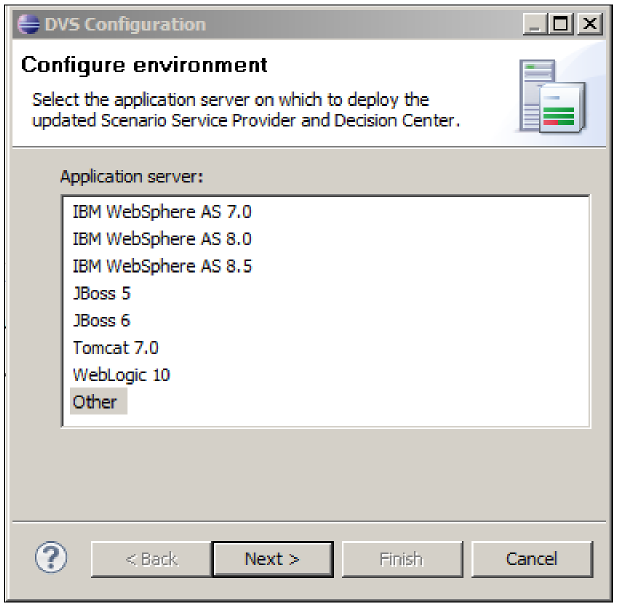


Configurations

Indicate the configuration(s) you want to use for this customization.

Figure 15-8 Configurations in the customization panel

5. Select **Other** and then click **Next**, as shown in Figure 15-9.



Configure environment

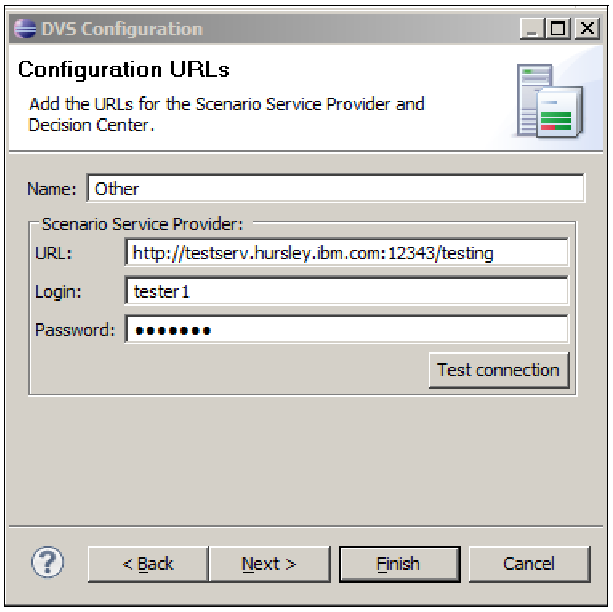
Select the application server on which to deploy the updated Scenario Service Provider and Decision Center.

Application server:

IBM WebSphere AS 7.0
IBM WebSphere AS 8.0
IBM WebSphere AS 8.5
JBoss 5
JBoss 6
Tomcat 7.0
WebLogic 10
Other

Figure 15-9 Selecting the environment to configure

6. Enter the details of your server by using the SSP port and pointing to the testing server, as shown in Figure 15-10. Click **Test connection** to check that your connection is correct, and then click **Finish** to create the configuration.



Add the URLs for the Scenario Service Provider and Decision Center.

Name: Other

Scenario Service Provider:

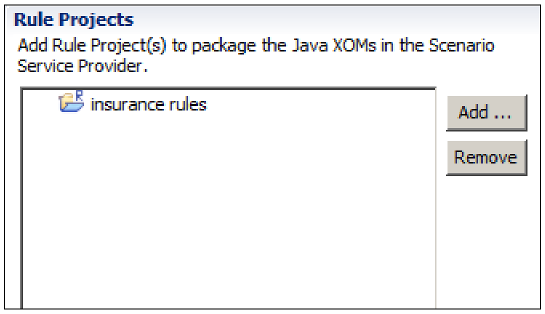
URL: http://testserv.hursley.ibm.com:12343/testing

Login: tester1

Password: •••••••

Figure 15-10 Entering the URL for the DVS configuration

7. As with all DVS projects, you now need to add the rule project to the Rule Projects list so that it displays, as shown in Figure 15-11.

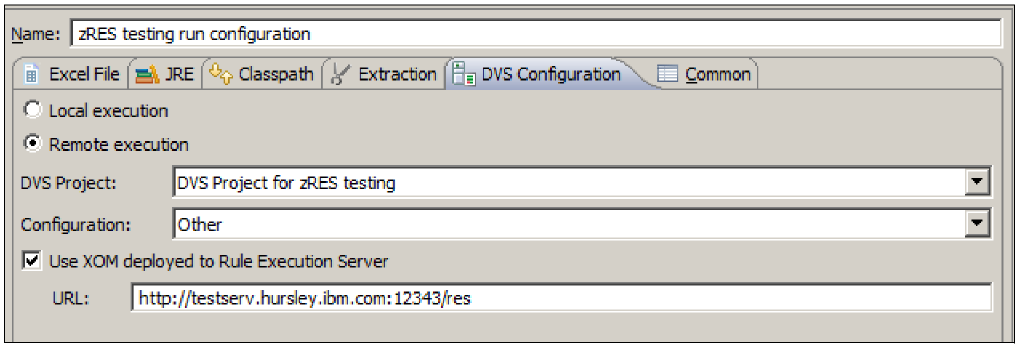


Add Rule Project(s) to package the Java XOMs in the Scenario Service Provider.

insurance rules

Figure 15-11 Adding the rule project

8. When you run your tests, after populating the tests to run, use the DVS Configuration tab to select the DVS project. Enter the details, using the SSP port in this instance, as shown in Figure 15-12. Click **Apply** to save the configuration, and click **Run** to run your tests.



Name: zRES testing run configuration

Local execution

Remote execution

DVS Project: DVS Project for zRES testing

Configuration: Other

Use XOM deployed to Rule Execution Server

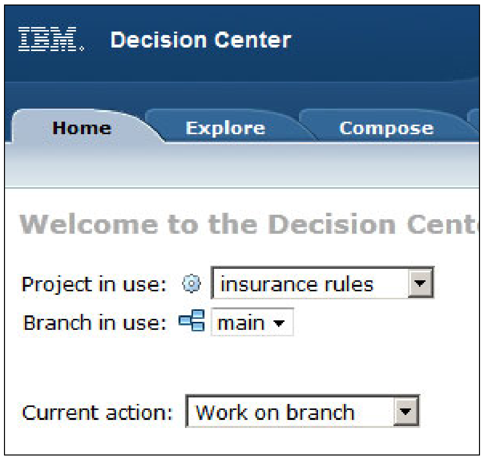
URL: http://testserv.hursley.ibm.com:12343/res

Figure 15-12 Setting up the run configuration

15.1.2 Running from Decision Center

After you set up zRES, complete the following steps from the Decision Center:

1. Sign in to the Decision Center by using the administrator user name. On the Home tab, select the rule project that you want to test, as shown in Figure 15-13.



Welcome to the Decision Cent

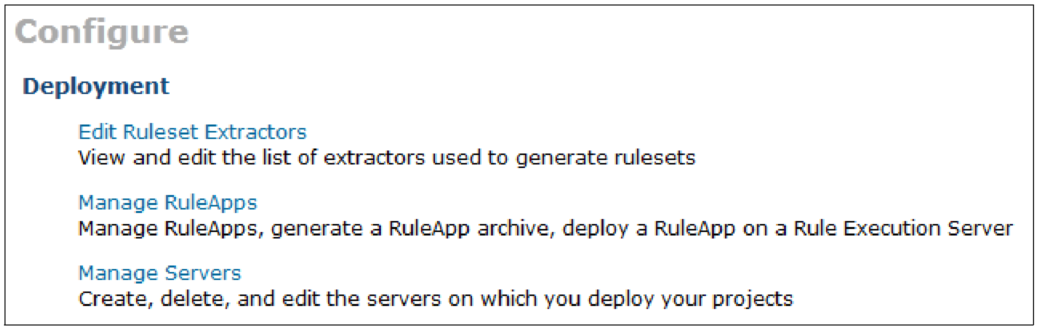
Project in use: insurance rules

Branch in use: main

Current action: Work on branch

Figure 15-13 Using the Decision Center to configure DVS

2. Click the **Configure** tab and select **Manage Servers** (Figure 15-14).



Configure

Deployment

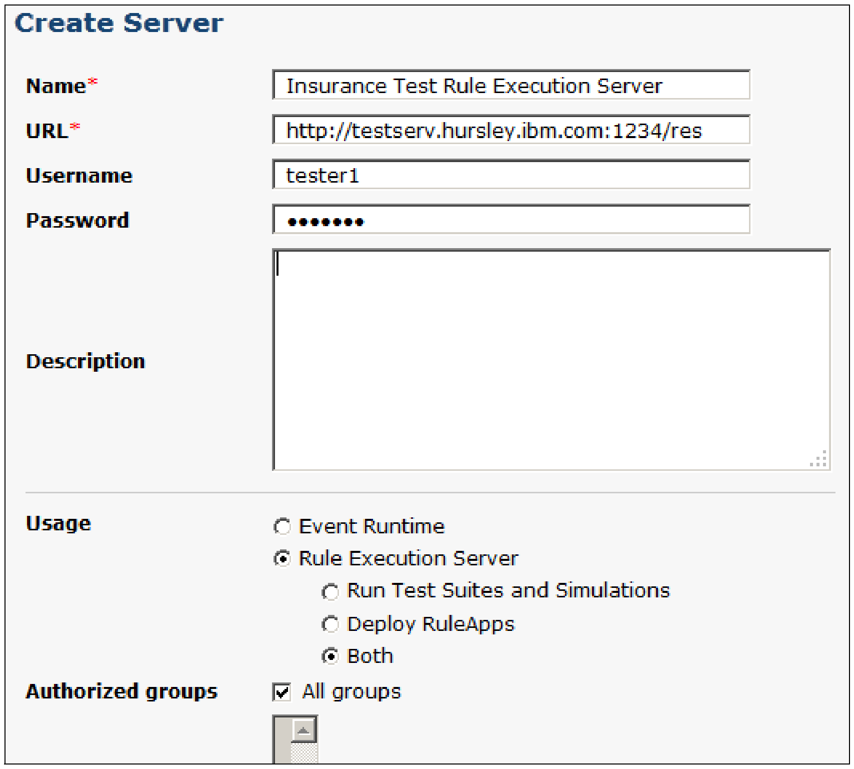
Edit Ruleset Extractors
View and edit the list of extractors used to generate rulesets

Manage RuleApps
Manage RuleApps, generate a RuleApp archive, deploy a RuleApp on a Rule Execution Server

Manage Servers
Create, delete, and edit the servers on which you deploy your projects

Figure 15-14 Selecting Manage Servers from the Configure tab

3. Click **New** to add a server. Enter the details for your server, as shown in Figure 15-15. Click **OK** to create the server. The server then appears in the server list, from where you can also check the connection.



Create Server

Name*	Insurance Test Rule Execution Server
URL*	http://testserv.hursley.ibm.com:1234/res
Username	tester1
Password	•••••••
Description	
Usage	Event Runtime [✓ Rule Execution Server] Run Test Suites and Simulations Deploy RuleApps [✓ Both]
Authorized groups	[✓ All groups]

Figure 15-15 Entering the new server details

4. Click the **Explore** tab. Click the icon that is indicated in Figure 15-16 to add a Smart folder.

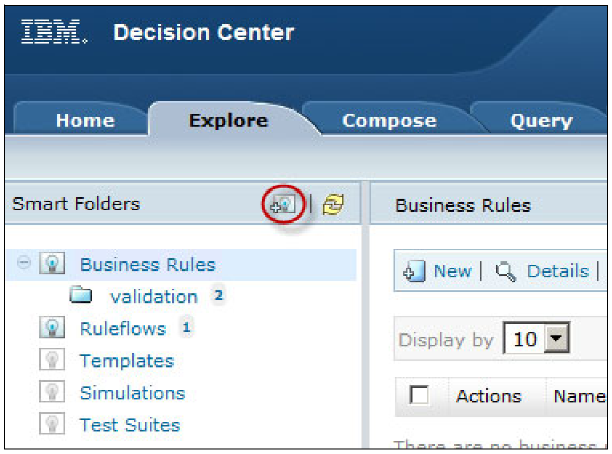
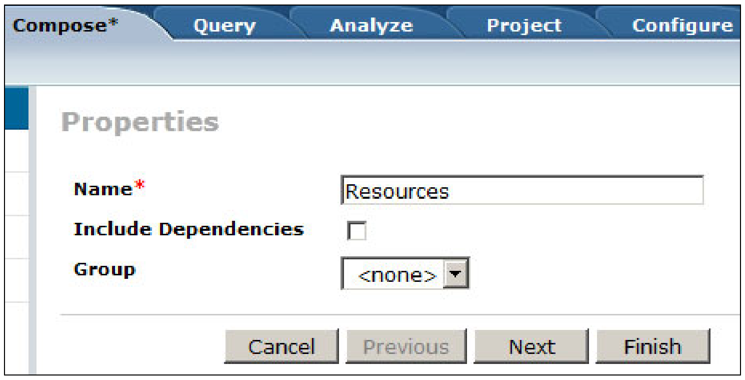


Figure 15-16 Adding a Smart folder

5. In the Smart View wizard, follow these steps:
   a. Type `Resources` as the name, as shown in Figure 15-17. Click **Next**.



Name* Resources
Include Dependencies ☐
Group <none>

Figure 15-17 Entering the name of the new Smart folder

   b. On the Query panel, click the **all business rules** link to show the possible options, as shown in Figure 15-18. Select **all resources**.

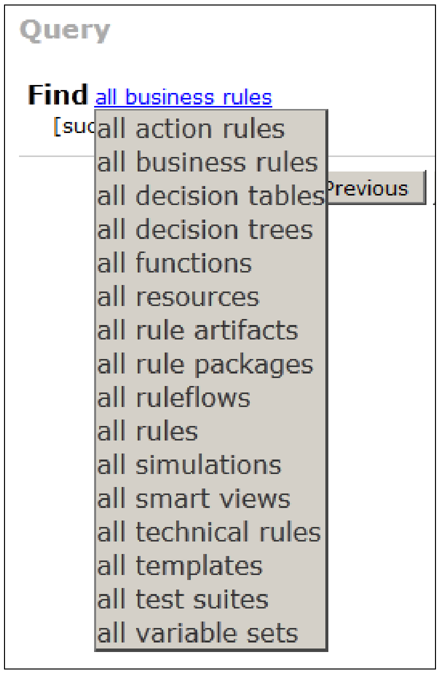
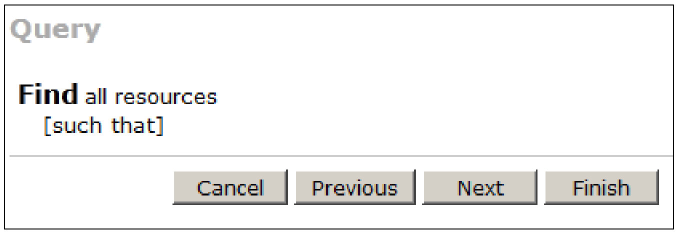


Figure 15-18 Selecting the object of the query

   c. The query then displays, as shown in Figure 15-19. Click **Next**.



Find all resources
[such that]

Figure 15-19 The selected query

d. From the list of Available Properties, select **Folder**, and click the right arrow to move it to the Displayed Properties list, as shown in Figure 15-20. Click **Finish** to create the Smart folder.

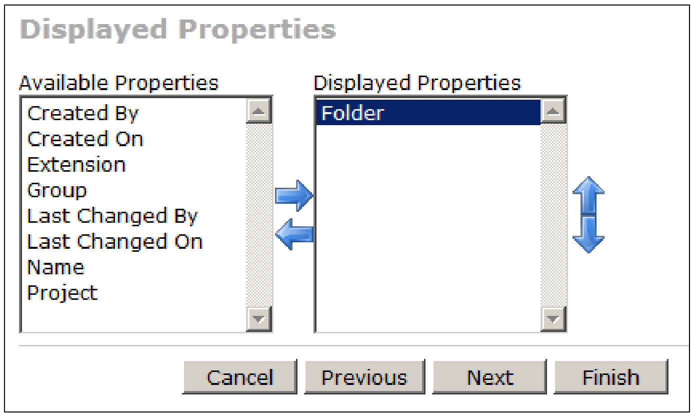


Figure 15-20 Selecting the displayed properties

6. Click the **Explore** tab to list the smart folders, as shown in Figure 15-21. The Resources node in the list represents the new smart folder. You see that two folders are already listed: `cobol` and `META-INF`.

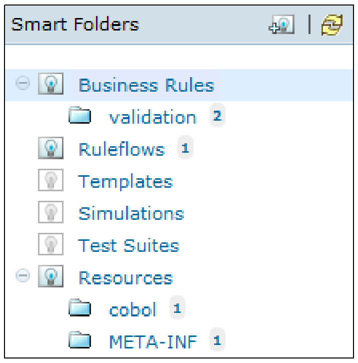


Figure 15-21 Resources smart folder displayed in the Smart Folders list

7. Click **META-INF**. You see the `deployment.xml` file published by the Rules Designer, as shown in Figure 15-22.

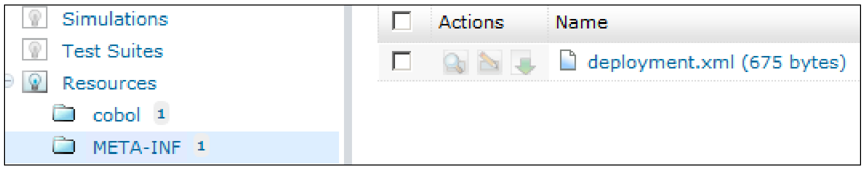


Figure 15-22 Displaying the contents of the META-INF folder

8. Click the download icon that is shown in Figure 15-23, and download the `deployment.xml` file to a suitable location where you can edit it.

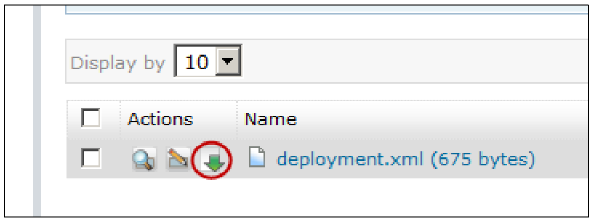
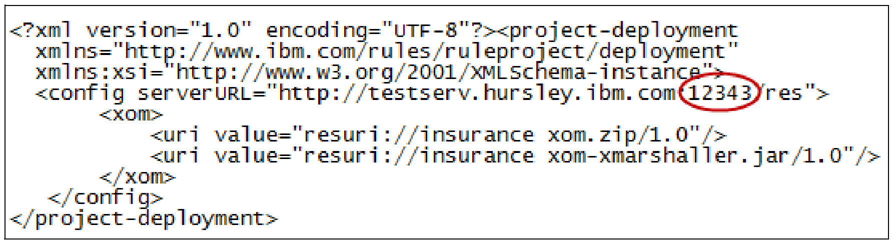


Figure 15-23 Downloading the deployment.xml file

9. Edit the `deployment.xml` file so that it points to the SSP port rather than the console port. This is the SSP port that you specified earlier. The port that requires this change is shown in Figure 15-24.



```
<?xml version="1.0" encoding="UTF-8"?><project-deployment
  xmlns="http://www.ibm.com/rules/ruleproject/deployment"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <config serverURL="http://testserv.hursley.ibm.com:12343/res">
       <xom>
           <uri value="resuri://insurance xom.zip/1.0"/>
           <uri value="resuri://insurance xom-xmarshaller.jar/1.0"/>
       </xom>
   </config>
</project-deployment>
```

Figure 15-24 Editing the deployment.xml file

10. Return to the Decision Console and re-import the file by selecting the item and clicking **Edit**, as shown in Figure 15-25.

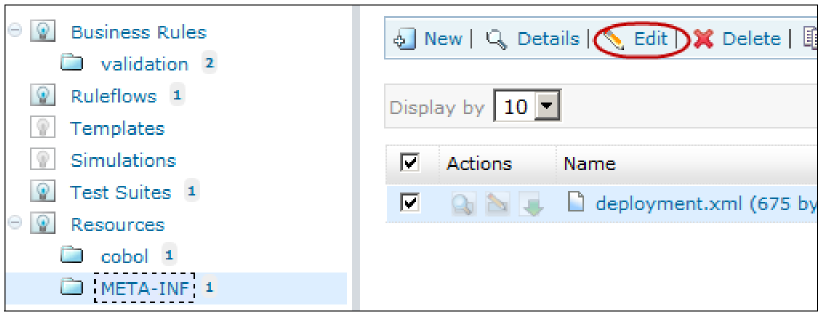
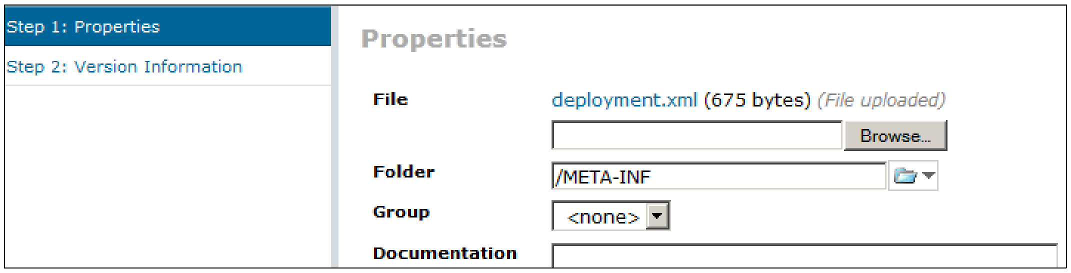


Figure 15-25 Selecting Edit

11. You can now browse for the new version of the `deployment.xml` file and upload it, as shown in Figure 15-26.



File	deployment.xml (675 bytes) *(File uploaded)*
Folder	/META-INF
Group	<none>
Documentation	

Figure 15-26 Uploading a replacement deployment.xml file

Important: If you republish the project from Rules Designer, the `deployment.xml file` might be overridden with the original version.

The managed XOM is now attached to the ruleset for the target server. Run the tests on RES on WebSphere Application Server for z/OS.

15.2 DVS using RES on WebSphere Application Server for z/OS

This section describes the necessary steps for configuring DVS when it is installed with RES on WebSphere Application Server for z/OS. You install the default SSP archive as an enterprise application within WebSphere Application Server. Subsequent updates to this archive are re-installed to replace the original.

The following configuration example assumes that you are using WebSphere Application Server V8.0 for z/OS. If you use a different version, some of the installation steps might differ slightly, but the following example is sufficient to assist you through the installation process.

15.2.1 Running from Rules Designer

Complete the following steps in this section from the Rules Designer.

Deploying the XOM to zRES

Complete the following steps to deploy the XOM to zRES:

1. In Rules Designer, right-click the rules project, and select **Decision Validation Services** → **Deploy XOM for Remote DVS Testing**, as shown in Figure 15-27.

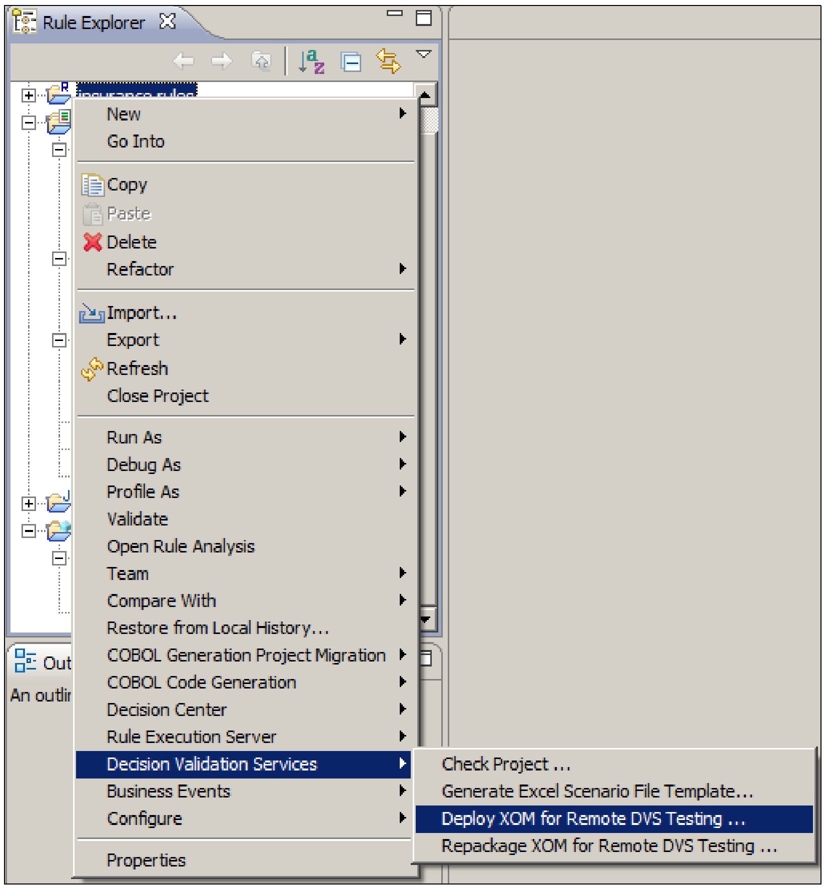
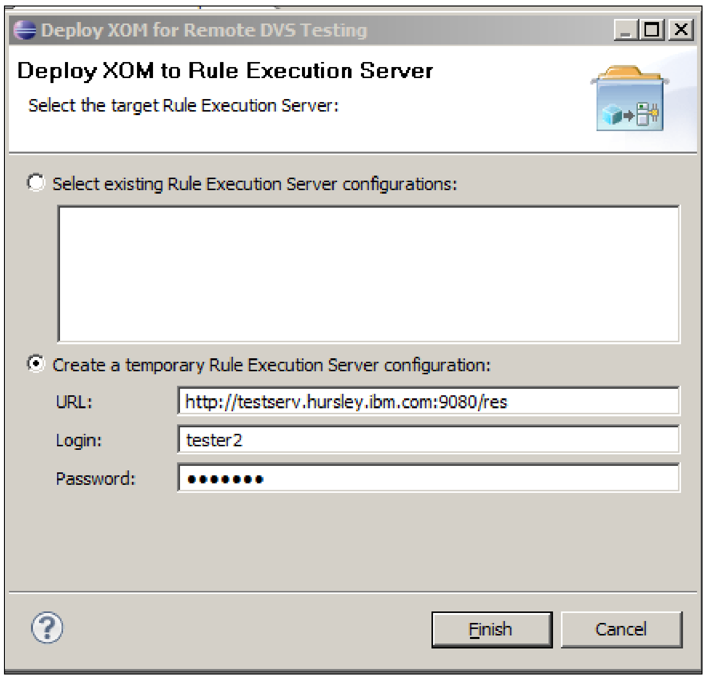


Figure 15-27 Selecting to deploy XOM for remote DVS testing

2. Populate the fields with the details of your RES on WebSphere Application Server for z/OS, as shown in Figure 15-28. Click **Finish** to deploy the XOM.


Deploy XOM for Remote DVS Testing
Deploy XOM to Rule Execution Server
Select the target Rule Execution Server:
Select existing Rule Execution Server configurations:
Create a temporary Rule Execution Server configuration:
Finish
Cancel

URL: http://testserv.hursley.ibm.com:9080/res
Login: tester2
Password: •••••••

Figure 15-28 Populating the fields to deploy to a remote server

Setting up the DVS project

Complete the following steps to configure the DVS project:

1. Right-click and select **New**. Select the **DVS Project** wizard, as shown in Figure 15-29. Click **Next** to open the wizard.

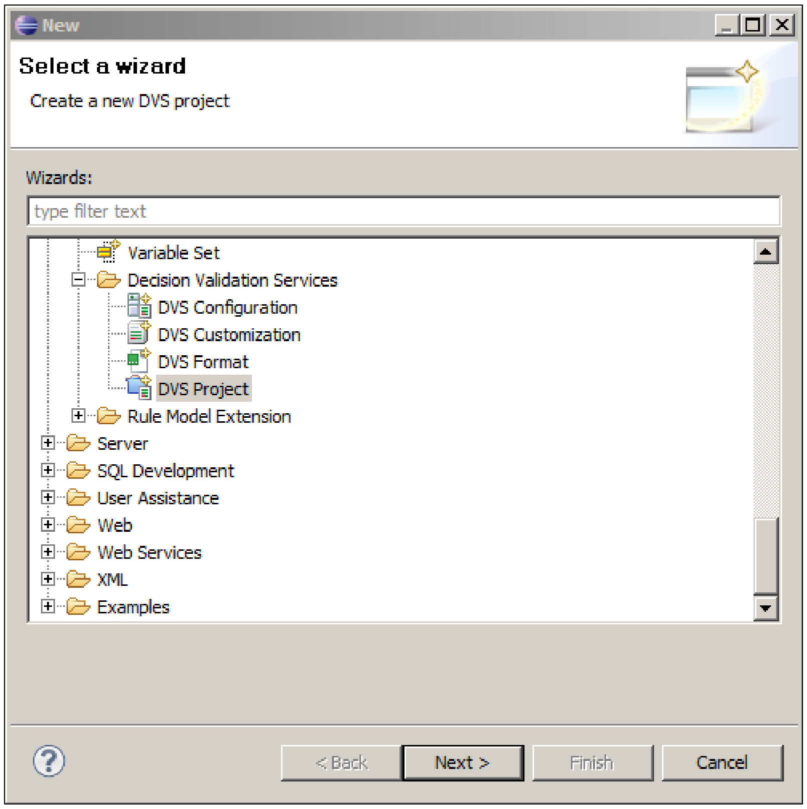
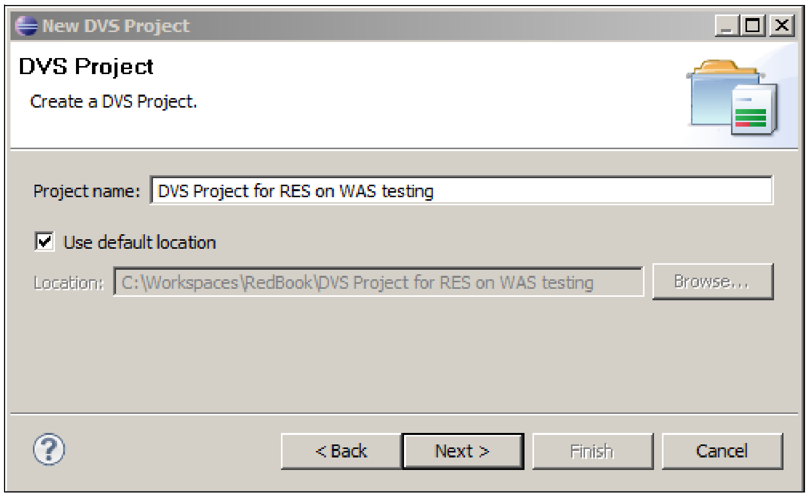


Figure 15-29 Selecting the DVS project wizard

2. Enter your project name and click **Next**, as shown in Figure 15-30.



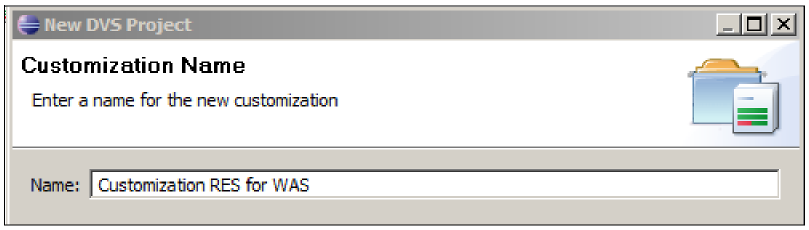
Project name: DVS Project for RES on WAS testing

☑ Use default location

Location: C:\Workspaces\RedBook\DVS Project for RES on WAS testing

Figure 15-30 Creating the DVS project and specifying the project name

3. Enter the name for the customization and click **Finish**, as shown in Figure 15-31.



Name: Customization RES for WAS

Figure 15-31 Creating the DVS project and specifying the customization name

4. Create a new configuration by clicking **Create**, as shown in Figure 15-32.

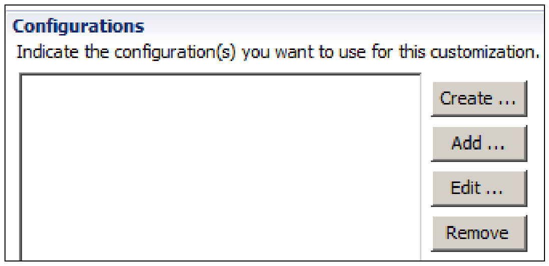

Configurations
Indicate the configuration(s) you want to use for this customization.
Create ...
Add ...
Edit ...
Remove

Figure 15-32 Configurations in the customization panel

5. Select **IBM WebSphere AS 8.0** and click **Next**, as shown in Figure 15-33.

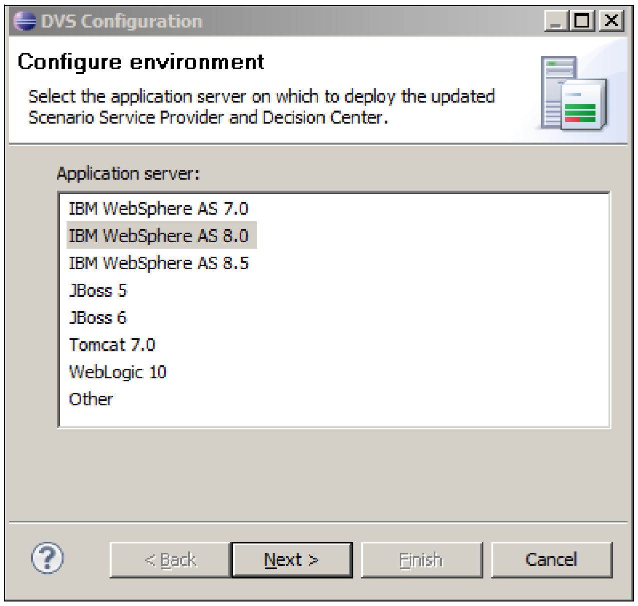
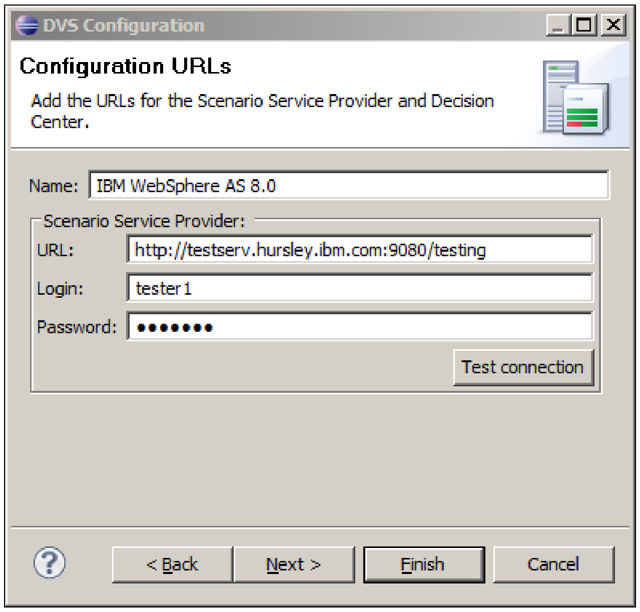


Figure 15-33 Selecting the environment for your WebSphere Application Server

6. Enter the details of your server by using the SSP port and pointing to the testing server, as shown in Figure 15-34. Click **Test connection** to check that your connection is correct, and click **Finish** to create the configuration.



Configuration URLs

Add the URLs for the Scenario Service Provider and Decision Center.

Name: IBM WebSphere AS 8.0

Scenario Service Provider:

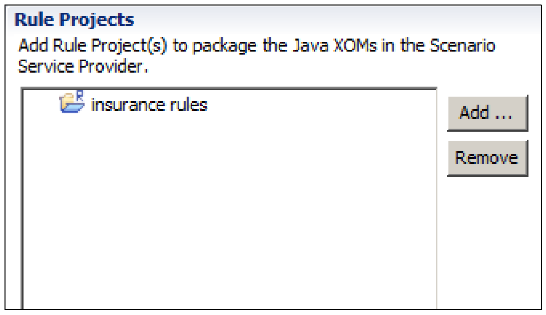
URL: http://testserv.hursley.ibm.com:9080/testing

Login: tester1

Password: •••••••

Figure 15-34   Entering the URL for the DVS configuration

7. As with all DVS projects, you need to add the rule project to the Rule Projects list so that the project displays, as shown in Figure 15-35.



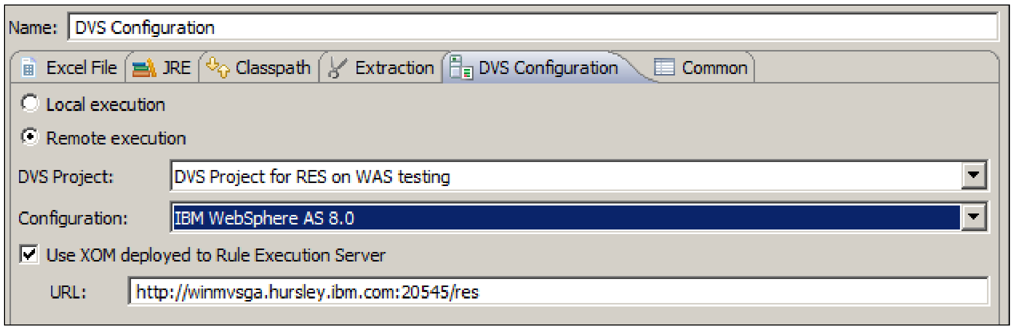
Rule Projects

Add Rule Project(s) to package the Java XOMs in the Scenario Service Provider.

insurance rules

Figure 15-35   Adding the rule project

8. When you run your tests, after populating the tests to run, use the DVS Configuration tab to select the DVS project. Enter the details, as shown in Figure 15-36. Click **Apply** to save the configuration, and click **Run** to run your tests.


Name: DVS Configuration

( ) Local execution
(●) Remote execution

DVS Project: DVS Project for RES on WAS testing

Configuration: IBM WebSphere AS 8.0

[✓] Use XOM deployed to Rule Execution Server

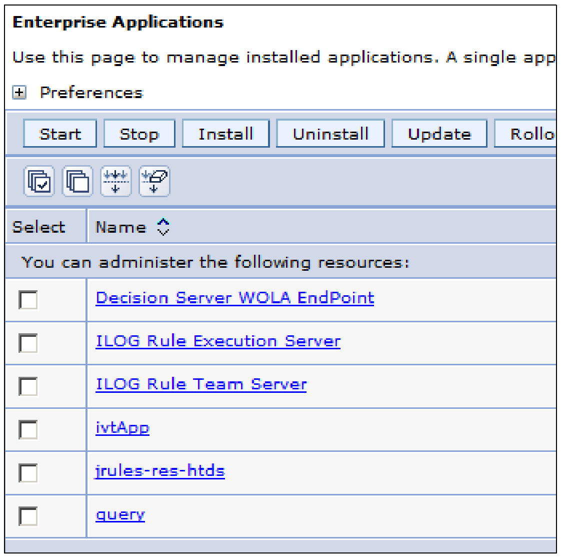
URL: http://winmvsga.hursley.ibm.com:20545/res

Figure 15-36 Setting up the run configuration

15.2.2 Running from the Decision Center

Complete the following steps from the Decision Center:

1. Start the console of WebSphere Application Server for z/OS, and select **Applications** → **Application Types** → **WebSphere Enterprise Applications**. Click **Install** to install a new application (Figure 15-37).


Enterprise Applications

Use this page to manage installed applications. A single app

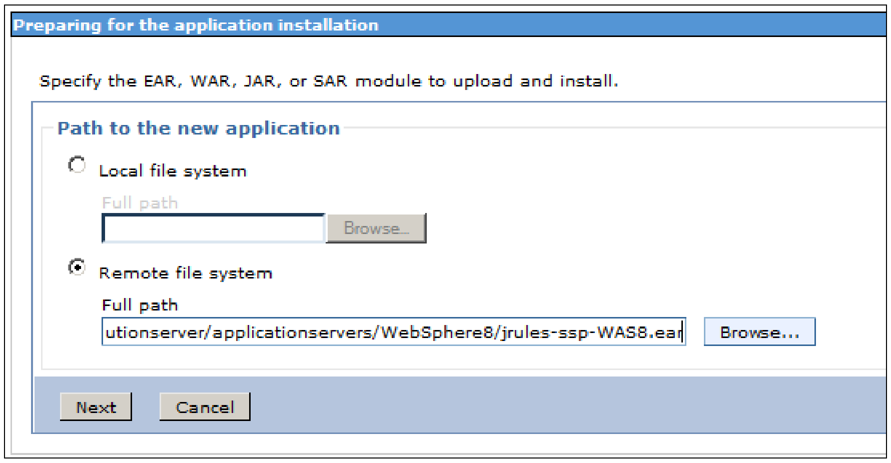
Select	Name
You can administer the following resources:	
☐	Decision Server WOLA EndPoint
☐	ILOG Rule Execution Server
☐	ILOG Rule Team Server
☐	ivtApp
☐	jrules-res-htds
☐	query

Figure 15-37 WebSphere Enterprise Applications panel

2. The default SSP file is in the following directory:

   `<installation>/executionserver/applicationservers/WebSphere8/jrules-ssp-WAS8.ear`

   See Figure 15-38. Click **Next**.



Preparing for the application installation

Specify the EAR, WAR, JAR, or SAR module to upload and install.

Path to the new application

( ) Local file system

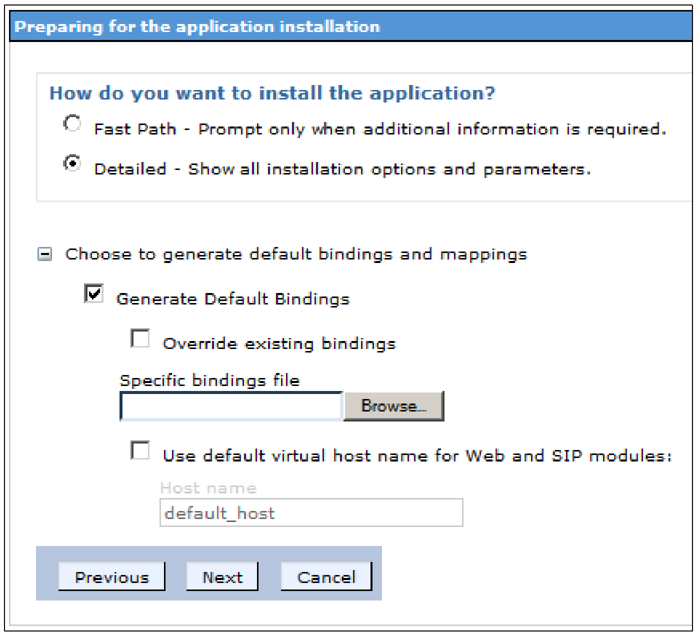
Full path

(●) Remote file system

Full path: utionserver/applicationservers/WebSphere8/jrules-ssp-WAS8.ear

Figure 15-38 Selecting the SSP file

3. On the Preparing for the application installation panel, select **Detailed - Show all installation options and parameters**. Expand **Choose to generate default bindings and mappings** and check **Generate Default Bindings**, as shown in Figure 15-39. Click **Next**.



Preparing for the application installation

How do you want to install the application?

( ) Fast Path - Prompt only when additional information is required.

(●) Detailed - Show all installation options and parameters.

Choose to generate default bindings and mappings

[✓] Generate Default Bindings

[ ] Override existing bindings

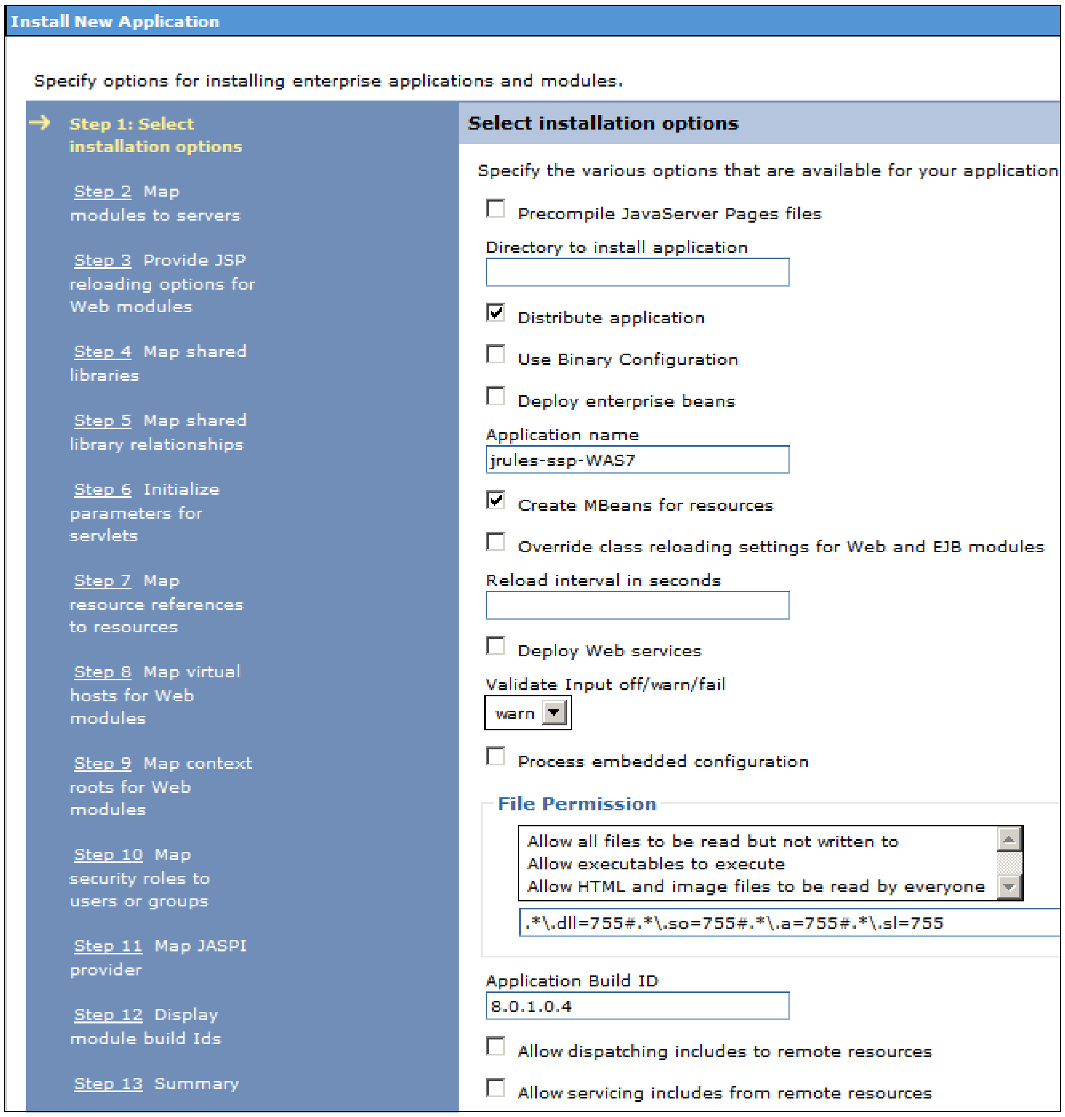
Specific bindings file

[ ] Use default virtual host name for Web and SIP modules:

Host name: default_host

Figure 15-39 Preparing for the application installation.

4. Click **Continue** to accept the security warning to display the Install New Application panel, as shown in Figure 15-40.


Install New Application

Specify options for installing enterprise applications and modules.

→ Step 1: Select installation options

Step 2 Map modules to servers

Step 3 Provide JSP reloading options for Web modules

Step 4 Map shared libraries

Step 5 Map shared library relationships

Step 6 Initialize parameters for servlets

Step 7 Map resource references to resources

Step 8 Map virtual hosts for Web modules

Step 9 Map context roots for Web modules

Step 10 Map security roles to users or groups

Step 11 Map JASPI provider

Step 12 Display module build Ids

Step 13 Summary

Select installation options

Specify the various options that are available for your application

☐ Precompile JavaServer Pages files

Directory to install application

☑ Distribute application

☐ Use Binary Configuration

☐ Deploy enterprise beans

Application name: jrules-ssp-WAS7

☑ Create MBeans for resources

☐ Override class reloading settings for Web and EJB modules

Reload interval in seconds

☐ Deploy Web services

Validate Input off/warn/fail: warn

☐ Process embedded configuration

File Permission

Allow all files to be read but not written to
Allow executables to execute
Allow HTML and image files to be read by everyone

.*\.dll=755#.*\.so=755#.*\.a=755#.*\.sl=755

Application Build ID: 8.0.1.0.4

☐ Allow dispatching includes to remote resources

☐ Allow servicing includes from remote resources

Figure 15-40 Install New Application: Step 1 Select installation options

5. Click **Next** to step through all the steps until you reach step 10, which is shown in Figure 15-41. If you are not using a federated repository, skip to step 6 on page 268. If you are using a federated repository for security, you need to map the groups:

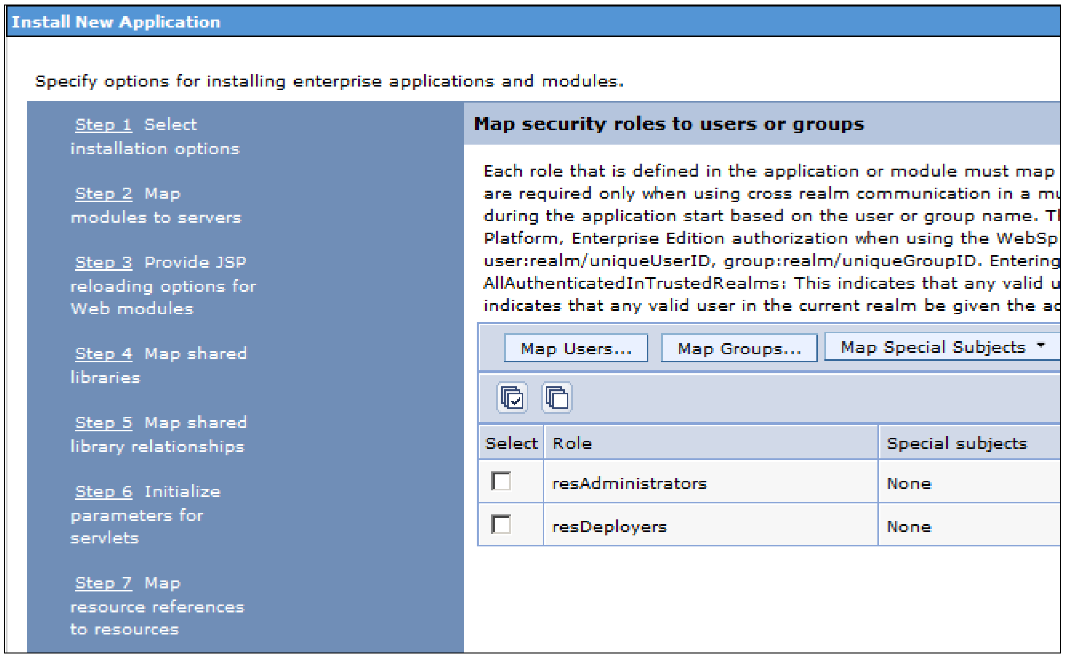
   a. Select the **resAdministrators** check box and click **Map Groups**.

   b. Click **Search**.

   c. Select **resAdministrators** in the Available list and click the right arrow to move it to the Selected column. Click **OK** to return to the Map security roles to users or groups panel.

   d. Repeat these steps for the **resDeployers** group.

   Click **Next**.


Install New Application

Specify options for installing enterprise applications and modules.

Step 1 Select installation options
Step 2 Map modules to servers
Step 3 Provide JSP reloading options for Web modules
Step 4 Map shared libraries
Step 5 Map shared library relationships
Step 6 Initialize parameters for servlets
Step 7 Map resource references to resources

Map security roles to users or groups

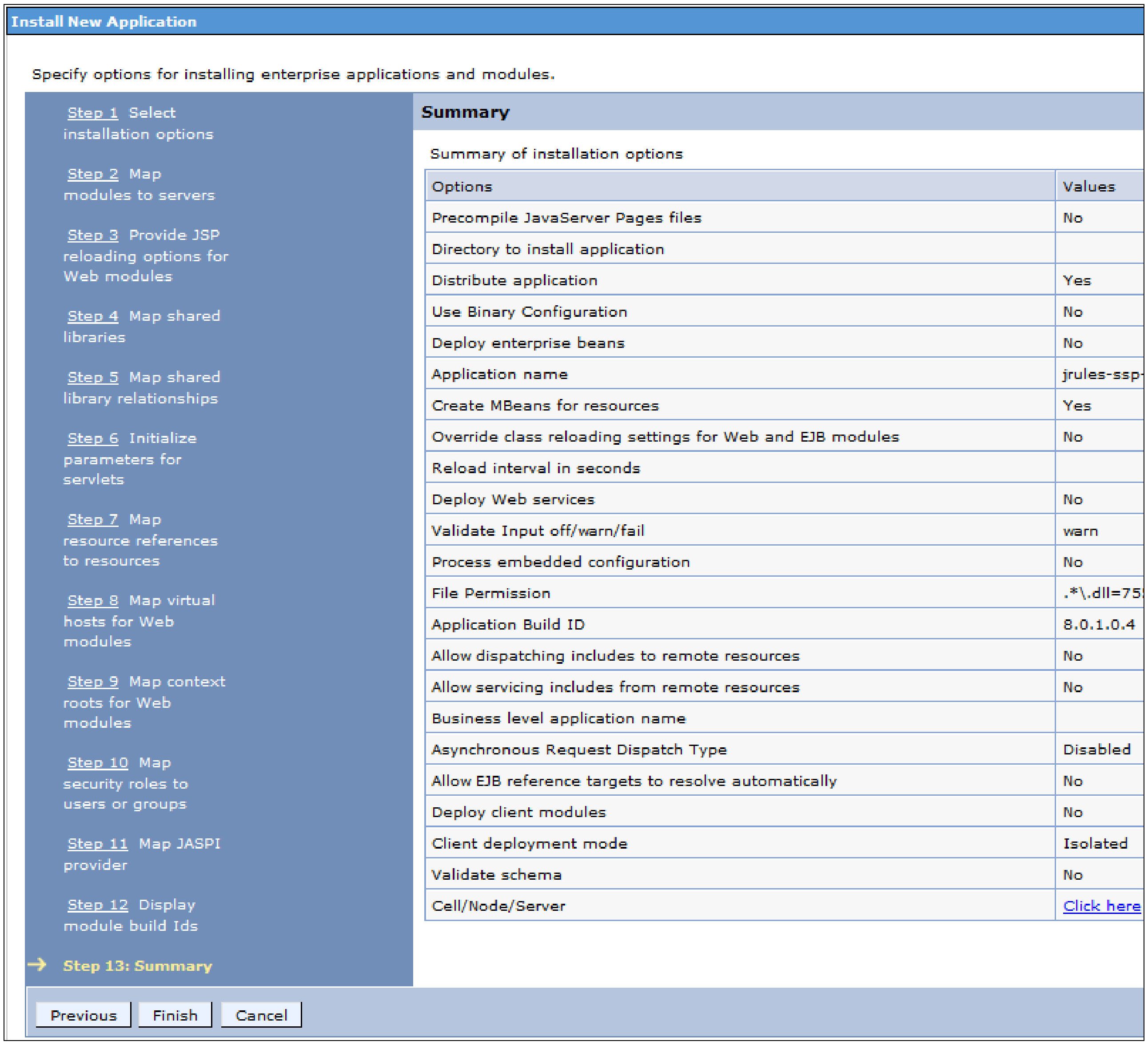
Each role that is defined in the application or module must map
are required only when using cross realm communication in a mu
during the application start based on the user or group name. Th
Platform, Enterprise Edition authorization when using the WebSp
user:realm/uniqueUserID, group:realm/uniqueGroupID. Entering
AllAuthenticatedInTrustedRealms: This indicates that any valid u
indicates that any valid user in the current realm be given the ac

Map Users... | Map Groups... | Map Special Subjects

Select	Role	Special subjects
☐	resAdministrators	None
☐	resDeployers	None

Figure 15-41 Install New Application: Step 10 Map security roles to users or groups

6. Click **Next** until the summary is displayed in step 13, as shown in Figure 15-42. Click **Finish**. When prompted, click **Save** to save the changes.


Install New Application
Step 1 Select installation options
Step 2 Map modules to servers
Step 3 Provide JSP reloading options for Web modules
Step 4 Map shared libraries
Step 5 Map shared library relationships
Step 6 Initialize parameters for servlets
Step 7 Map resource references to resources
Step 8 Map virtual hosts for Web modules
Step 9 Map context roots for Web modules
Step 10 Map security roles to users or groups
Step 11 Map JASPI provider
Step 12 Display module build Ids
Step 13: Summary
Previous
Finish
Cancel

Specify options for installing enterprise applications and modules.

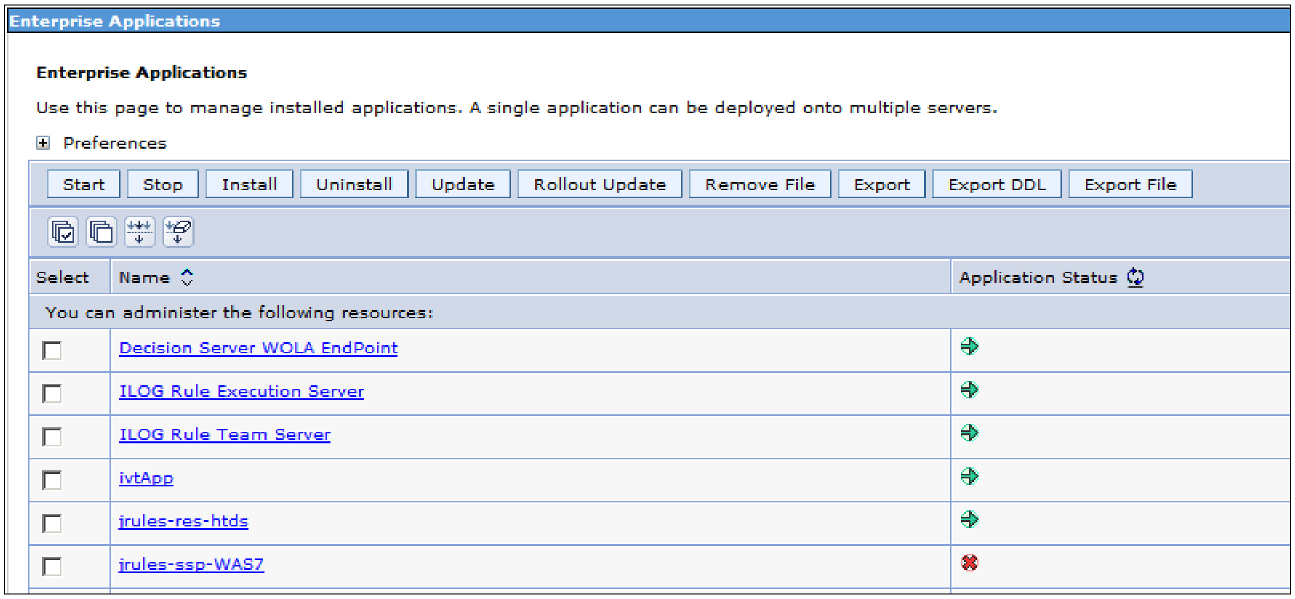
Summary

Summary of installation options

Options	Values
Precompile JavaServer Pages files	No
Directory to install application	
Distribute application	Yes
Use Binary Configuration	No
Deploy enterprise beans	No
Application name	jrules-ssp-
Create MBeans for resources	Yes
Override class reloading settings for Web and EJB modules	No
Reload interval in seconds	
Deploy Web services	No
Validate Input off/warn/fail	warn
Process embedded configuration	No
File Permission	.*\.dll=75
Application Build ID	8.0.1.0.4
Allow dispatching includes to remote resources	No
Allow servicing includes from remote resources	No
Business level application name	
Asynchronous Request Dispatch Type	Disabled
Allow EJB reference targets to resolve automatically	No
Deploy client modules	No
Client deployment mode	Isolated
Validate schema	No
Cell/Node/Server	Click here

Figure 15-42 Install New Application: Step 13 Summary

7. Select **Applications** → **Application Types** → **WebSphere Enterprise Applications**. Click the newly installed `jrules-ssp-WAS7` from the resulting list, as shown in Figure 15-43.



Enterprise Applications

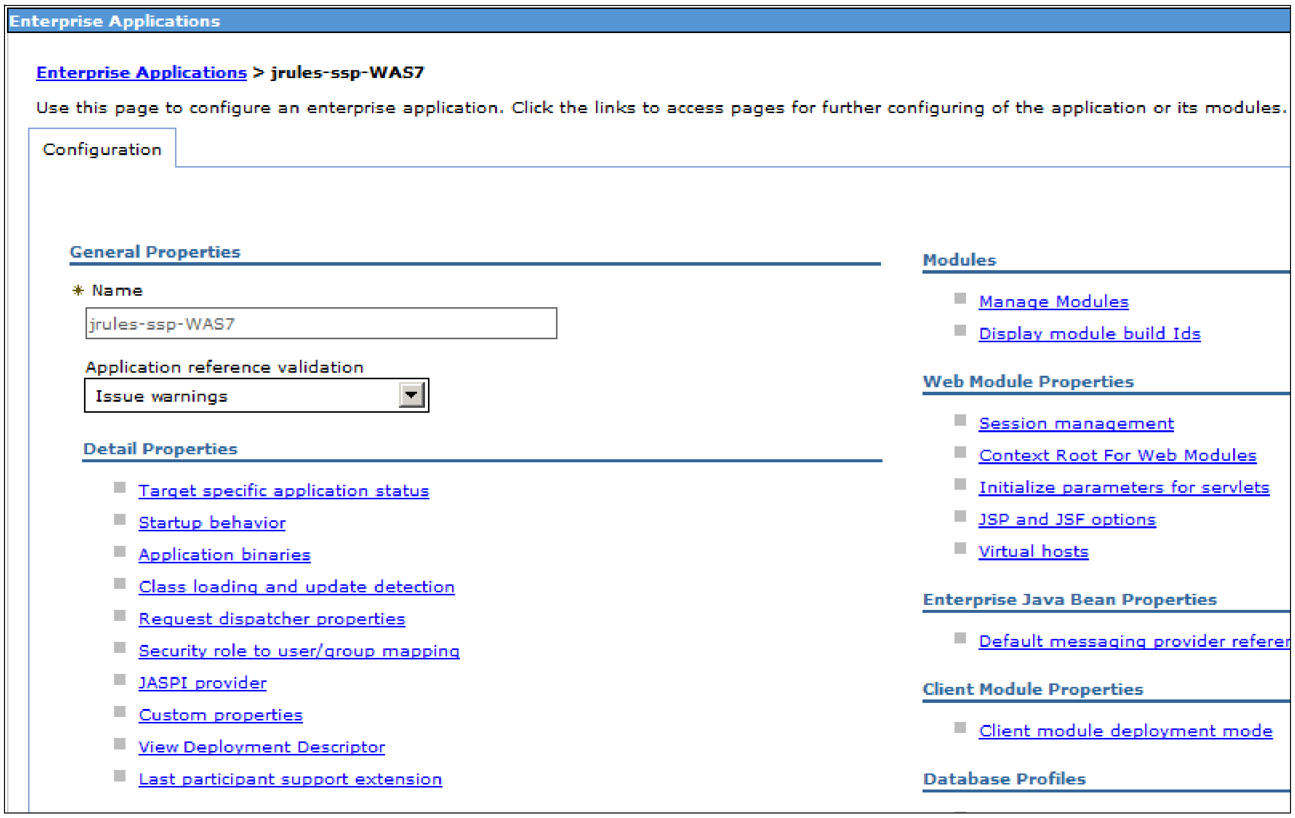
Use this page to manage installed applications. A single application can be deployed onto multiple servers.

⊞ Preferences

Select	Name	Application Status
You can administer the following resources:		
☐	Decision Server WOLA EndPoint	➔
☐	ILOG Rule Execution Server	➔
☐	ILOG Rule Team Server	➔
☐	ivtApp	➔
☐	jrules-res-htds	➔
☐	jrules-ssp-WAS7	✖

Figure 15-43 Enterprise applications with installed SSP

8. On the resulting window (Figure 15-44), under the Modules list, select **Manage Modules**.



Enterprise Applications > jrules-ssp-WAS7

Use this page to configure an enterprise application. Click the links to access pages for further configuring of the application or its modules.

General Properties

* Name
jrules-ssp-WAS7

Application reference validation
Issue warnings

Detail Properties
- Target specific application status
- Startup behavior
- Application binaries
- Class loading and update detection
- Request dispatcher properties
- Security role to user/group mapping
- JASPI provider
- Custom properties
- View Deployment Descriptor
- Last participant support extension

Modules
- Manage Modules
- Display module build Ids

Web Module Properties
- Session management
- Context Root For Web Modules
- Initialize parameters for servlets
- JSP and JSF options
- Virtual hosts

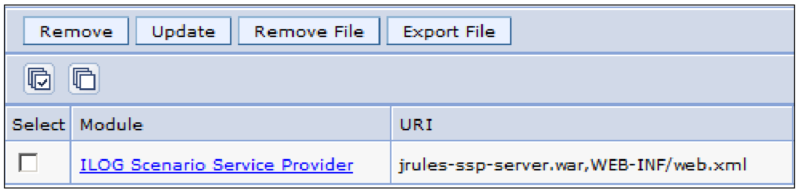
Enterprise Java Bean Properties
- Default messaging provider refere

Client Module Properties
- Client module deployment mode

Database Profiles

Figure 15-44 Displaying the jrules-ssp-WAS7 application

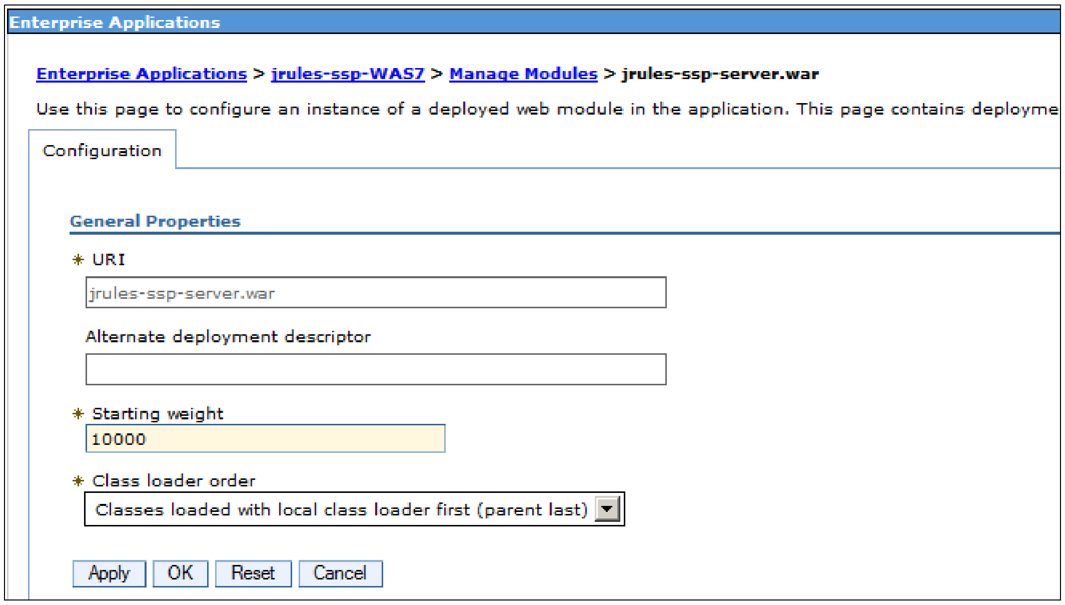
9. In the Module list, select **ILOG Scenario Service Provider**, as shown in Figure 15-45.



Select	Module	URI
☐	ILOG Scenario Service Provider	jrules-ssp-server.war,WEB-INF/web.xml

Figure 15-45 Selecting the IBM ILOG® Scenario Service Provider

10. In the resulting window, change the Class loader order field to **Classes loaded with local class loader first (parent last)**, as shown in Figure 15-46. Click **OK**, and when prompted, save the changes.



Enterprise Applications > jrules-ssp-WAS7 > Manage Modules > jrules-ssp-server.war

Use this page to configure an instance of a deployed web module in the application. This page contains deployme

* URI
jrules-ssp-server.war

Alternate deployment descriptor

* Starting weight
10000

* Class loader order
Classes loaded with local class loader first (parent last)

Figure 15-46 Selecting the class loader order

11. Select **Applications** → **Application Types** → **WebSphere Enterprise Applications**. Select **jrules-ssp-WAS7** and click **Start** to start the application.

Tip: To work with an updated version, you uninstall the original application from the WebSphere Enterprise Applications menu and then repeat this procedure to install the new application.

15.3 Service scenario provider and key performance indicator architecture

This section provides more details about the service scenario provider and the key performance indicator architecture that was first described in 6.4.4, "Runtime components" on page 133.

15.3.1 Runtime client API

The Java runtime client API that is provided with Operational Decision Manager allows a developer to write a Java application that runs an existing scenario suite either locally (in the same Java virtual machine (JVM), without deploying an SSP module) or against a deployed SSP. A typical use case is to write a batch that runs test suites and simulations every night and generates a complete execution report.

The main entry point in the runtime client API for running a scenario suite against a deployed SSP is the package `ilog.rules.dvs.client`. This package provides extensive online documentation with code samples that explain how to use it. For more information, see the *Package ilog.rules.dvs.client* topic in the Decision Server V8.0.1 Information Center:

http://pic.dhe.ibm.com/infocenter/dmanager/v8r0m1/index.jsp?topic=%2Fcom.ibm.wodm.dserver.rules.ref.designer%2Fhtml%2Fapi%2Fhtml%2Filog%2Frules%2Fdvs%2Fclient%2Fpackage-summary.html

For running a scenario suite locally, the class `ilog.rules.dvs.ssp.IlrLocalSSPService` is the main entry point. This class can be used with the previous code sample to replace the SSP service implementation that is returned by the SSP service factory.

15.3.2 Service scenario provider

The scenario provider is specified in the scenario suite format. The *scenario provider* is the low-level execution component that provides the scenario suite data in a suitable format for execution by the rule engine. The scenario provider provides the test component in the case of a test suite (see Figure 15-47 on page 272).

Unless you want to use an existing data store (a relational database or a set of files, for example) to specify your scenario suite data, you do not write a custom scenario provider. You can reuse `ilog.rules.dvs.core.scenarioproviders.IlrExcel2003ScenarioProvider`, which is the Excel scenario provider, in all your scenario suite formats.

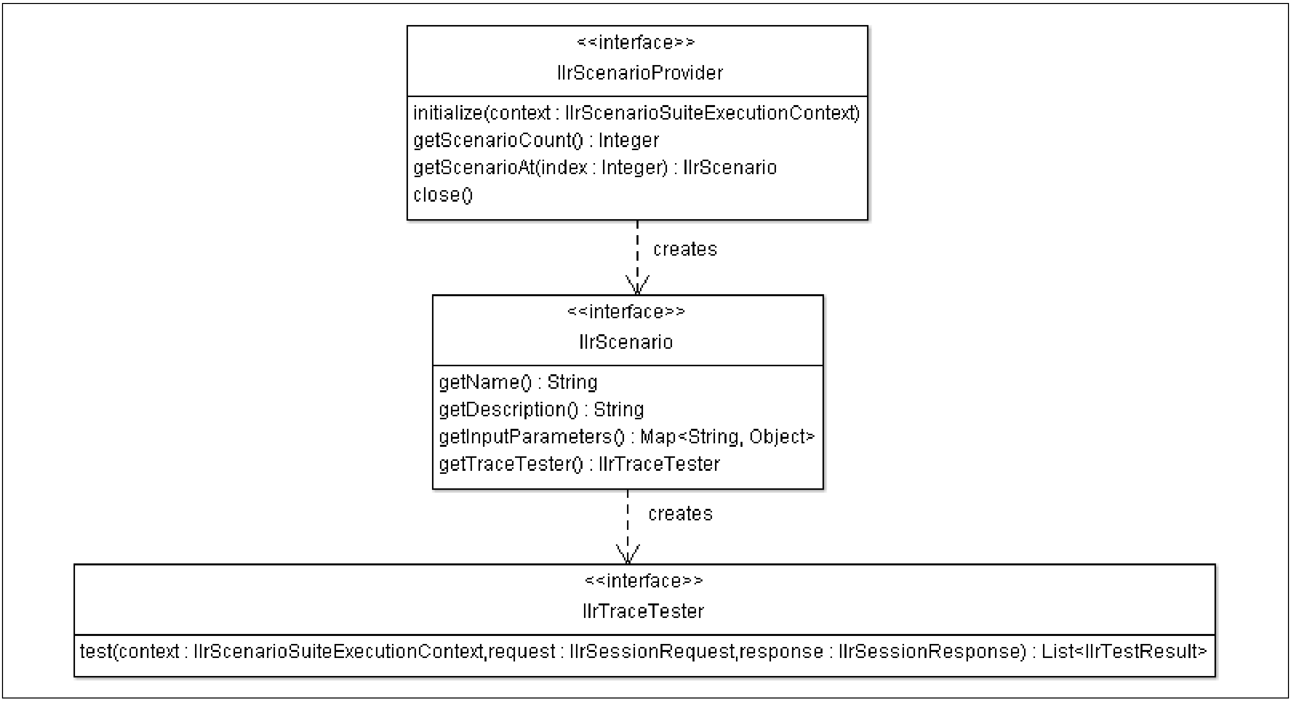
The *Testing in Decision Center* topic in the Decision Server V8.0.1 Information Center includes a generic sample that explains how to retrieve scenario data from a relational database:

http://pic.dhe.ibm.com/infocenter/dmanager/v8r0m1/index.jsp?topic=%2Fcom.ibm.wodm.dcenter.samples%2Ftopics%2Ftpc_tstesting_toc.html

A more specific code sample is provided that demonstrates authoring a custom scenario provider that accesses a VSAM data store:

http://pic.dhe.ibm.com/infocenter/dmanager/v8r0m1/index.jsp?topic=%2Fcom.ibm.wodm.zos.dcenter.samples%2Ftopics%2Fsmp_custprovider_vsam.html

Figure 15-47 shows the scenario provider Java API.



<<interface>>
IlrScenarioProvider
initialize(context : IlrScenarioSuiteExecutionContext)
getScenarioCount() : Integer
getScenarioAt(index : Integer) : IlrScenario
close()
creates
<<interface>>
IlrScenario
getName() : String
getDescription() : String
getInputParameters() : Map<String, Object>
getTraceTester() : IlrTraceTester
creates
<<interface>>
IlrTraceTester
test(context : IlrScenarioSuiteExecutionContext,request : IlrSessionRequest,response : IlrSessionResponse) : List<IlrTestResult>

Figure 15-47 The scenario provider Java API

Scenario provider

A scenario provider is a Java class. It implements the `ilog.rules.dvs.core.IlrScenarioProvider` interface from the testing and simulation API. This component performs the following actions:

- Initialization

  An initialization method that is provided by the component is called by the SSP before the execution of a scenario suite. This method can access a context that contains the data that the user entered in the Decision Center GUI (or in a Rule Designer launch configuration) when the user created the scenario suite. This way, the scenario that was provided can use this method to access the corresponding data store.

  For example, the Excel scenario provider that is provided with the Excel scenario suite formats uses this context to retrieve the Microsoft Excel file that was uploaded by the Decision Center user (or that was selected in the Rule Designer launch configuration) when the user created the scenario suite.

- Scenario count

  After initialization, the component is expected to be able to return the total number of scenarios in the suite when requested by the SSP that is running it.

- Scenario retrieval

  After initialization, the component is expected to be able to return any scenario from the suite. Each scenario is identified by its index in the suite.

- Resource cleanup

  When requested, the component is expected to clean up any internal or external resource that it uses, such as a connection to a database.

  The Decision Center public API defines a component that provides a GUI to a business user to initialize the scenario provider with data that is entered when a scenario suite is created. This is `log.rules.teamserver.web.components.renderer.IlrScenarioSuiteResourcesRenderer`. A sample implementation of this component is provided in 15.4, “Example” on page 275.

Scenario

Every scenario in the suite is modeled by a scenario component, which is a Java class that implements the `ilog.rules.dvs.core.IlrScenario` interface from the testing and simulation API.

Primarily identified by its index in the scenario suite, a scenario also carries a name and description that can be useful for identifying its purpose within this scenario suite. Both the name and description are provided by the scenario component. In addition to this identification data, the scenario component provides this important information:

- The map of Java input parameters to be used by the rule engine for the execution of the ruleset under test or simulation
- The trace tester component that verifies the execution result if the scenario is a test scenario

Trace tester

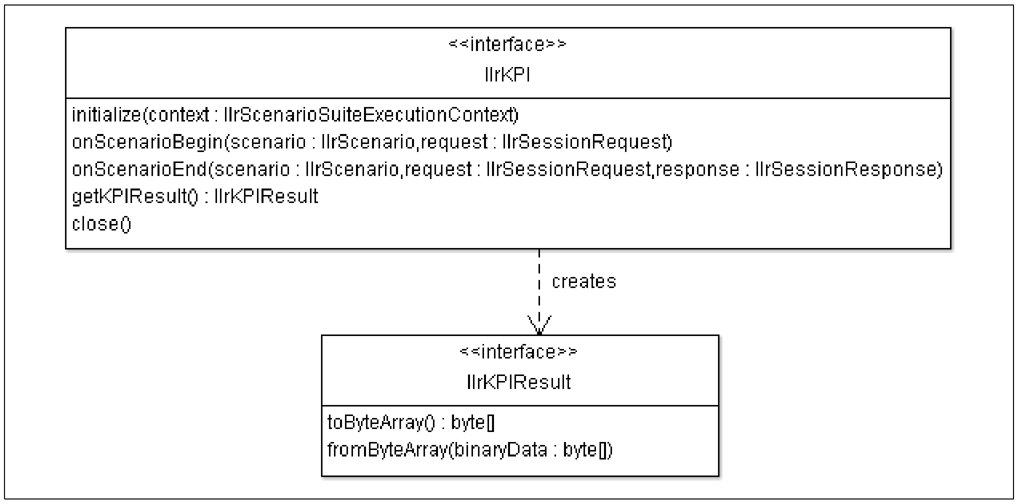
The trace tester component is a Java class that implements the `ilog.rules.dvs.core.IlrTraceTester` interface. When the SSP makes a request, the trace tester component performs tests on the execution result of a scenario to verify that this result conforms to the user expectation.

The trace tester component is created by the scenario component, which is created by the scenario provider component. The test specification is generally retrieved at the initialization of the scenario provider, as part of the scenario data.

15.3.3 Key performance indicator

If authoring a custom scenario provider is not mandatory (unless you plan to use another data store than Microsoft Excel to define your scenario suites), authoring a custom KPI is mandatory if you want to run a simulation.

The test and simulation KPI API only defines two components, KPI and KPI result, that you can use for all your simulation needs. These components are shown in Figure 15-48.



<<interface>>
IlrKPI
initialize(context : IlrScenarioSuiteExecutionContext)
onScenarioBegin(scenario : IlrScenario,request : IlrSessionRequest)
onScenarioEnd(scenario : IlrScenario,request : IlrSessionRequest,response : IlrSessionResponse)
getKPIResult() : IlrKPIResult
close()
creates
<<interface>>
IlrKPIResult
toByteArray() : byte[]
fromByteArray(binaryData : byte[])

Figure 15-48 The test and simulation KPI Java API

The KPI component is a Java class that implements the `ilog.rules.dvs.core.IlrKPI` interface. This component implements the following behavior:

- Initialization

  An initialization method that is provided by the KPI component is called by the SSP before the execution of a scenario suite. This method has access to the same context as the scenario provider. This capability can be useful to retrieve information that is entered by the user when the user created the simulation to share a single KPI implementation/scenario suite format between multiple rule projects, using user data to decide how the KPI calculation needs to be performed for the current simulation.

- Beginning of a scenario

  The KPI component is notified by the SSP each time that a scenario is going to be executed by the rule engine. To perform its calculation, if needed at this stage of the processing, the KPI component has access to the scenario that is going to be executed and to the technical request that is going to be sent to the rule engine.

- End of a scenario

  The KPI component is notified by the SSP each time that a scenario is executed by the rule engine. To perform its calculation, the KPI component has access to the scenario that is executed and to the execution result (execution response) that is provided by the rule engine. Generally, at this stage, the KPI implementation performs its most useful calculations, based on the result of the scenario execution.

- End of the simulation

  At the end of the simulation, when all scenarios are executed, the SSP requests that the KPI component return a KPI result for the simulation.

KPI result

The KPI result component is a Java class that implements the interface `ilog.rules.dvs.common.output.IlrKPIResult`. It implements two symmetrical methods that are able to encode the KPI result as an array of bytes and to decode it.

Two common KPI result implementations are predefined for your use. The `ilog.rules.dvs.common.output.impl.IlrKPIResultInteger` encapsulates a KPI result that is an integer value. The `ilog.rules.dvs.common.output.impl.IlrKPIResultString` encapsulates a KPI result that is a string value.

To be able to display a KPI result in a simulation report, you must author one more component, `ilog.rules.teamserver.web.components.renderers.IlrScenarioSuiteKPIRenderer`. This component is not necessary if the simulations are run using only the testing and simulation client API.

15.4 Example

This example tests the insurance eligibility project using the Excel scenario suite format from Rule Designer and Decision Center. This example demonstrates the following functions on a use case:

- Create an Excel test suite in Rule Designer
- Run the Excel test suite in Rule Designer and display the execution report
- Repackage the SSP with the insurance eligibility project XOM and redeploy the SSP
- Publish the insurance eligibility project in Decision Center
- Configure Decision Center to use the SSP for running tests and simulations
- Create a test suite in Decision Center, run it, and display the execution report

The prerequisite to this example is to install zRule Execution Server for z/OS and Decision Center, including the testing and simulation modules, in a WebSphere Application Server instance on z/OS.

15.4.1 Creating an Excel test suite in Rule Designer

Complete the following steps to create an Excel test suite in Rule Designer:

1. In Rule Designer, start by opening the **insurance rules** project that was created in Chapter 3, “Getting started with business rules” on page 33, as shown in Figure 15-49.

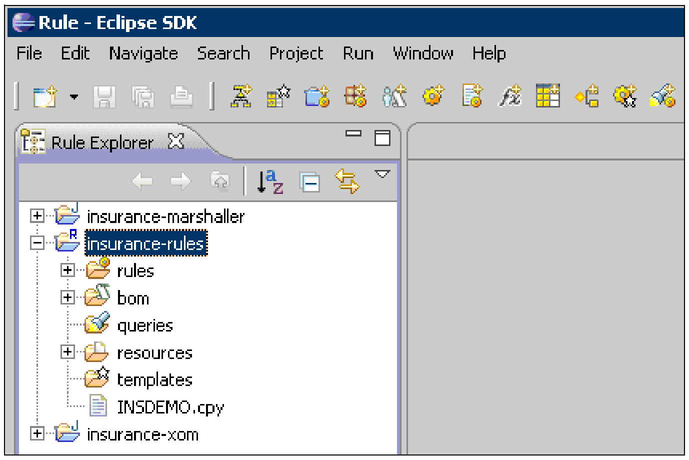


Figure 15-49 The insurance-rules project opened in the Rule Explorer

2. Check that the business object module (BOM) of this project is compatible with the Excel scenario suite format that was used to define a test suite for the project. Right-click the **insurance rules** project and select **Decision Validation Services** → **Check Project**, as shown in Figure 15-50.

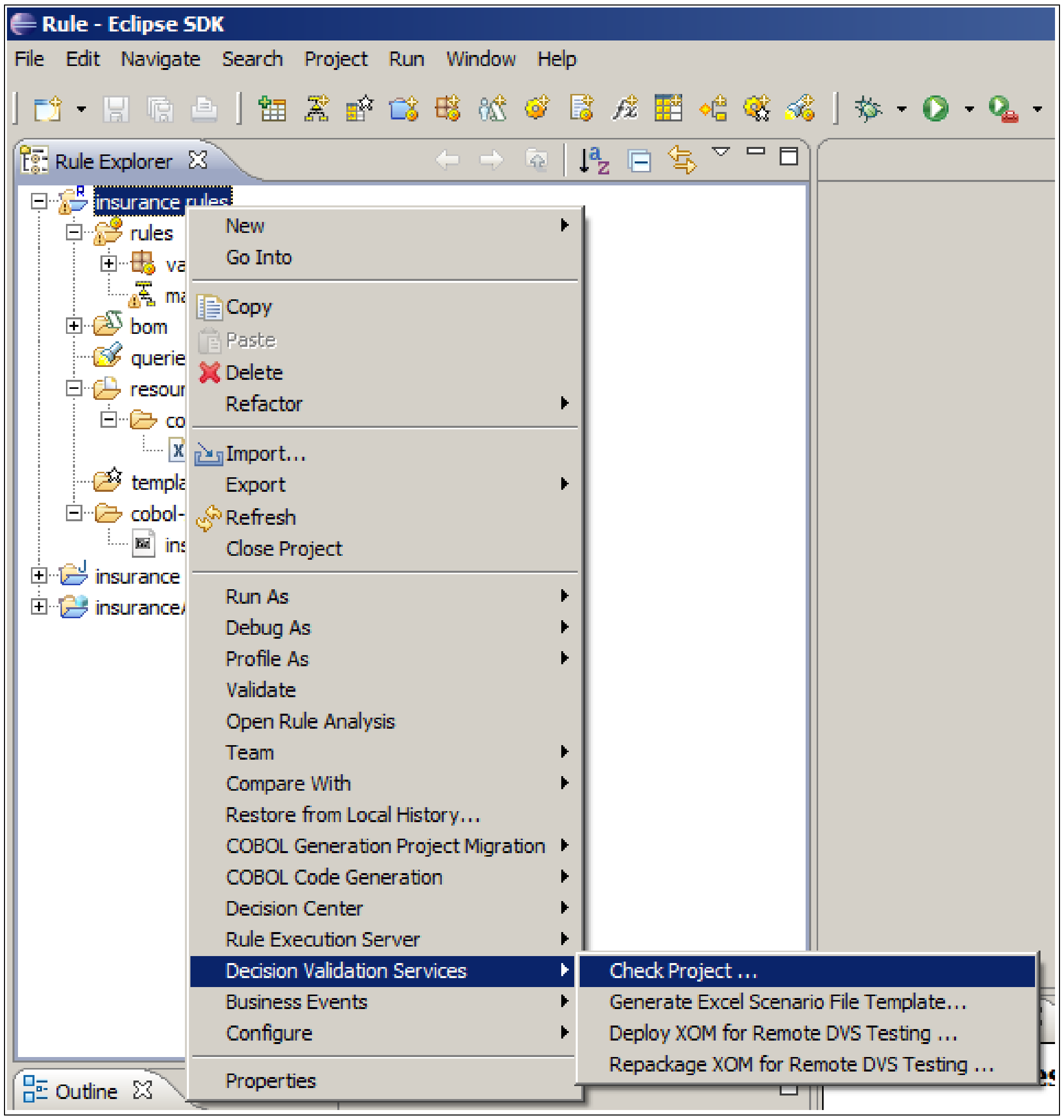
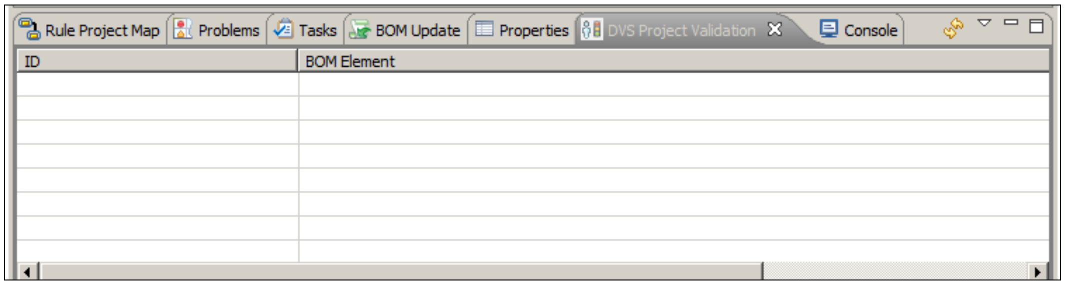


Figure 15-50 Checking the project

3. A new view, DVS Project Validation, opens in Rule Designer. It displays a list of incompatibilities between the project and the Excel scenario suite formats that are provided for testing and simulation, along with the actions to resolve these incompatibilities. For the insurance-rules project, no incompatibilities are displayed in the DVS Project Validation view, as shown in Figure 15-51. Therefore, the project is compatible with all the Excel scenario suite formats.

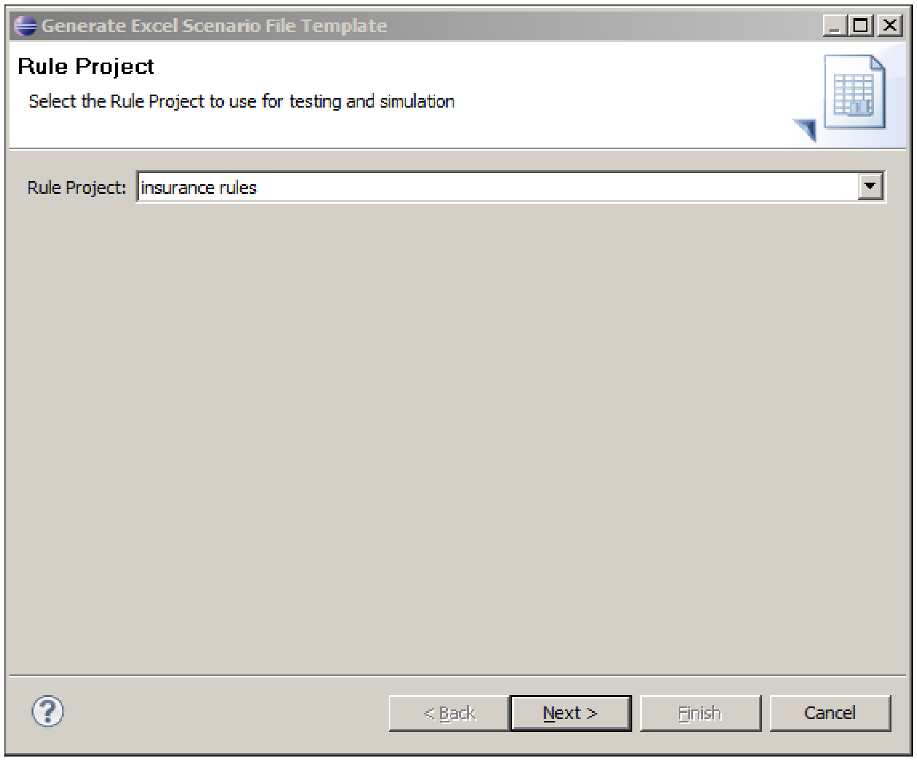


ID	BOM Element

Figure 15-51   DVS Project Validation view that displays no incompatibilities

4. After you are sure that your project is compatible with the Excel scenario suite formats, generate an Excel workbook to define your test suite. To generate the Excel workbook, right-click the **insurance rules** project and select **Decision Validation Services** → **Generate Excel Scenario File Template**. This action opens a new Generate Excel Scenario File Template wizard, which is used to specify the types of tests that you want to define in the Excel scenario suite.

   The first panel of the wizard displays the name of the selected rule project for which the scenario suite is created, as shown in Figure 15-52. Click **Next**.



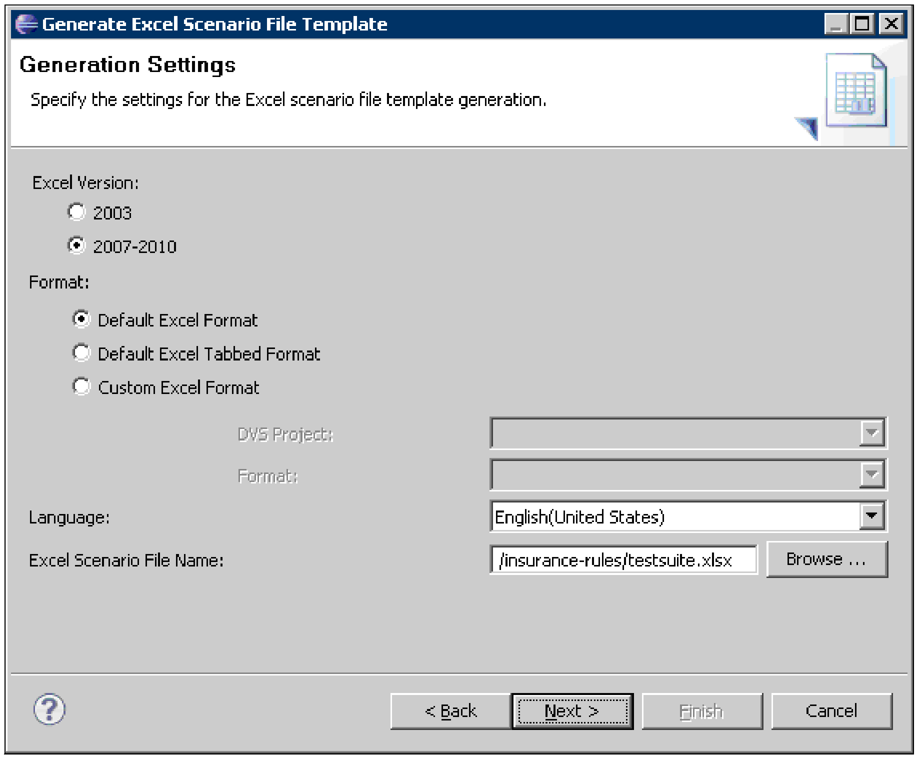
Rule Project

Select the Rule Project to use for testing and simulation

Rule Project: insurance rules

Figure 15-52   The Generate Excel Scenario File Template wizard

5. The Generation Settings panel displays the options for the Excel scenario suite format as shown in Figure 15-53. Select the version of Microsoft Excel to use to edit the test suite. Leave the **Default Excel Format** option selected. Select the language in which to generate the test suite. Click **Next**.


Generate Excel Scenario File Template

Generation Settings

Specify the settings for the Excel scenario file template generation.

Excel Version:
( ) 2003
(●) 2007-2010

Format:
(●) Default Excel Format
( ) Default Excel Tabbed Format
( ) Custom Excel Format

DVS Project:
Format:
Language: English(United States)
Excel Scenario File Name: /insurance-rules/testsuite.xlsx

Figure 15-53 Generation Settings panel

6. In the Expected Results panel, Figure 15-54, define the set of tests that you want to perform on the output of the insurance-rules ruleset.

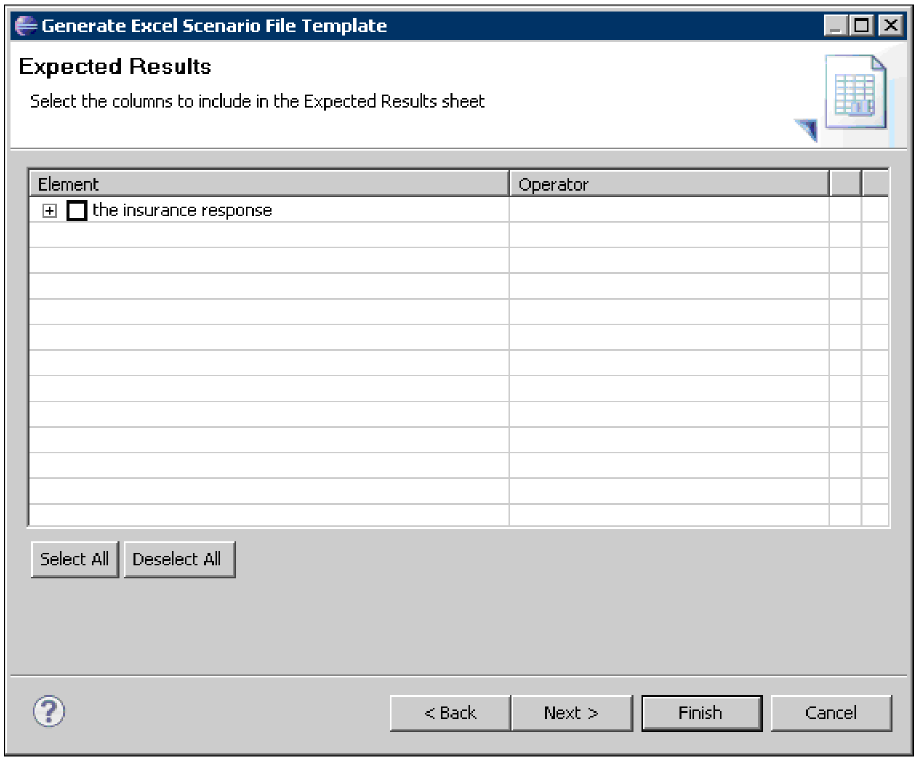
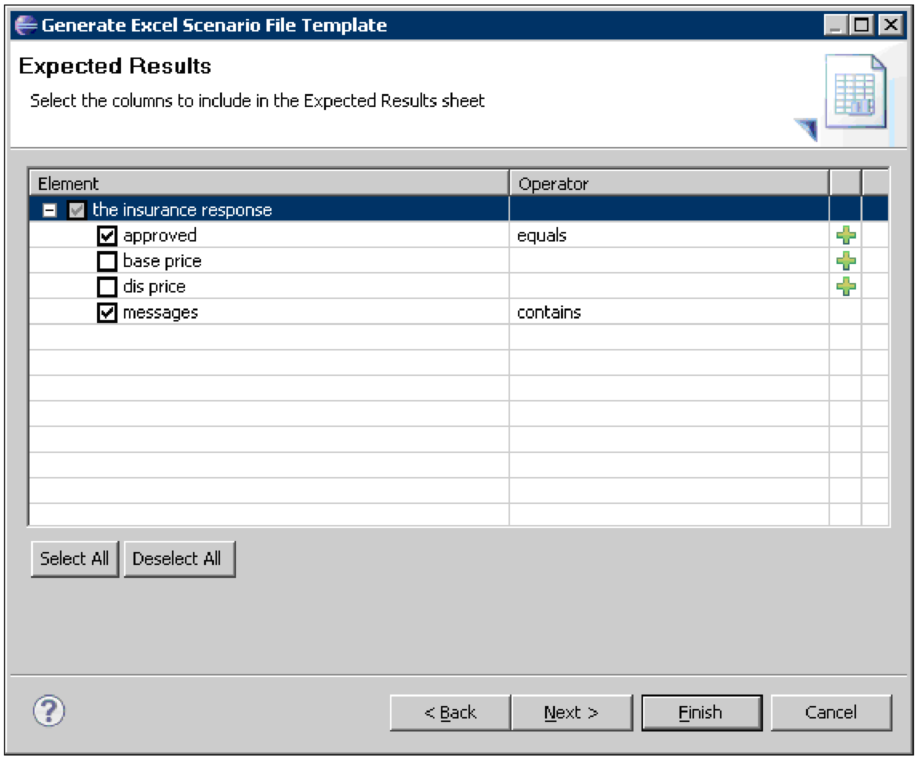


Figure 15-54 Expected results for an Excel test suite

7. For this scenario, select a test on the **approved** attribute of the insurance response, with the operator **equals**, and select a test on the **messages** attribute of the insurance response, with the operator **contains**. You select these tests by checking the corresponding attributes in the panel and then, in the Operator column, by selecting the operator to use, as shown in Figure 15-55. Click **Next**.


Generate Excel Scenario File Template
Expected Results
Select the columns to include in the Expected Results sheet
Select All
Deselect All
< Back
Next >
Finish
Cancel

Element	Operator
☑ the insurance response	
☑ approved	equals
☐ base price	
☐ dis price	
☑ messages	contains

Figure 15-55 Expected Results panel after the selection of tests

8. To add tests on the Expected Execution Details panel, click **Next** (Figure 15-56).

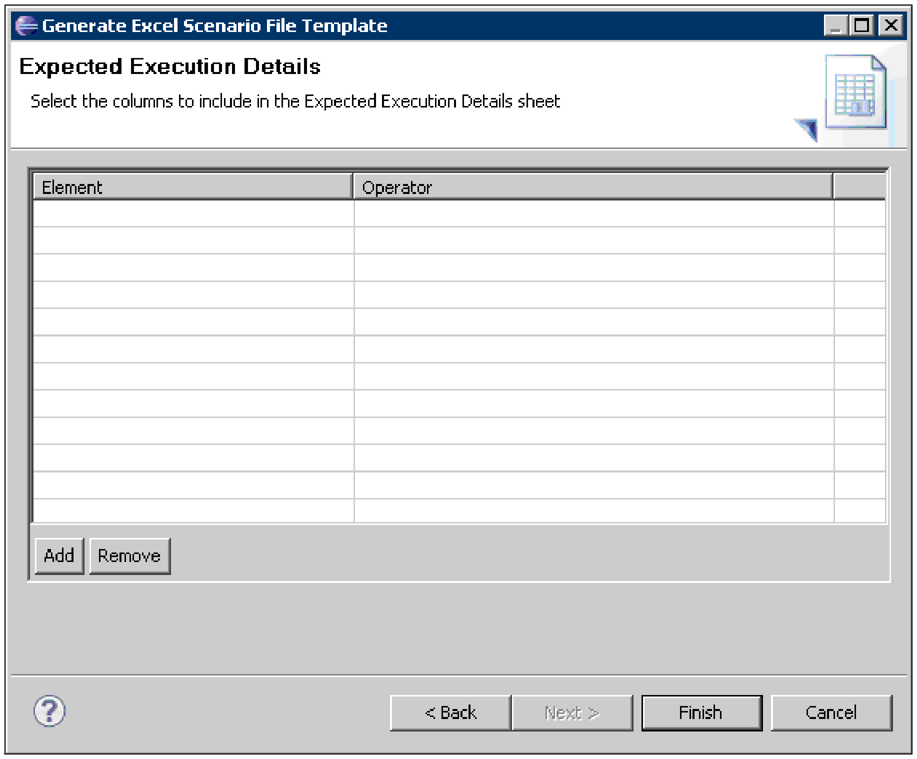
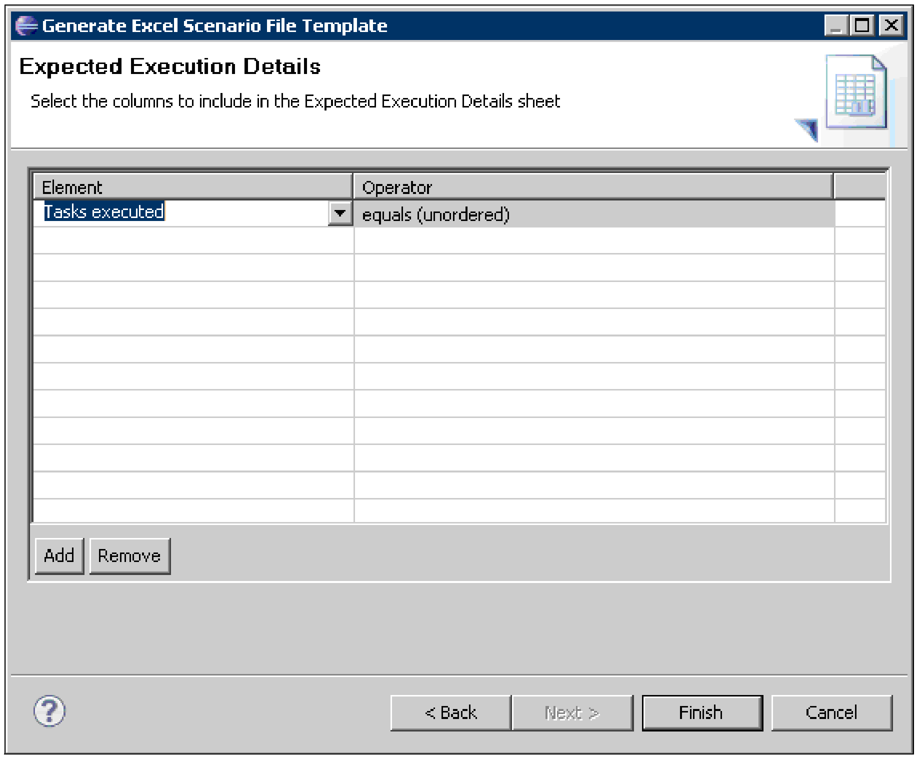

Generate Excel Scenario File Template
Expected Execution Details
Select the columns to include in the Expected Execution Details sheet
Element
Operator
Add
Remove
< Back
Next >
Finish
Cancel

Figure 15-56 Expected execution details

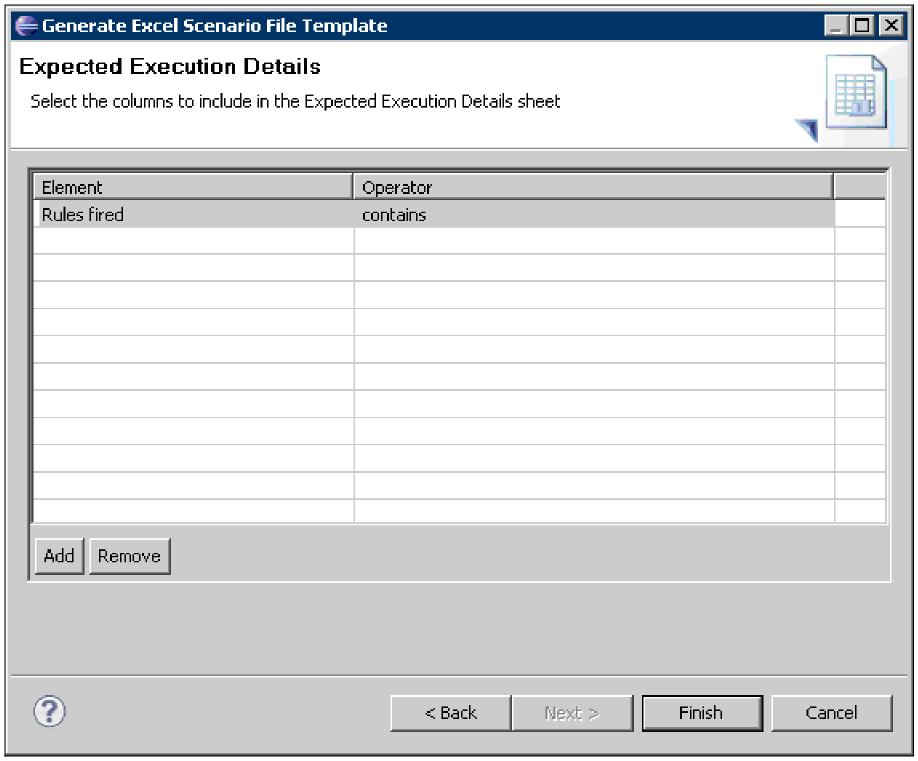
9. To add a test on one execution detail, click **Add**, as shown in Figure 15-57.


Generate Excel Scenario File Template
Expected Execution Details
Select the columns to include in the Expected Execution Details sheet
Add
Remove
< Back
Next >
Finish
Cancel

Element	Operator
Tasks executed	equals (unordered)

Figure 15-57 Adding a test on the execution details

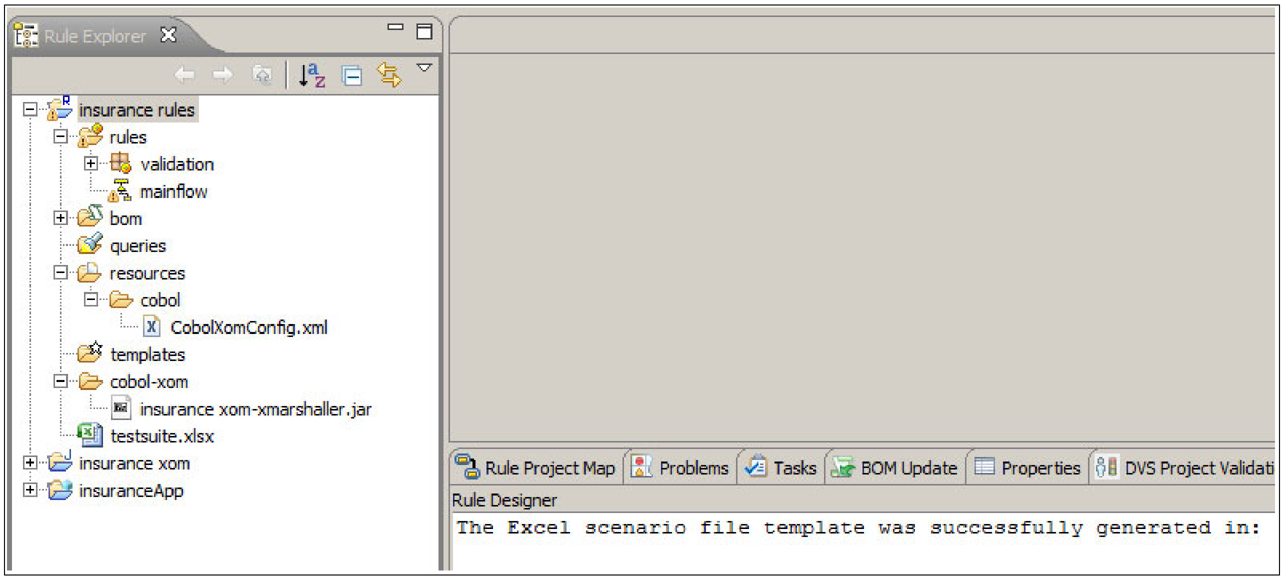
10. This example tests which rules are fired for each scenario. In the Element column, select the **Rules fired** entry. Then, select the **contains** operator in the Operator column (Figure 15-58). Click **Finish**.


Generate Excel Scenario File Template
Expected Execution Details
Select the columns to include in the Expected Execution Details sheet
Add
Remove
< Back
Next >
Finish
Cancel

Element	Operator
Rules fired	contains

Figure 15-58 Selecting a "contains" test that is defined for the list of rules fired

11. The file `testsuite.xls` is generated in the insurance-rules project directory, as confirmed by a message in the console view (Figure 15-59).



```
The Excel scenario file template was successfully generated in:
```

Figure 15-59 The testsuite.xls file generated in the insurance-rules project

12. To open the file for editing in Microsoft Excel, right-click **testsuite.xls** in the Rule Explorer and select **Open With** → **System Editor** (Figure 15-60).

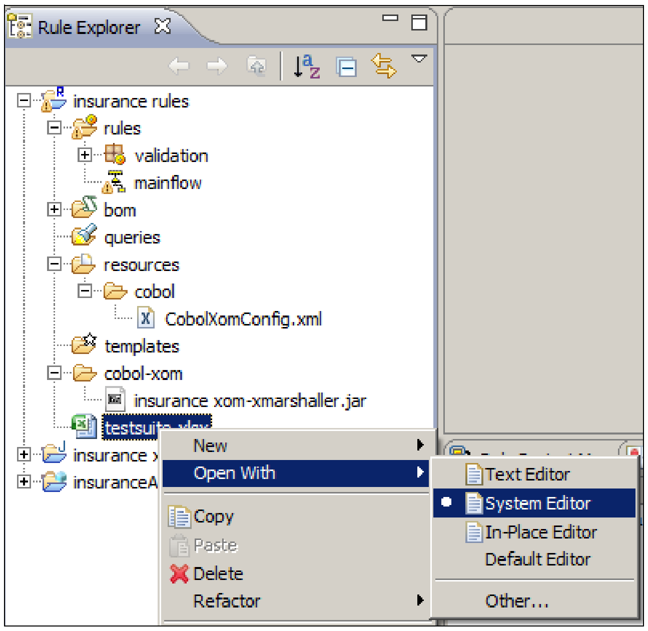


Figure 15-60 Selecting the Open With System Editor menu option

13. In Microsoft Excel, you can see that the generated workbook contains four worksheets, as shown by the tabs in Figure 15-61:

   – Scenarios worksheet to enter data for each of your test scenarios

   – Expected Results worksheet to optionally enter tests on the output parameters for each scenario

   – Expected Execution Details worksheet to optionally enter tests on the execution details for each scenario

   – HELP worksheet, which provides an embedded help page to help you define your test suite in the workbook

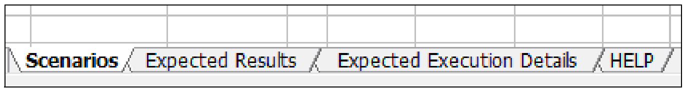
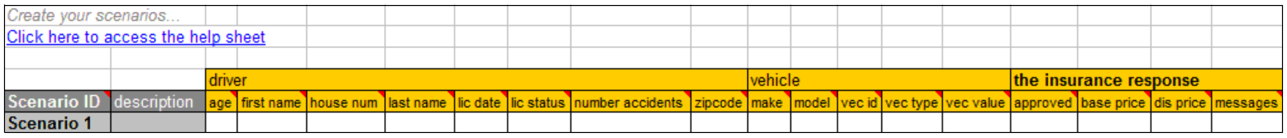


Figure 15-61 Worksheets in an Excel test suite

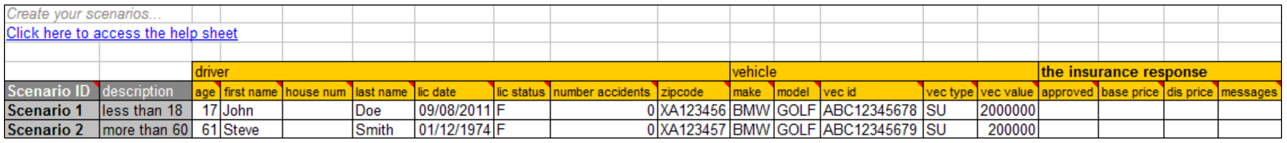
14. Start by using the Scenarios worksheet to create the first scenario for the insurance-rules ruleset. The Scenario worksheet displays a table in which each row represents a test scenario. For each row, the first column of the table is to define the name of the scenario and the second column is to define an optional description for the scenario. The other columns are to define values for the input parameter attributes of the scenario. By default, the worksheet displays an empty row to be filled with the values for the first scenario (Figure 15-62).



Create your scenarios...																		
Click here to access the help sheet																		
		driver								vehicle					**the insurance response**			
Scenario ID	description	age	first name	house num	last name	lic date	lic status	number accidents	zipcode	make	model	vec id	vec type	vec value	approved	base price	dis price	messages
Scenario 1																		

Figure 15-62 Initial content of the Scenarios worksheet

15. Enter the values in the Scenario worksheet, as shown in Figure 15-63, to define two similar scenarios that differ by the age of the driver. Both scenarios (Scenario 1 and Scenario 2) are supposed to trigger the execution of the rule validation.MaxiMinimumAge.



Create your scenarios...

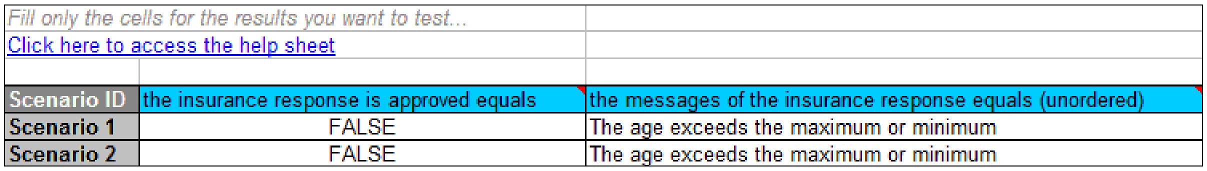
Click here to access the help sheet

		driver								vehicle					the insurance response			
Scenario ID	description	age	first name	house num	last name	lic date	lic status	number accidents	zipcode	make	model	vec id	vec type	vec value	approved	base price	dis price	messages
Scenario 1	less than 18	17	John		Doe	09/08/2011	F	0	XA123456	BMW	GOLF	ABC12345678	SU	2000000				
Scenario 2	more than 60	61	Steve		Smith	01/12/1974	F	0	XA123457	BMW	GOLF	ABC12345679	SU	200000				

Figure 15-63 Two test scenarios for the insurance-rules ruleset

16. The next step is to specify the tests on the output parameter for each scenario. Display the Expected Results worksheet by clicking the **Expected Results** worksheet tab at the bottom of the Excel workbook. Enter the rows that are shown in Figure 15-64 to define the expectations for Scenario 1 and Scenario 2:

 – For Scenario 1, type `FALSE` and `The age exceeds the maximum or minimum`
 – For Scenario 2, type `FALSE` and `The age exceeds the maximum or minimum`



Fill only the cells for the results you want to test...

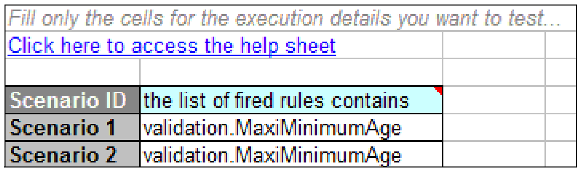
Click here to access the help sheet

Scenario ID	the insurance response is approved equals	the messages of the insurance response equals (unordered)
Scenario 1	FALSE	The age exceeds the maximum or minimum
Scenario 2	FALSE	The age exceeds the maximum or minimum

Figure 15-64 Testing the output parameter of the insurance-rules ruleset

17. Specify the tests on the execution details. Display the Expected Execution Details worksheet by clicking the **Expected Execution Details** worksheet tab at the bottom of the Excel workbook. Enter the rows that are shown in Figure 15-65:

 – For Scenario 1, type `validation.MaxiMinimumAge`
 – For Scenario 2, type `validation.MaxiMinimumAge`



Fill only the cells for the execution details you want to test...

Click here to access the help sheet

Scenario ID	the list of fired rules contains
Scenario 1	validation.MaxiMinimumAge
Scenario 2	validation.MaxiMinimumAge

Figure 15-65 Testing the list of rules fired for the insurance-rules ruleset

18. Save the content of the Excel workbook and proceed to 15.4.2, “Running the Excel test suite in Rule Designer and then displaying the execution report” on page 285, which is running the test suite in Rule Designer and displaying the execution report.

15.4.2 Running the Excel test suite in Rule Designer and then displaying the execution report

Complete the following steps to run the Excel test suite in Rule Designer and display the execution report:

1. To run an Excel test suite in Rule Designer, you must create a run configuration, which is also called a *launch configuration*. To create a run configuration, click **Run** → **Run Configurations**, as shown in Figure 15-66.

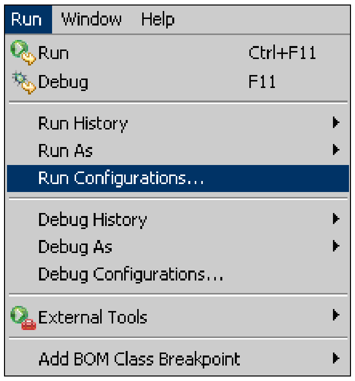
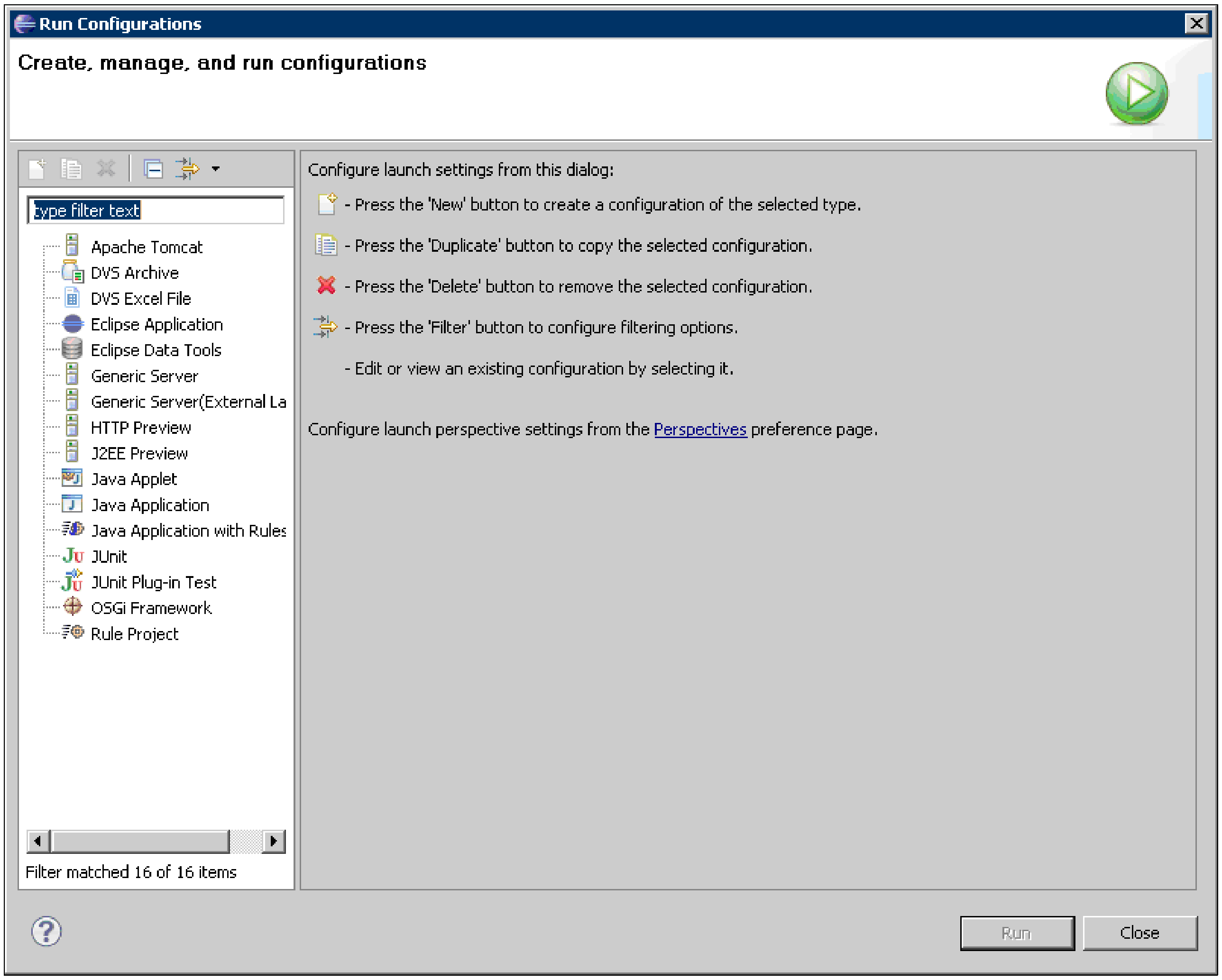


Figure 15-66 Run Configurations menu

2. The Run Configurations editor opens, as shown in Figure 15-67.



Filter matched 16 of 16 items

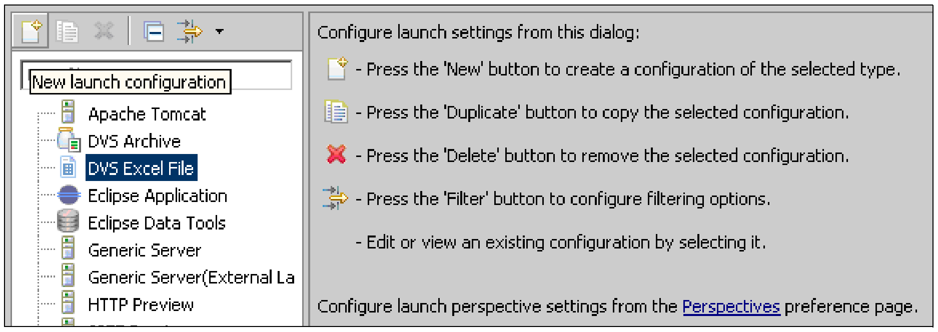
Configure launch settings from this dialog:

- Press the 'New' button to create a configuration of the selected type.
- Press the 'Duplicate' button to copy the selected configuration.
- Press the 'Delete' button to remove the selected configuration.
- Press the 'Filter' button to configure filtering options.
- Edit or view an existing configuration by selecting it.

Configure launch perspective settings from the Perspectives preference page.

Figure 15-67 Run Configurations editor

3. To create a run configuration for the Excel test suite in this editor, select the **DVS Excel File** entry in the left section of the editor. Then, click the **New Launch Configuration** icon (the leftmost icon in the editor toolbar), as shown in Figure 15-68.



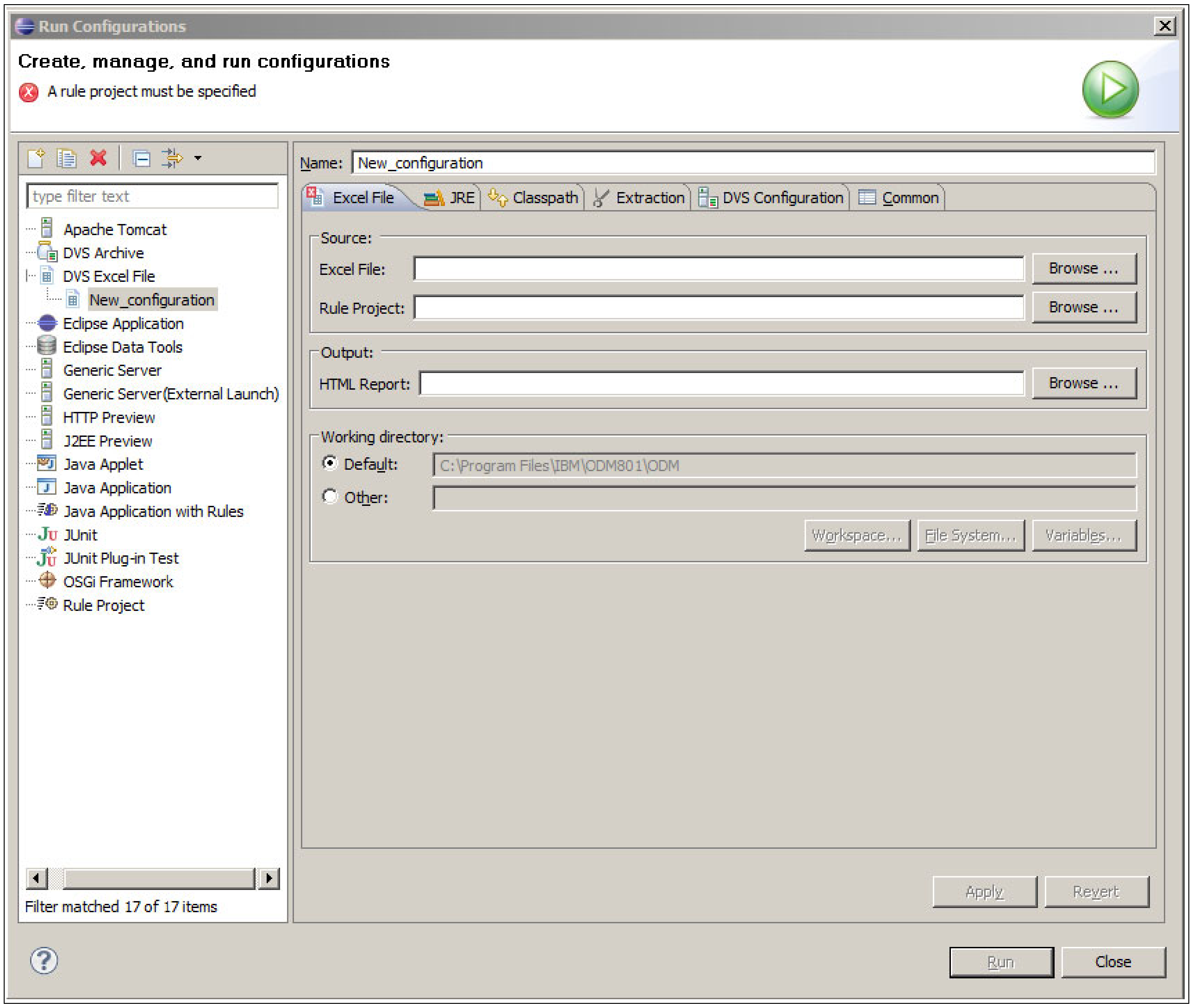
Configure launch settings from this dialog:

- Press the 'New' button to create a configuration of the selected type.
- Press the 'Duplicate' button to copy the selected configuration.
- Press the 'Delete' button to remove the selected configuration.
- Press the 'Filter' button to configure filtering options.
- Edit or view an existing configuration by selecting it.

Configure launch perspective settings from the Perspectives preference page.

Figure 15-68 Selecting the DVS Excel File type

4. The right side of the editor is updated to display a dedicated editor for DVS Excel File run configurations (Figure 15-69).


Run Configurations

Create, manage, and run configurations

A rule project must be specified

type filter text

- Apache Tomcat
- DVS Archive
- DVS Excel File
  - New_configuration
- Eclipse Application
- Eclipse Data Tools
- Generic Server
- Generic Server(External Launch)
- HTTP Preview
- J2EE Preview
- Java Applet
- Java Application
- Java Application with Rules
- JUnit
- JUnit Plug-in Test
- OSGi Framework
- Rule Project

Filter matched 17 of 17 items

Name: New_configuration

Excel File | JRE | Classpath | Extraction | DVS Configuration | Common

Source:

Excel File:

Rule Project:

Output:

HTML Report:

Working directory:

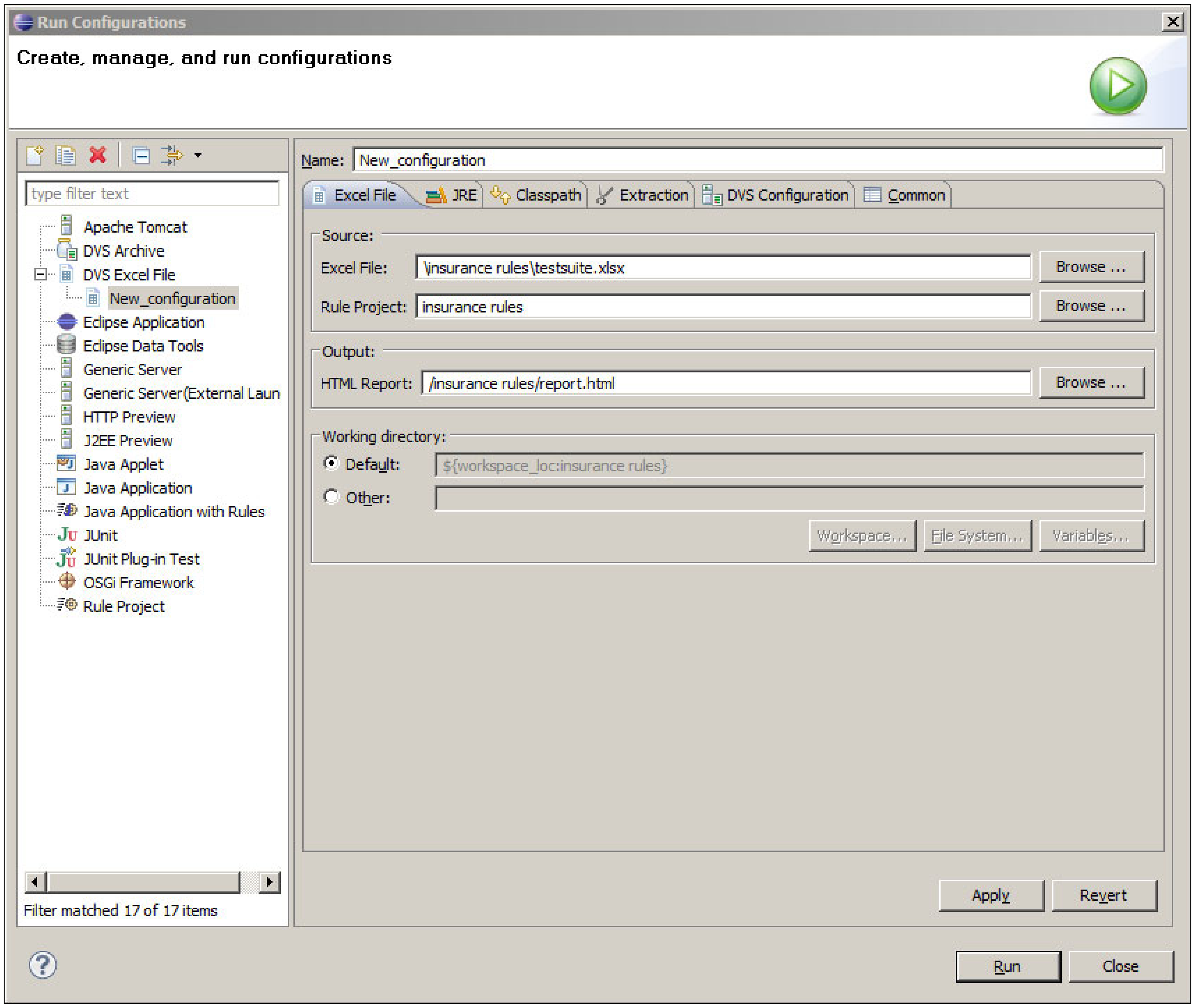
Default: C:\Program Files\IBM\ODM801\ODM

Other:

Figure 15-69 DVS Excel File Run Configurations editor

5. Update the name of this new run configuration from New_configuration to `Run test suite against insurance rules`. Click **Browse** to select the following information for this run configuration (Figure 15-70):
   - For Excel File, select **\insurance-rules\testsuite.xls**
   - For Rule Project, select **insurance-rules**
   - For HTML Report, select **\insurance-rules\report.html**

   Click **Apply** to save the run configuration. Then, click **Run** to execute the test suite. The execution starts.


Run Configurations

Create, manage, and run configurations

type filter text

- Apache Tomcat
- DVS Archive
- DVS Excel File
  - New_configuration
- Eclipse Application
- Eclipse Data Tools
- Generic Server
- Generic Server(External Laun
- HTTP Preview
- J2EE Preview
- Java Applet
- Java Application
- Java Application with Rules
- JUnit
- JUnit Plug-in Test
- OSGi Framework
- Rule Project

Filter matched 17 of 17 items

Name: New_configuration

Excel File | JRE | Classpath | Extraction | DVS Configuration | Common

Source:

Excel File: \insurance rules\testsuite.xlsx

Rule Project: insurance rules

Output:

HTML Report: /insurance rules/report.html

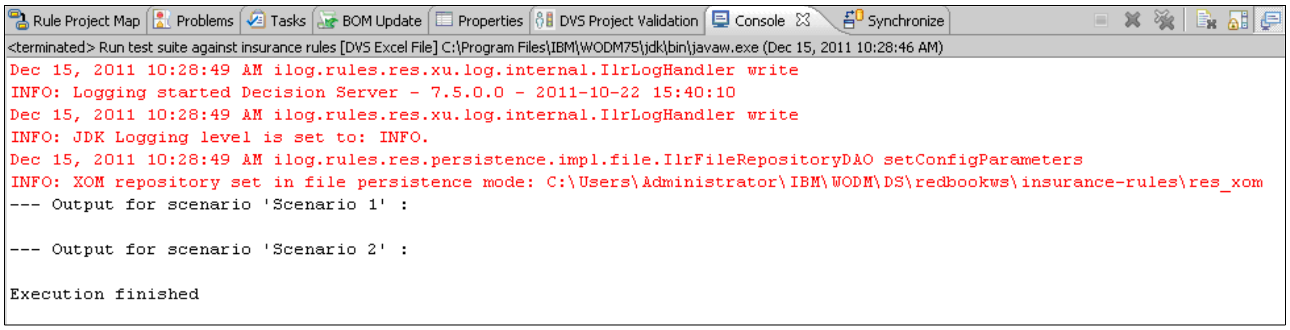
Working directory:

[✓ Default:] ${workspace_loc:insurance rules}

Other:

Figure 15-70 The run configuration for the insurance-rules Excel test suite

6. The content of the Console view is updated in real time with the execution logs for the test suite (Figure 15-71).



```
<terminated> Run test suite against insurance rules [DVS Excel File] C:\Program Files\IBM\WODM75\jdk\bin\javaw.exe (Dec 15, 2011 10:28:46 AM)
Dec 15, 2011 10:28:49 AM ilog.rules.res.xu.log.internal.IlrLogHandler write
INFO: Logging started Decision Server - 7.5.0.0 - 2011-10-22 15:40:10
Dec 15, 2011 10:28:49 AM ilog.rules.res.xu.log.internal.IlrLogHandler write
INFO: JDK Logging level is set to: INFO.
Dec 15, 2011 10:28:49 AM ilog.rules.res.persistence.impl.file.IlrFileRepositoryDAO setConfigParameters
INFO: XOM repository set in file persistence mode: C:\Users\Administrator\IBM\WODM\DS\redbookws\insurance-rules\res_xom
--- Output for scenario 'Scenario 1' :

--- Output for scenario 'Scenario 2' :

Execution finished
```

Figure 15-71 Test suite execution logs in the Console view of Rule Designer

7. When the message “`Execution finished`” is displayed in the console view log (Figure 15-71), the HTML execution report is generated. Display the HTML execution report to see the result of the test suite execution. To see the report in the Rule Explorer, first refresh the content of the insurance-rules project so that Rule Designer checks for new files in the project directory. To refresh the content, right-click the **insurance-rules** project in the Rule Explorer and select **Refresh** (Figure 15-72).

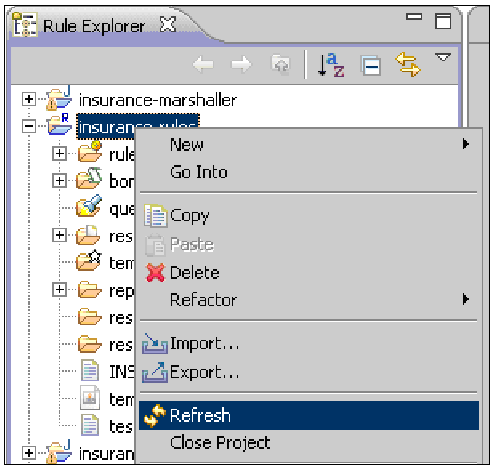


Figure 15-72 Refresh menu option for the insurance-rules project

8. After the content of the rule project is refreshed, the report file `report.html`, which was generated after the execution of the test suite, is displayed in the Rule Explorer. Right-click **report.html** and select **Open With** → **Web Browser** to display the report in Rule Designer (Figure 15-73).

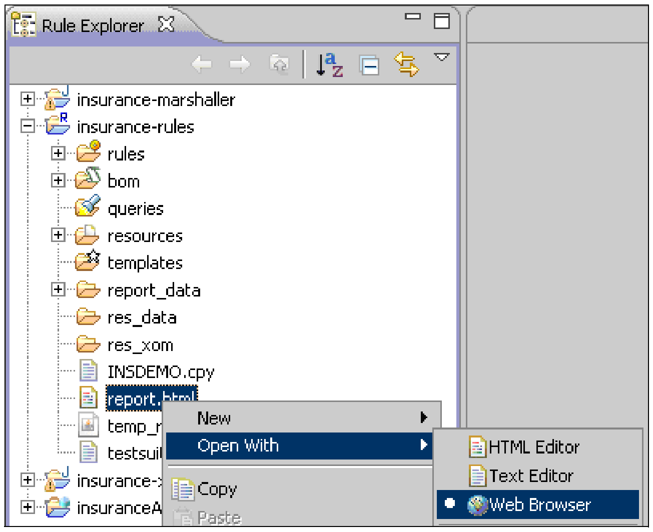
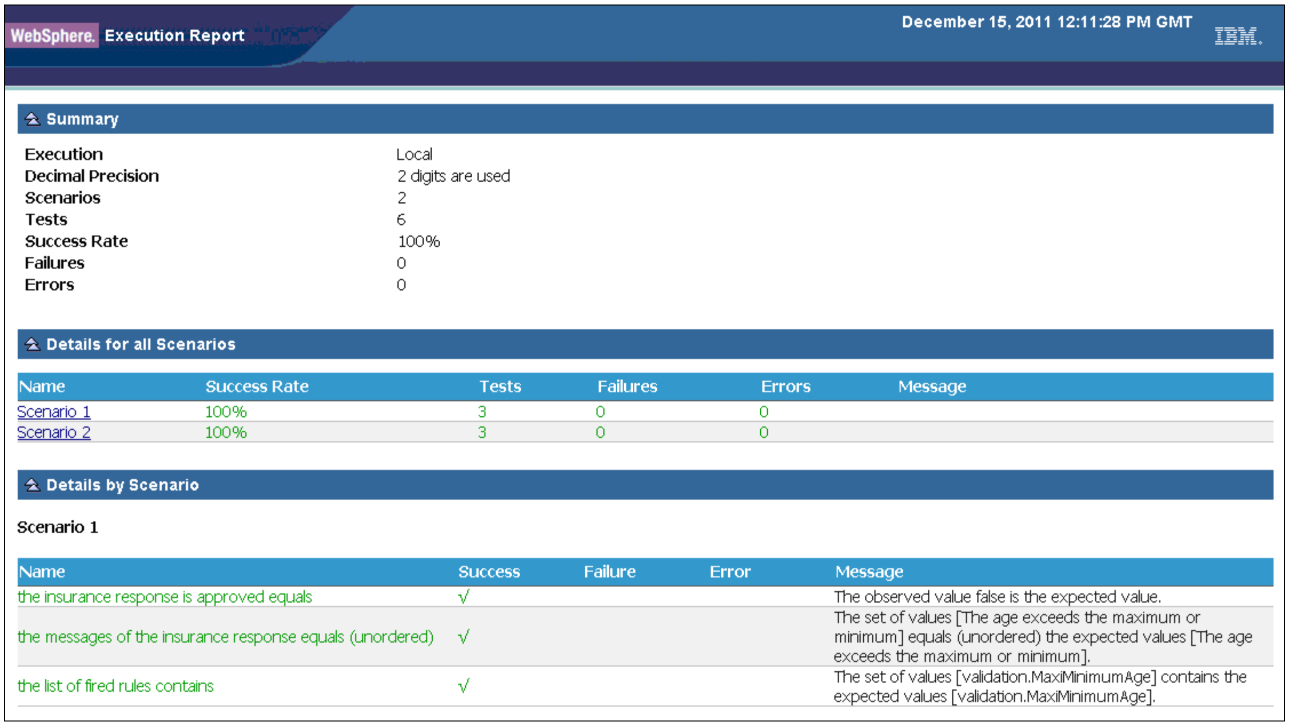


Figure 15-73 Opening a test report with the Rule Designer web browser

9. The Execution Report opens in the internal web browser of Rule Designer. You see that the test suite executed successfully and that the test success rates are 100%, as shown in Figure 15-74.



WebSphere. Execution Report

December 15, 2011 12:11:28 PM GMT

IBM.

Summary

Execution	Local
Decimal Precision	2 digits are used
Scenarios	2
Tests	6
Success Rate	100%
Failures	0
Errors	0

Details for all Scenarios

Name	Success Rate	Tests	Failures	Errors	Message
Scenario 1	100%	3	0	0	
Scenario 2	100%	3	0	0	

Details by Scenario

Scenario 1

Name	Success	Failure	Error	Message
the insurance response is approved equals	√			The observed value false is the expected value.
the messages of the insurance response equals (unordered)	√			The set of values [The age exceeds the maximum or minimum] equals (unordered) the expected values [The age exceeds the maximum or minimum].
the list of fired rules contains	√			The set of values [validation.MaxiMinimumAge] contains the expected values [validation.MaxiMinimumAge].

Figure 15-74 Rule Designer Execution Report for the insurance-rules test suite

15.4.3 Repackaging the SSP

Complete the following steps to repackage the SSP with the insurance eligibility project XOM and redeploy the SSP:

1. To execute the same test suite from Decision Center against the SSP, first repackage the XOM of the insurance eligibility project into the SSP and redeploy it. Create a new type of project, which is a DVS project that defines a customization for testing and simulation in the SSP and Decision Center. To create this project, right-click in the Rule Explorer white space and select **New** → **Other** (Figure 15-75).

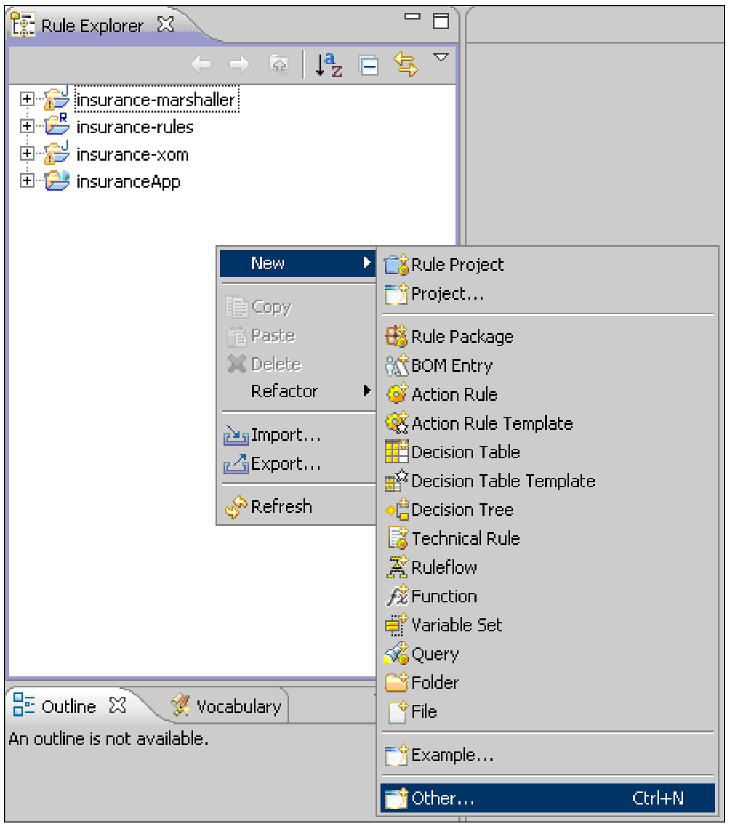


Figure 15-75 Creating a new DVS project

2. This action displays the New wizard panel that prompts you to select a wizard. Select the wizard type **Rule Designer** → **Decision Validation Services** → **DVS Project** (Figure 15-76). Click **Next**.

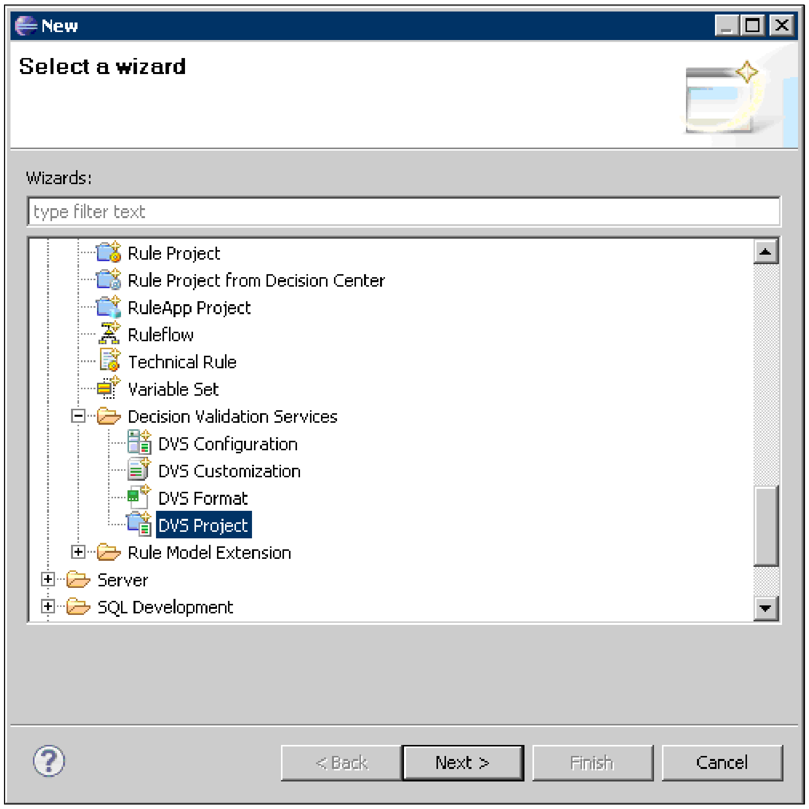
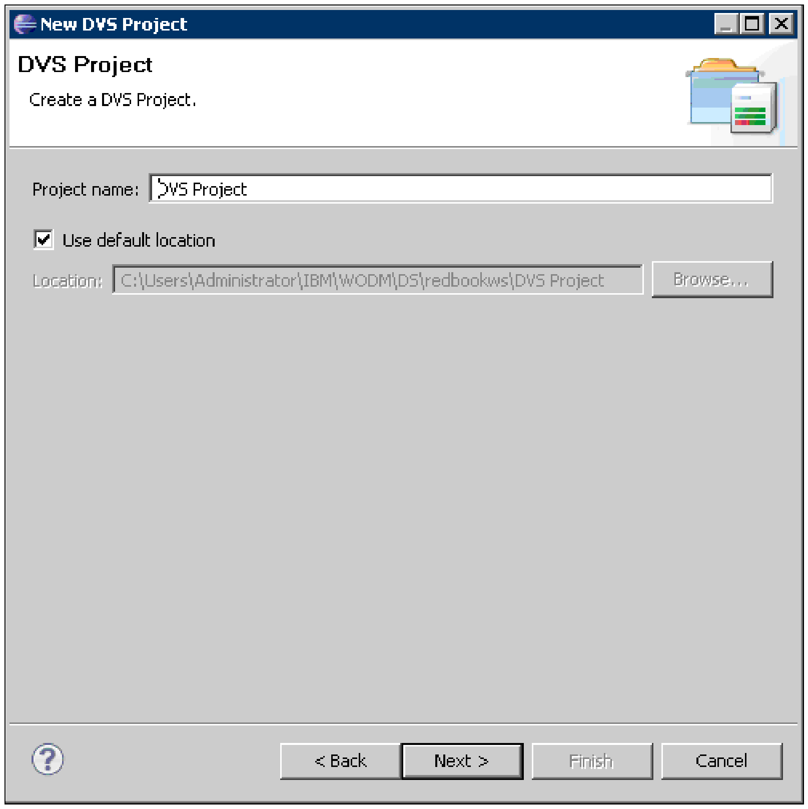


Figure 15-76 Selecting the new DVS Project wizard

3. This DVS Project wizard prompts you to specify a project name and location for the new project (Figure 15-77). Keep the default options and click **Next**.


New DVS Project

DVS Project

Create a DVS Project.

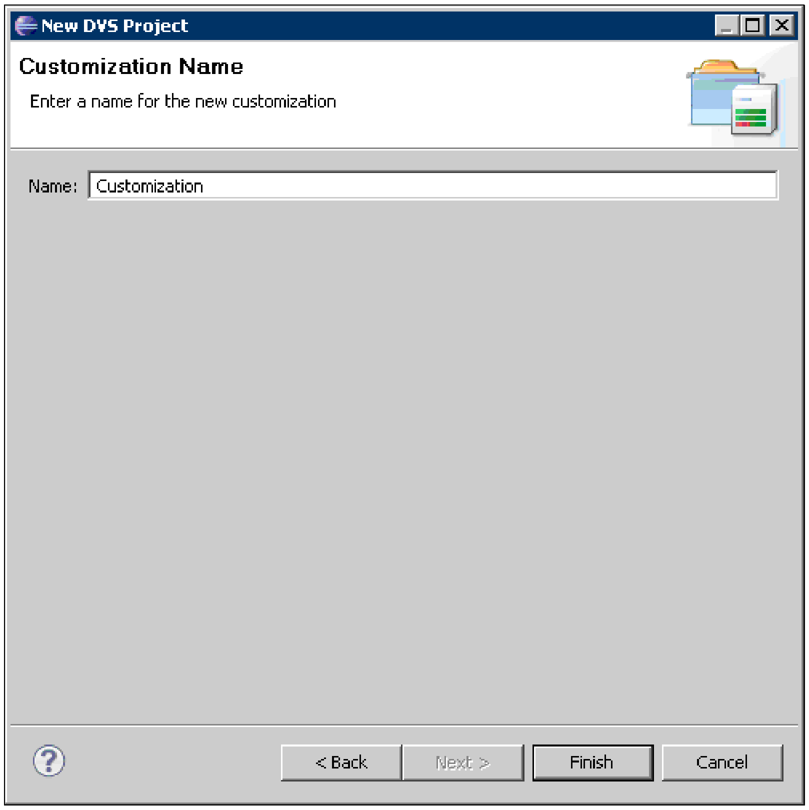
Project name: DVS Project

[✓] Use default location

Location: C:\Users\Administrator\IBM\WODM\DS\redbookws\DVS Project

Figure 15-77 Specifying a project name for the new DVS project

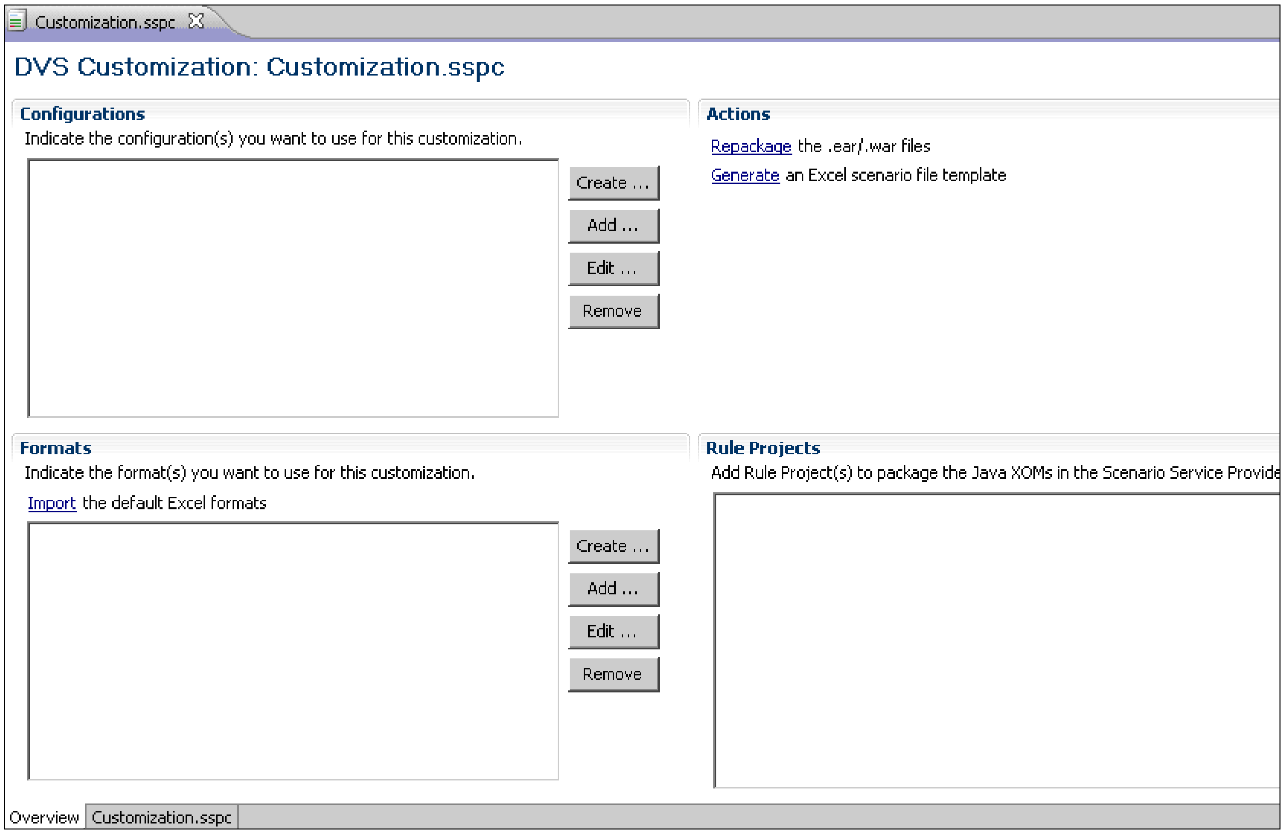
4. On the Customization Name panel, specify the name of the new customization to create in the project (Figure 15-78). Keep the default name for the customization and click **Finish**.



Name: Customization

Figure 15-78 Specifying the name of the new customization

5. The DVS Customization editor is then displayed in Rule Designer, allowing you to edit the new customization (Figure 15-79). First, add a server configuration to the customization so that the repackaging mechanism knows to use WebSphere Application Server to host the SSP and Decision Center. On the DVS Customization page that is shown in Figure 15-79, in the Configurations section, click **Create**.


Customization.sspc

DVS Customization: Customization.sspc

Configurations
Indicate the configuration(s) you want to use for this customization.

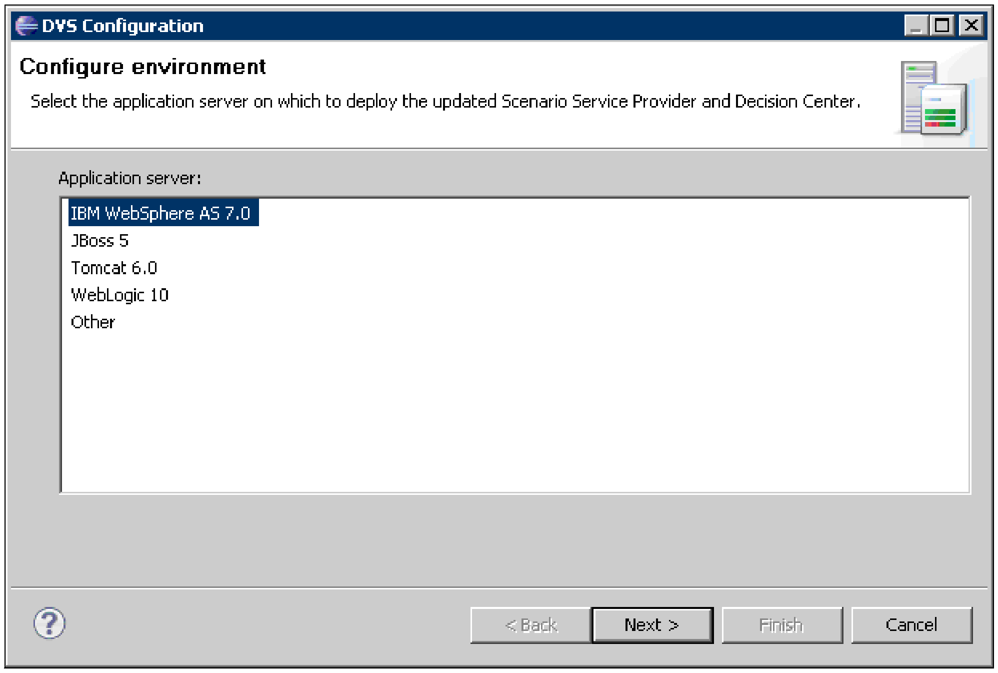
Actions
Repackage the .ear/.war files
Generate an Excel scenario file template

Formats
Indicate the format(s) you want to use for this customization.
Import the default Excel formats

Rule Projects
Add Rule Project(s) to package the Java XOMs in the Scenario Service Provide

Figure 15-79 DVS Customization editor

6. On the Configure environment panel of the DVS Configuration wizard (Figure 15-80), select **IBM WebSphere AS 7.0** as the application server on which to deploy the SSP and Decision Center. Click **Next**.


DVS Configuration

Configure environment

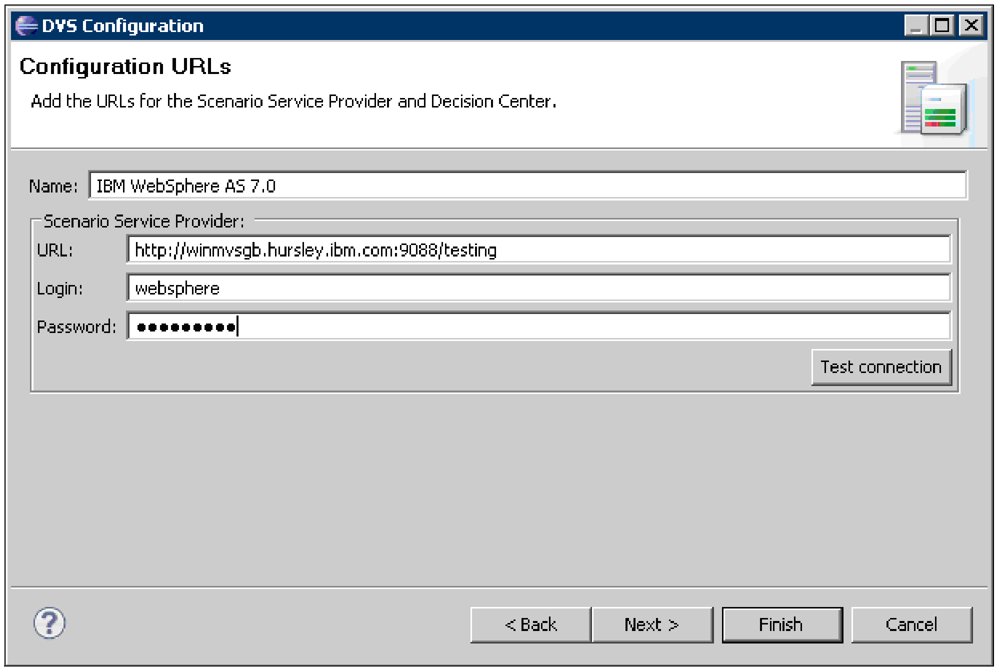
Select the application server on which to deploy the updated Scenario Service Provider and Decision Center.

Application server:

- IBM WebSphere AS 7.0
- JBoss 5
- Tomcat 6.0
- WebLogic 10
- Other

Figure 15-80 DVS Configuration wizard

7. On the Configuration URLs panel, enter the URL to your Scenario Service Provider instance, which is the URL to your WebSphere Application Server 7 server, ending with `/testing`. Enter the login and password credentials to connect to the SSP (Figure 15-81). Click **Test connection** to check that the SSP can be contacted with the provided URL and credentials. Click **Finish**.


DVS Configuration

Configuration URLs

Add the URLs for the Scenario Service Provider and Decision Center.

Name: IBM WebSphere AS 7.0

Scenario Service Provider:

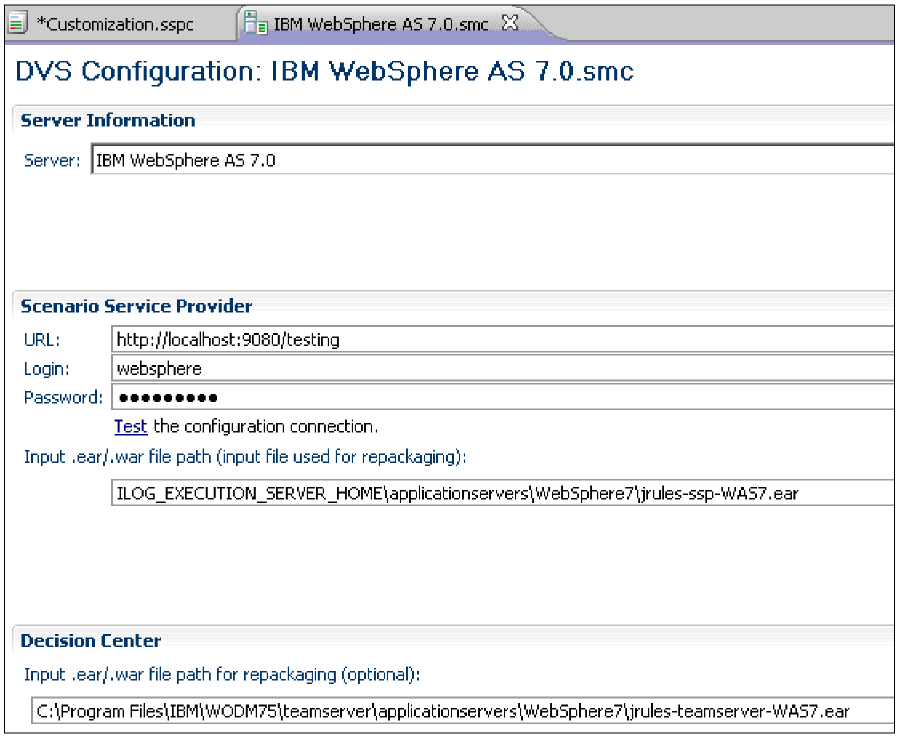
URL: http://winmvsgb.hursley.ibm.com:9088/testing

Login: websphere

Password: ●●●●●●●●●

Figure 15-81 Configuration URL for a DVS configuration

8. The DVS Configuration editor opens for the WebSphere Application Server instance that was specified (Figure 15-82). On the DVS Configuration page that is shown in Figure 15-82, click the **Customization.sspc** tab to display the DVS Customization editor, and press Ctrl+S to save its content.



DVS Configuration: IBM WebSphere AS 7.0.smc

Server Information

Server: IBM WebSphere AS 7.0

Scenario Service Provider

URL: http://localhost:9080/testing

Login: websphere

Password: ●●●●●●●●●

Test the configuration connection.

Input .ear/.war file path (input file used for repackaging):

ILOG_EXECUTION_SERVER_HOME\applicationservers\WebSphere7\jrules-ssp-WAS7.ear

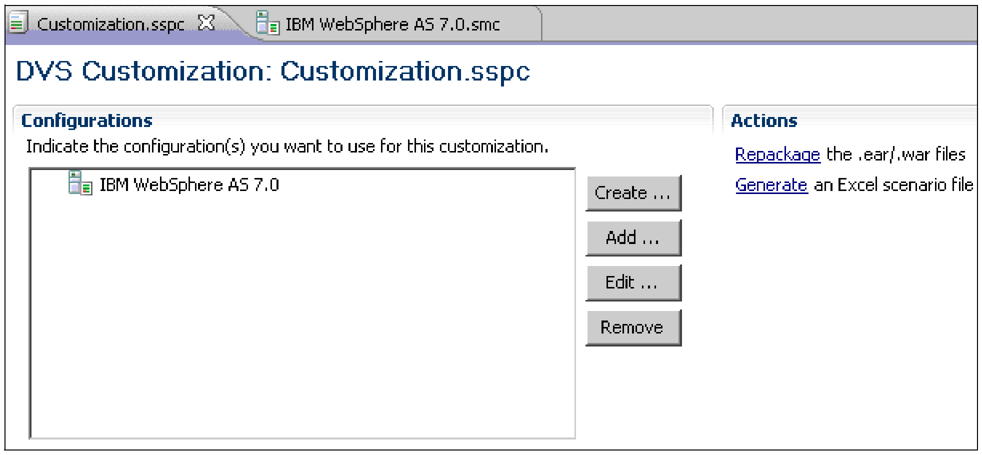
Decision Center

Input .ear/.war file path for repackaging (optional):

C:\Program Files\IBM\WODM75\teamserver\applicationservers\WebSphere7\jrules-teamserver-WAS7.ear

Figure 15-82 DVS Configuration editor for a WebSphere Application Server 7.0 instance

9. The Configurations section is updated with the new server configuration that was defined (Figure 15-83). Add the insurance-rules rule project to the customization so that Rule Designer is aware of the XOM to deploy to the SSP when repackaging it. Click **Add** in Configurations section of the editor.



DVS Customization: Customization.sspc

Configurations

Indicate the configuration(s) you want to use for this customization.

IBM WebSphere AS 7.0

Actions

Repackage the .ear/.war files

Generate an Excel scenario file

Figure 15-83 DVS customization updated with a new server configuration

10. A pop-up window opens to select the rule project to add to the customization (Figure 15-84). Select the **insurance-rules** project and click **OK** to add insurance-rules to the Rule Projects section of the Customization editor.

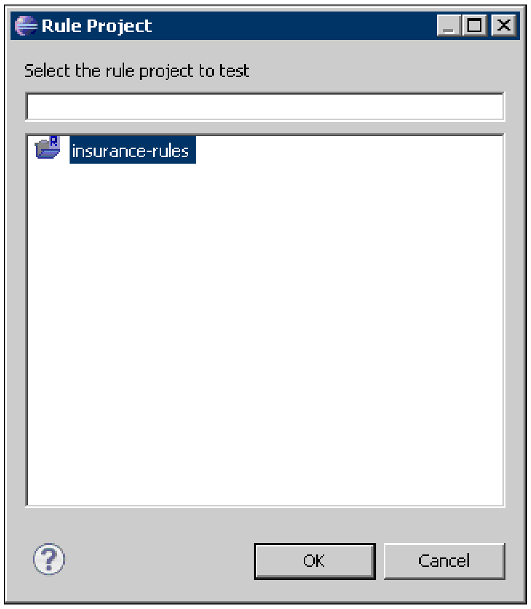
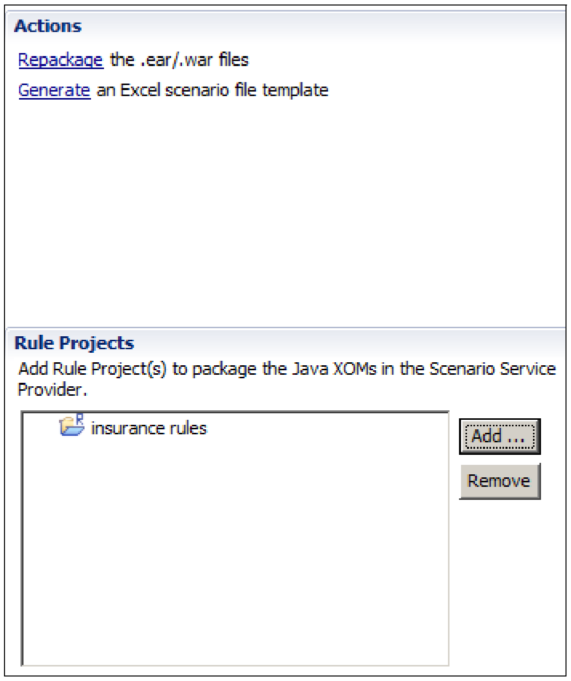


Figure 15-84 Selecting the rule project for the DVS customization

11. Figure 15-85 shows the insurance-rules project added to the Rule Projects section of the DVS Customization editor. Save the content of the editor by pressing Ctrl+S. On the page that is shown in Figure 15-85, in the Actions section of the editor, click the **Repackage** the .ear/.war files link.


Actions

Repackage the .ear/.war files

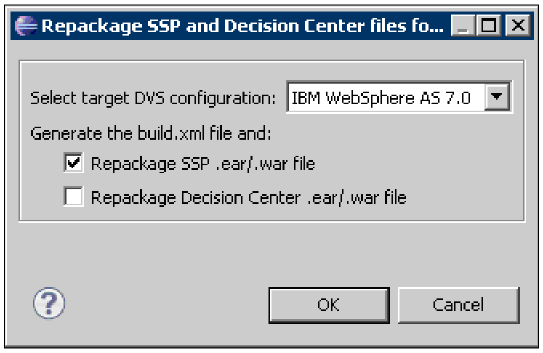
Generate an Excel scenario file template

Rule Projects

Add Rule Project(s) to package the Java XOMs in the Scenario Service Provider.

Figure 15-85 Adding the insurance rules project

12. A pop-up window opens so that you can specify the file to repackage (Figure 15-86). Because you repackaged the SSP with the XOM of the insurance-rules project only (this option was already selected), clear the **Repackage Decision Center .ear/.war file** option and click **OK**.



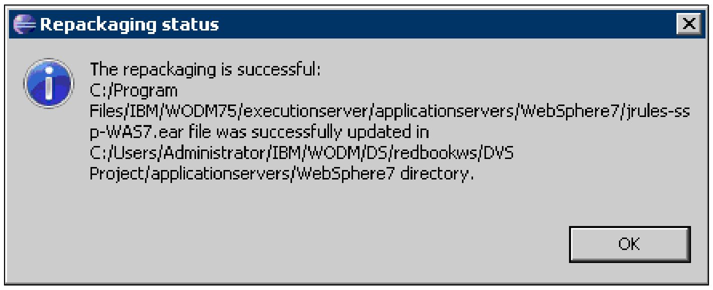
Select target DVS configuration: IBM WebSphere AS 7.0

Generate the build.xml file and:

- [x] Repackage SSP .ear/.war file
- [ ] Repackage Decision Center .ear/.war file

Figure 15-86 Option to repackage only the SSP

13. When the repackaging is complete, a message window opens to confirm the status of the repackaging operation. The last file path that is displayed in this message window informs you of the location of the repackaged SSP, which is redeployed in the WebSphere Application Server instance (Figure 15-87).



The repackaging is successful:
C:/Program Files/IBM/WODM75/executionserver/applicationservers/WebSphere7/jrules-ssp-WAS7.ear file was successfully updated in C:/Users/Administrator/IBM/WODM/DS/redbookws/DVS Project/applicationservers/WebSphere7 directory.

Figure 15-87 The message that is displayed after a successful repackaging of the SSP

14. Next, you redeploy the repackaged SSP. Complete the instructions in the *Deploying a repackaged SSP archive* topic of the IBM Operational Decision Manager Version 8.0.1 Information Center:

    `http://pic.dhe.ibm.com/infocenter/dmanager/v8r0m1/index.jsp?topic=%2Fcom.ibm.wodm.family.config.zos%2Ftopics%2Ftsk_ds_redeploy_dvs_archive.html`

15.4.4 Publishing the insurance eligibility project in Decision Center

Complete the following steps to publish the insurance eligibility project in Decision Center:

1. When the repackaged SSP is deployed in your application server, the next step is to publish the insurance-rules project in Decision Center so that the business users can access it and define a test suite for it. In the Rule Explorer, right-click the **insurance-rules** project and select **Decision Center** → **Connect** (Figure 15-88).

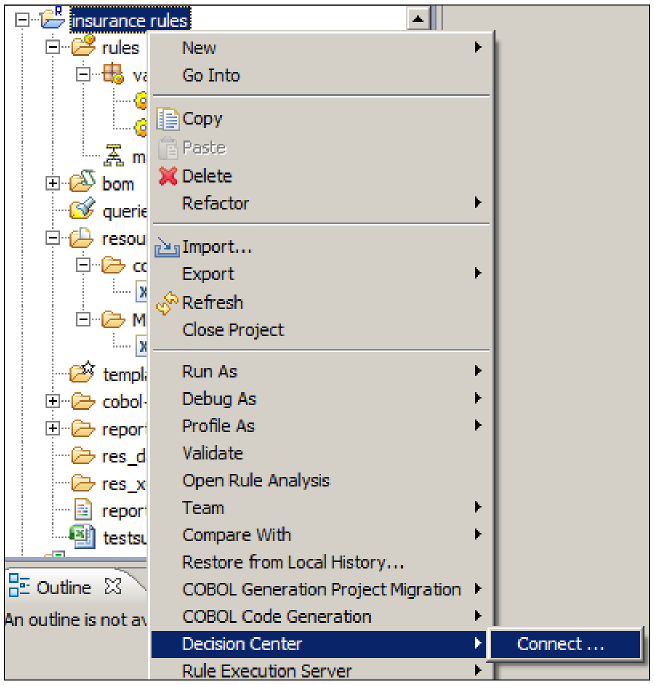
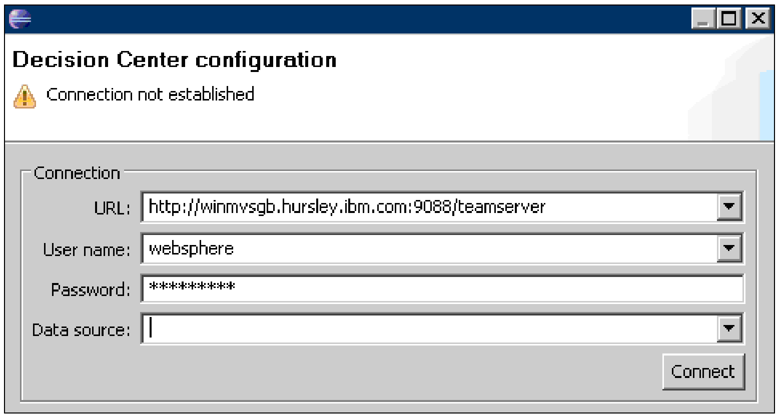


Figure 15-88 Selecting Decision Center → Connect

2. This action displays the Decision Center configuration window. Update this content so that the connection information matches the current Decision Center deployment (Figure 15-89). Enter the URL, user name, and password. Click **Connect**.



Decision Center configuration

Connection not established

Connection

URL: http://winmvsgb.hursley.ibm.com:9088/teamserver

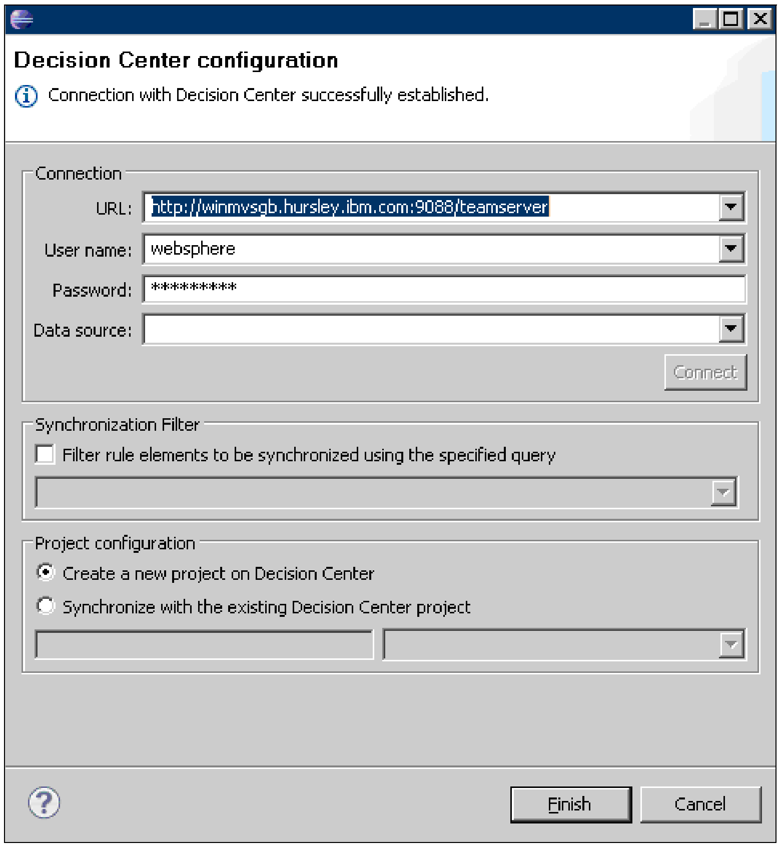
User name: websphere

Password: **********

Data source:

Figure 15-89 Connection information for a Decision Center configuration

3. Figure 15-90 shows that a connection is established. After the connection to Decision Center is successfully established, select the operation to perform in the Project configuration section (Figure 15-90). In this case, you want to create the project in Decision Center, so select **Create a new project on Decision Center**, and click **Finish**.


Decision Center configuration

Connection with Decision Center successfully established.

Connection

URL: http://winmvsgb.hursley.ibm.com:9088/teamserver

User name: websphere

Password: **********

Data source:

Synchronization Filter

☐ Filter rule elements to be synchronized using the specified query

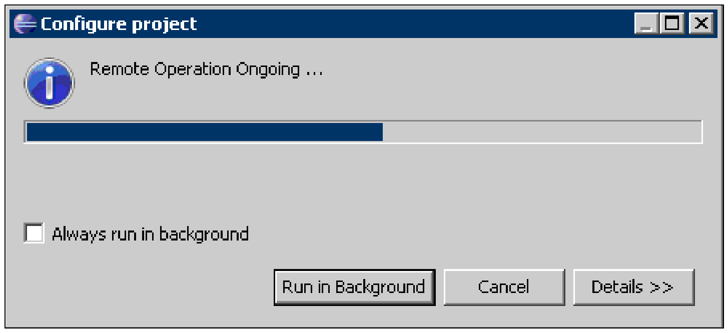
Project configuration

● Create a new project on Decision Center

○ Synchronize with the existing Decision Center project

Figure 15-90 Creating a project on Decision Center from Rule Designer

4. A pop-up window opens to inform you of the progress of the publication of your project in Decision Center (Figure 15-91).


Configure project

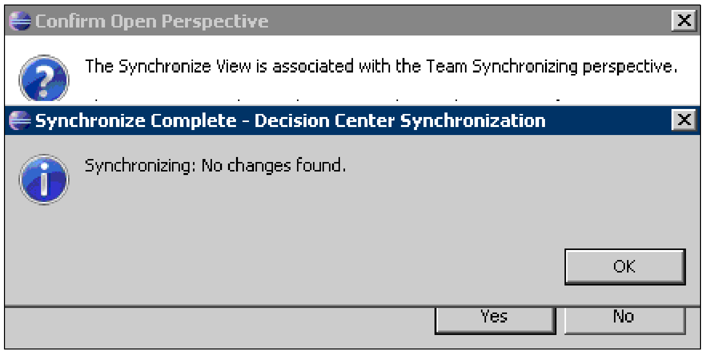
Remote Operation Ongoing ...

☐ Always run in background

Figure 15-91 Publishing a project in Decision Center

5. After the publication to Decision Center is complete, the first pop-up window opens to give you the option to switch to the Team Synchronizing perspective (ignore this option and stay on the Rules perspective).

6. A second pop-up window opens to inform you that the synchronization is complete. The projects are now the same in both Rule Designer and Decision Center (Figure 15-92).


Confirm Open Perspective

The Synchronize View is associated with the Team Synchronizing perspective.

Synchronize Complete - Decision Center Synchronization

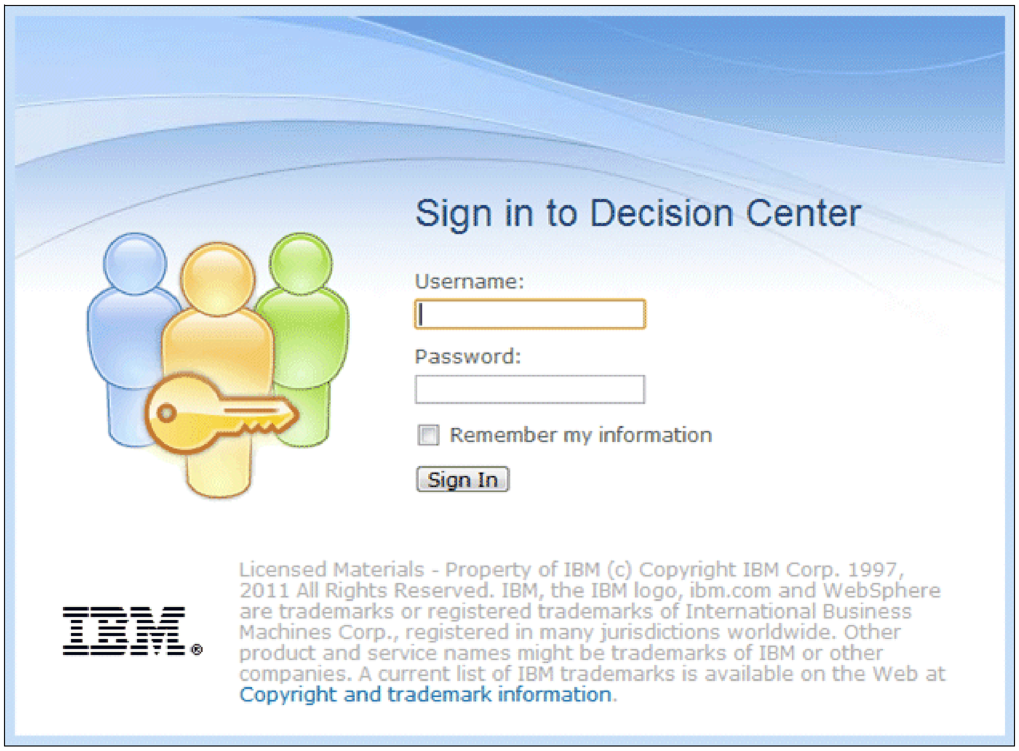
Synchronizing: No changes found.

Figure 15-92 Confirmation that the publication in Decision Center is complete

15.4.5 Configuring Decision Center to use the SSP to run tests and simulations

Complete the following steps to configure Decision Center to use the SSP for running tests and simulations:

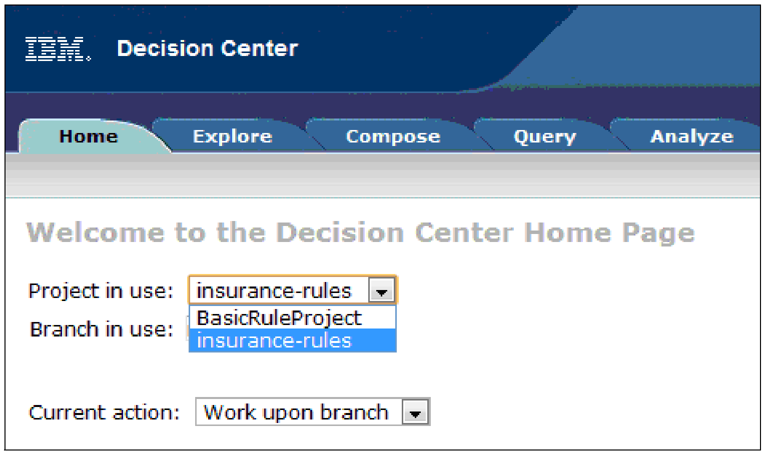
1. Before business users can define and run a test suite for the insurance-rules project in Decision Center, a few administrative operations must be completed. Open a web browser to the Decision Center URL and sign in as a Decision Center administrator (Figure 15-93).


Sign in to Decision Center

Licensed Materials - Property of IBM (c) Copyright IBM Corp. 1997, 2011 All Rights Reserved. IBM, the IBM logo, ibm.com and WebSphere are trademarks or registered trademarks of International Business Machines Corp., registered in many jurisdictions worldwide. Other product and service names might be trademarks of IBM or other companies. A current list of IBM trademarks is available on the Web at Copyright and trademark information.

Figure 15-93 Signing on to Decision Center

2. On the Decision Center Home page, select the newly published **insurance-rules** project as the project in use (Figure 15-94). Then, click the **Configure** tab to display the Configure menu for Decision Center.



Welcome to the Decision Center Home Page

Project in use: insurance-rules

BasicRuleProject

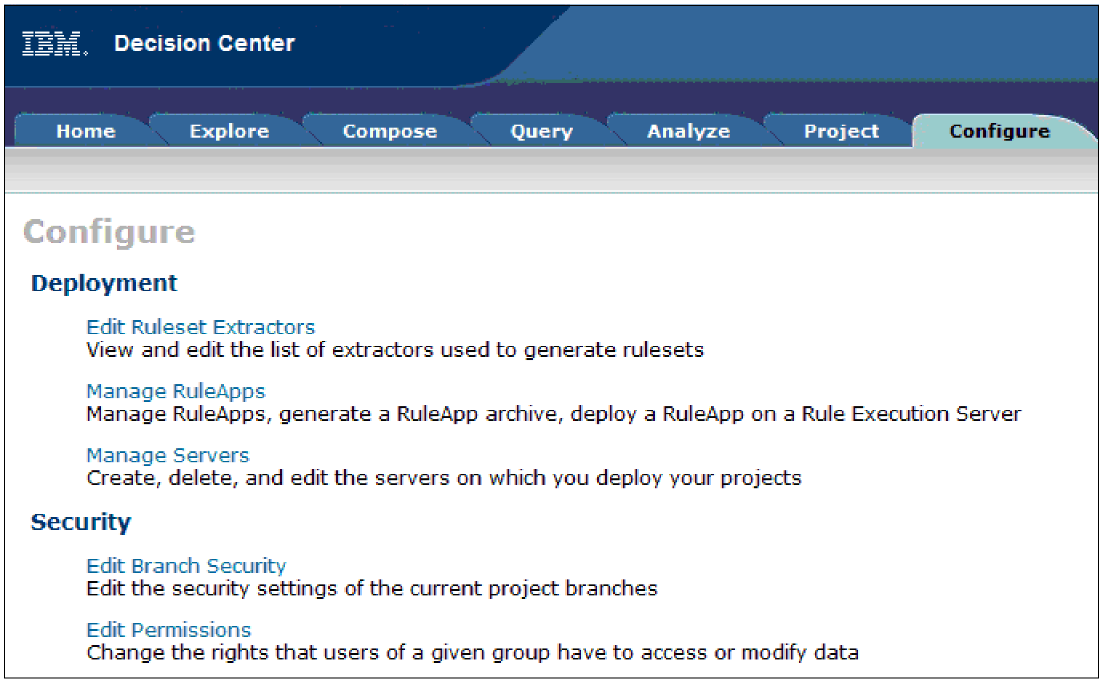
insurance-rules

Branch in use:

Current action: Work upon branch

Figure 15-94 Selecting the insurance-rules project in Decision Center

3. On the Configure menu for Decision Center (Figure 15-95), click the **Manage Servers** link.



Configure

Deployment

Edit Ruleset Extractors
View and edit the list of extractors used to generate rulesets

Manage RuleApps
Manage RuleApps, generate a RuleApp archive, deploy a RuleApp on a Rule Execution Server

Manage Servers
Create, delete, and edit the servers on which you deploy your projects

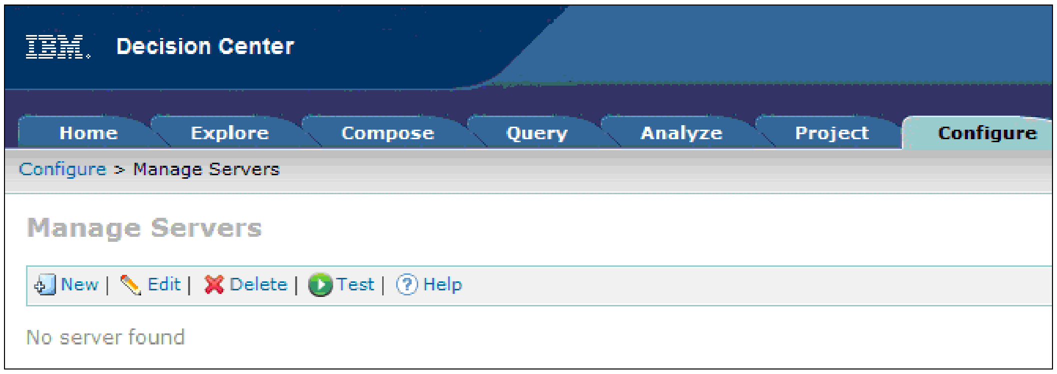
Security

Edit Branch Security
Edit the security settings of the current project branches

Edit Permissions
Change the rights that users of a given group have to access or modify data

Figure 15-95 Configure menu for Decision Center

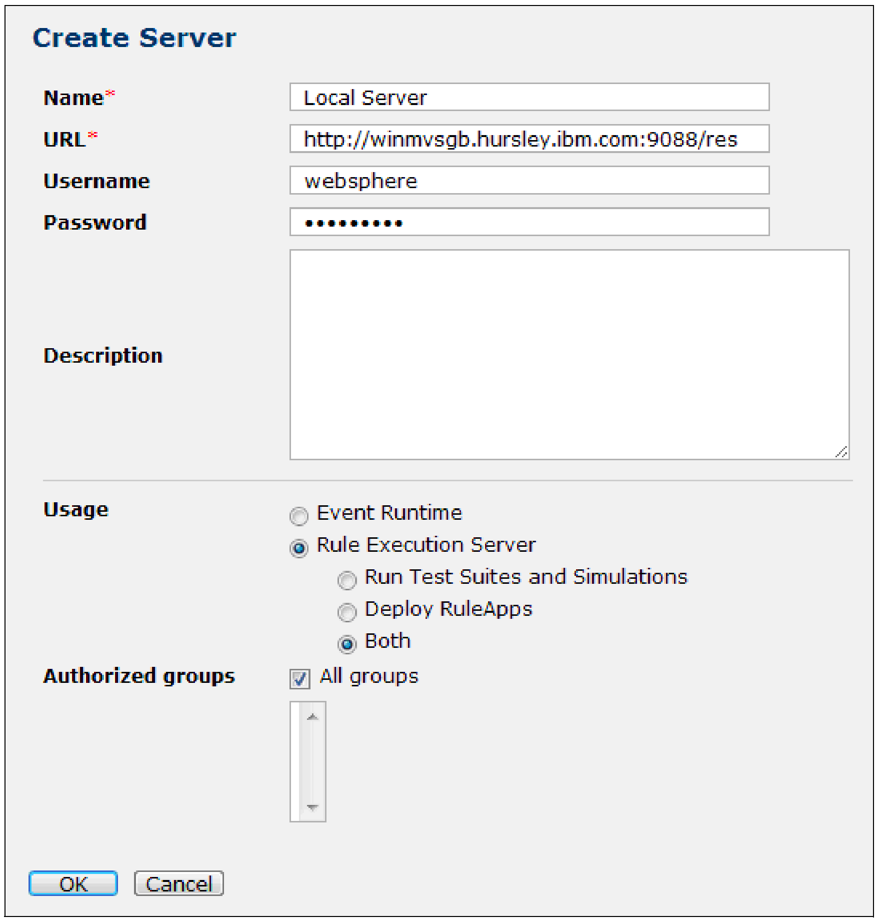
4. Add the SSP to the list of servers known to Decision Center to run tests and simulation. On the Manage Servers page (Figure 15-96), click **New** to create a server definition. This operation is only performed one time. Then, the added SSP is available for all projects in Decision Center.



No server found

Figure 15-96 The Manage Servers page in Decision Center

5. On the Create Server page (Figure 15-97), enter the connection information for your server (URL and credentials). Select the usage options of **Rule Execution Server** and **Both**. Select the **All groups** Authorized groups option. When finished, click **OK** to save this new server definition.

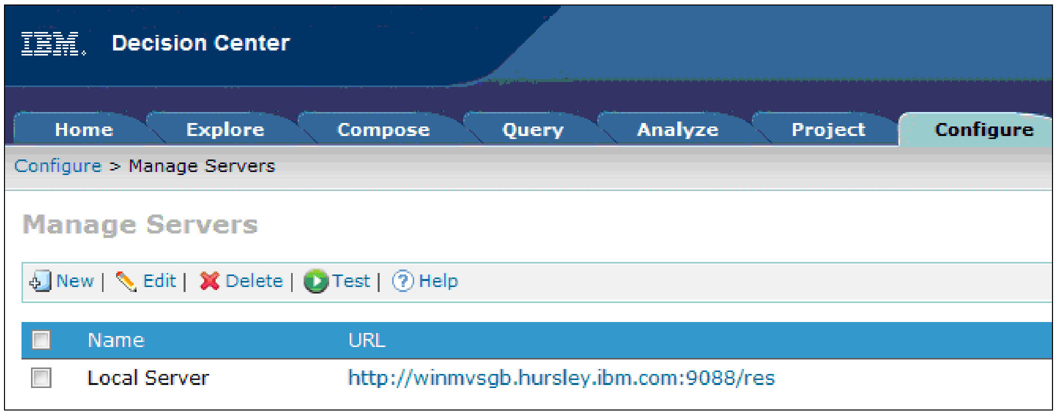


Create Server

Name*	Local Server
URL*	http://winmvsgb.hursley.ibm.com:9088/res
Username	websphere
Password	●●●●●●●●●
Description	
Usage	Event Runtime [✓ Rule Execution Server] Run Test Suites and Simulations / Deploy RuleApps / [✓ Both]
Authorized groups	[✓ All groups]

Figure 15-97 Create Server page in Decision Center

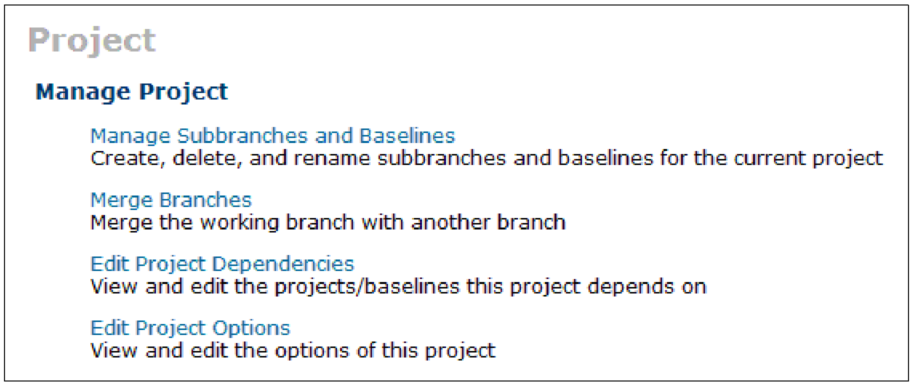
6. The Manage Servers page opens again. Its content is updated with the new server information (Figure 15-98). Click the **Project** tab to display the project menu that applies to the currently selected project.



Name	URL
Local Server	http://winmvsgb.hursley.ibm.com:9088/res

Figure 15-98   Manage Servers page displays information for a server definition

7. Now that a server definition is available to the business users, ensure that all the scenario suite formats that are used to define scenario suites for the insurance-rules project are enabled. Click the **Edit Project Options** link to edit the list of scenario suite formats that are available for the current project (Figure 15-99).



Project

Manage Project

Manage Subbranches and Baselines
Create, delete, and rename subbranches and baselines for the current project

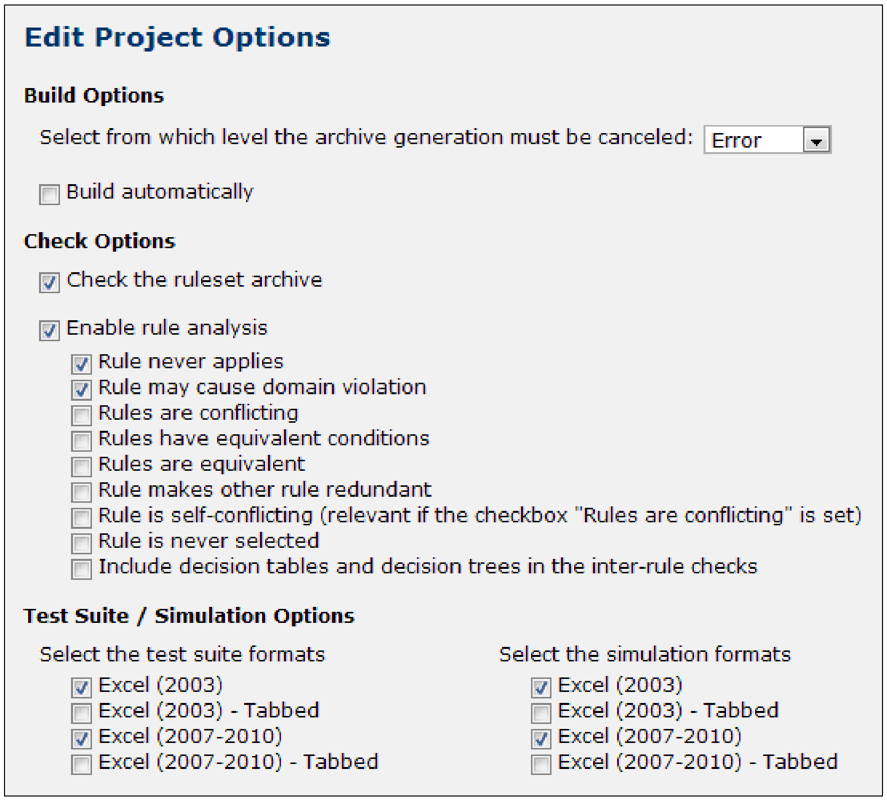
Merge Branches
Merge the working branch with another branch

Edit Project Dependencies
View and edit the projects/baselines this project depends on

Edit Project Options
View and edit the options of this project

Figure 15-99   Project menu in Decision Center

8. On the Edit Project Options page (Figure 15-100), in the Test Suite / Simulation Options section, select the test suite and simulation formats that you want to make available for the current project. If you used the Excel (2003) format to create the test suite in Rule Designer, ensure that you enable it for the project in Decision Center. Only the Excel (2007-2010) format is enabled, by default. After you complete the format selection, click **OK** to save the changes.


Edit Project Options

Build Options

Select from which level the archive generation must be canceled: Error

- [ ] Build automatically

Check Options

- [x] Check the ruleset archive
- [x] Enable rule analysis
  - [x] Rule never applies
  - [x] Rule may cause domain violation
  - [ ] Rules are conflicting
  - [ ] Rules have equivalent conditions
  - [ ] Rules are equivalent
  - [ ] Rule makes other rule redundant
  - [ ] Rule is self-conflicting (relevant if the checkbox "Rules are conflicting" is set)
  - [ ] Rule is never selected
  - [ ] Include decision tables and decision trees in the inter-rule checks

Test Suite / Simulation Options

Select the test suite formats
- [x] Excel (2003)
- [ ] Excel (2003) - Tabbed
- [x] Excel (2007-2010)
- [ ] Excel (2007-2010) - Tabbed

Select the simulation formats
- [x] Excel (2003)
- [ ] Excel (2003) - Tabbed
- [x] Excel (2007-2010)
- [ ] Excel (2007-2010) - Tabbed

Figure 15-100 Edit Project Options in Decision Center

15.4.6 Creating a test suite in Decision Center

Now that a server definition is created in Decision Center and the scenario suite formats to define test suites and simulation are enabled for the project, create a test suite in Decision Center. Then, run it and display its execution report.

Complete the following steps to create a test suite in Decision Center, run it, and display the execution report:

1. In the Decision Center, click the **Compose** tab, as shown in Figure 15-101.

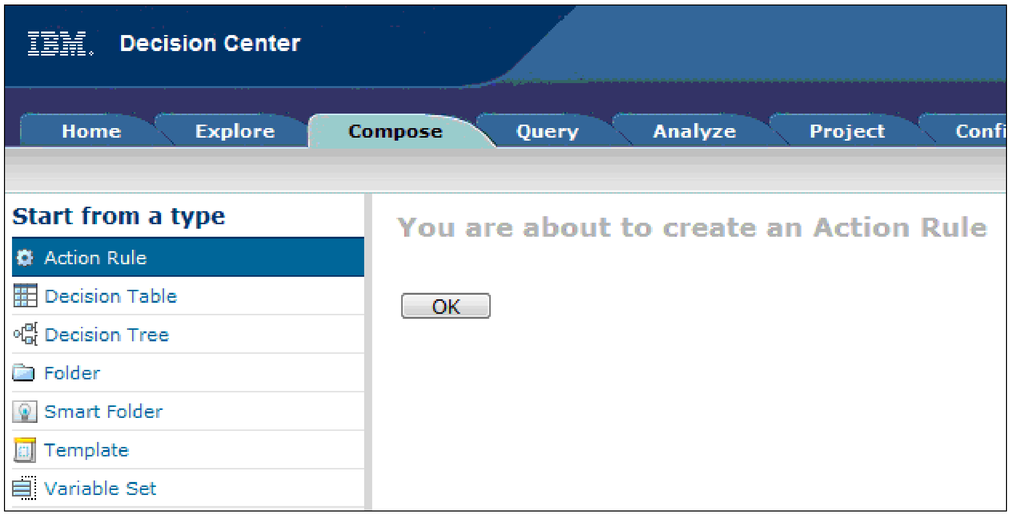


Figure 15-101 Compose tab with the Action Rule type default selection

2. By default, the first time that you open the Compose page, Decision Center defaults to creating an Action Rule (Figure 15-101). To create a Test Suite instead, click **Test Suite**.

   After selecting Test Suite as the type of artifact to create, the right side of the page is updated. To start the new test suite wizard, click **OK** on the right side of the page.

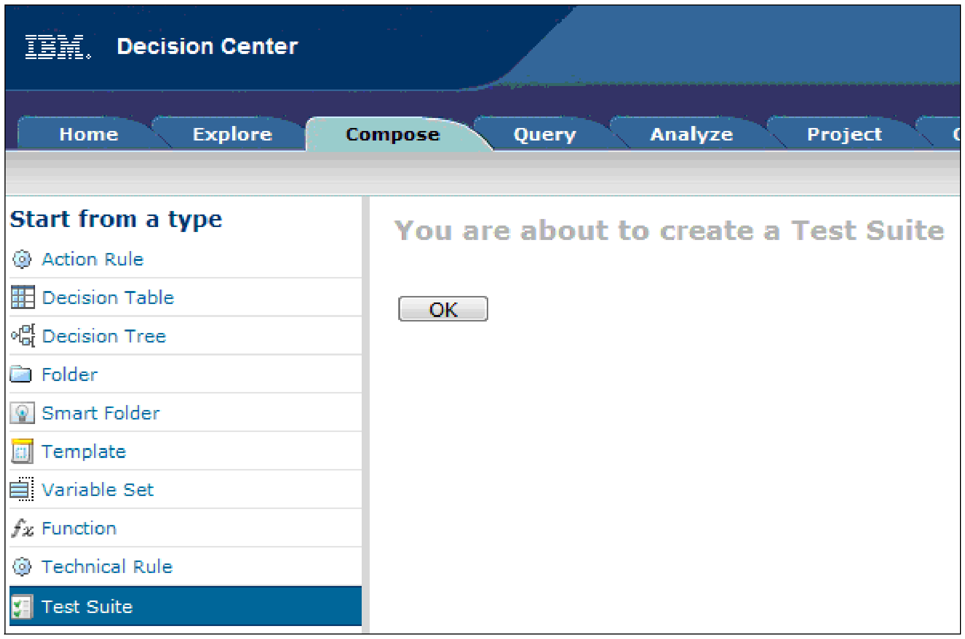
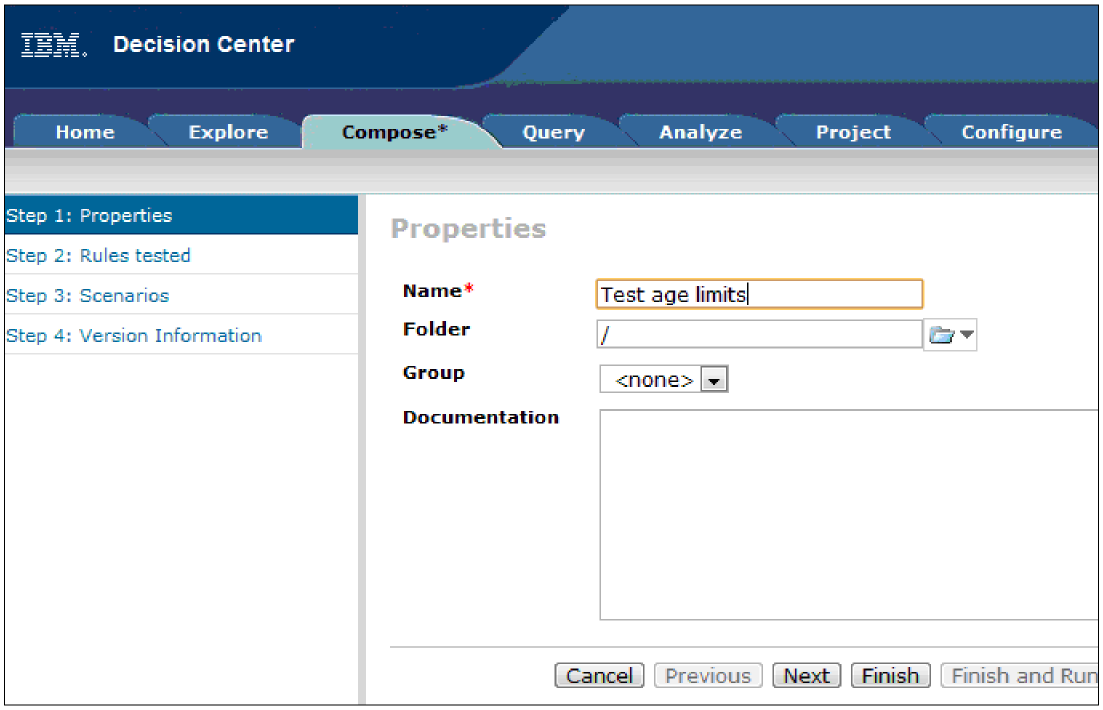


Figure 15-102 Compose page with the Test Suite type selected

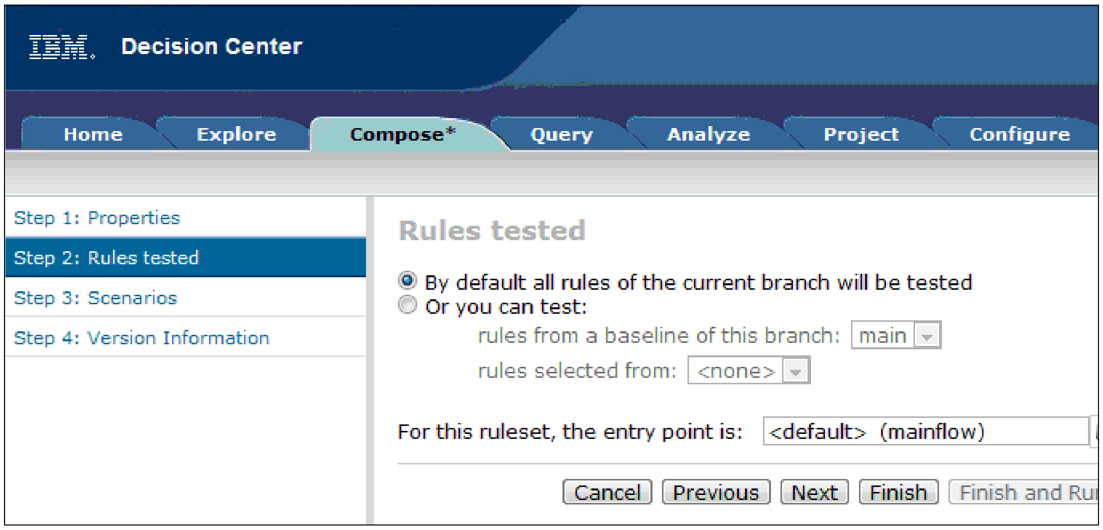
3. On the Properties page that is shown in Figure 15-103, enter the name, folder (location), group, and documentation for the new test suite. Click **Next**.



Name* Test age limits
Folder /
Group <none>
Documentation

Figure 15-103 Test suite creation wizard in Decision Center: Properties

4. On the Rules tested page that is shown in Figure 15-104, you can choose to test the ruleset in its current status or select another version of the ruleset from a baseline. Also, you can limit the testing to a subset of the rules and select the entry point for the ruleset. Leave the default options that are selected, and click **Next**.

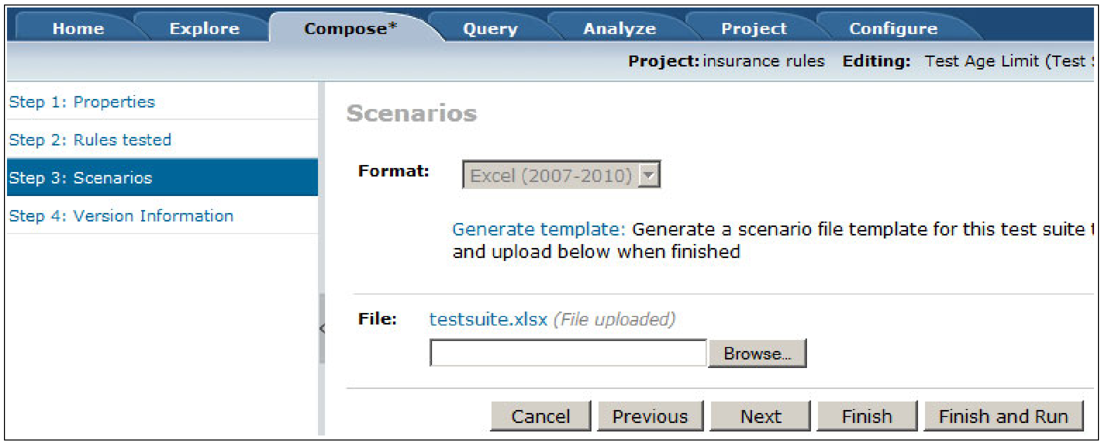


[✓ By default all rules of the current branch will be tested]
Or you can test:
rules from a baseline of this branch: main
rules selected from: <none>

For this ruleset, the entry point is: <default> (mainflow)

Figure 15-104 Test suite creation wizard in Decision Center: Rules tested

5. On the Scenarios page that is shown in Figure 15-105, select the scenario suite format to use. Select the scenario suite format that was used to create the test suite in Rule Designer. Click **Choose File** to upload the Excel test suite that was created with Rule Designer. Then, click **Finish and Run** to save the new test suite.



Project: insurance rules **Editing:** Test Age Limit (Test :

Step 1: Properties
Step 2: Rules tested
Step 3: Scenarios
Step 4: Version Information

Scenarios

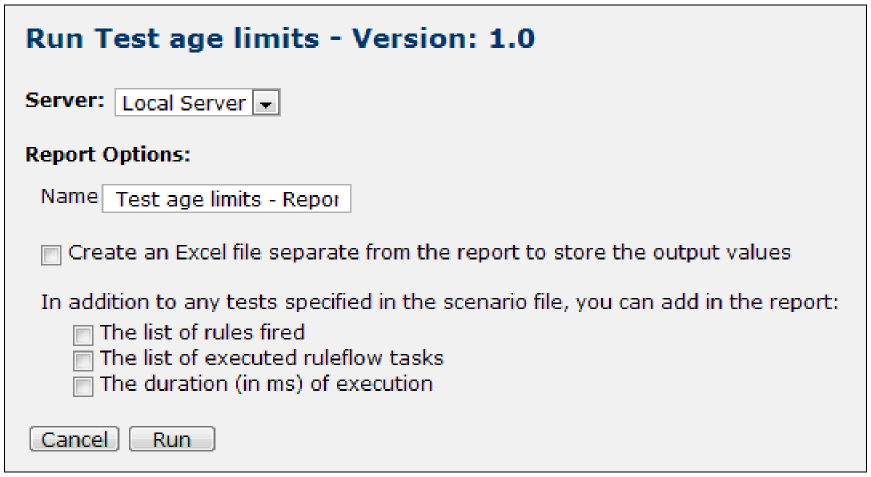
Format: Excel (2007-2010)

Generate template: Generate a scenario file template for this test suite t and upload below when finished

File: testsuite.xlsx *(File uploaded)*

Figure 15-105 Test suite creation wizard in Decision Center: Scenarios

6. The Run Test age limits page is displayed (Figure 15-106).



Run Test age limits - Version: 1.0

Server: Local Server

Report Options:

Name Test age limits - Repor

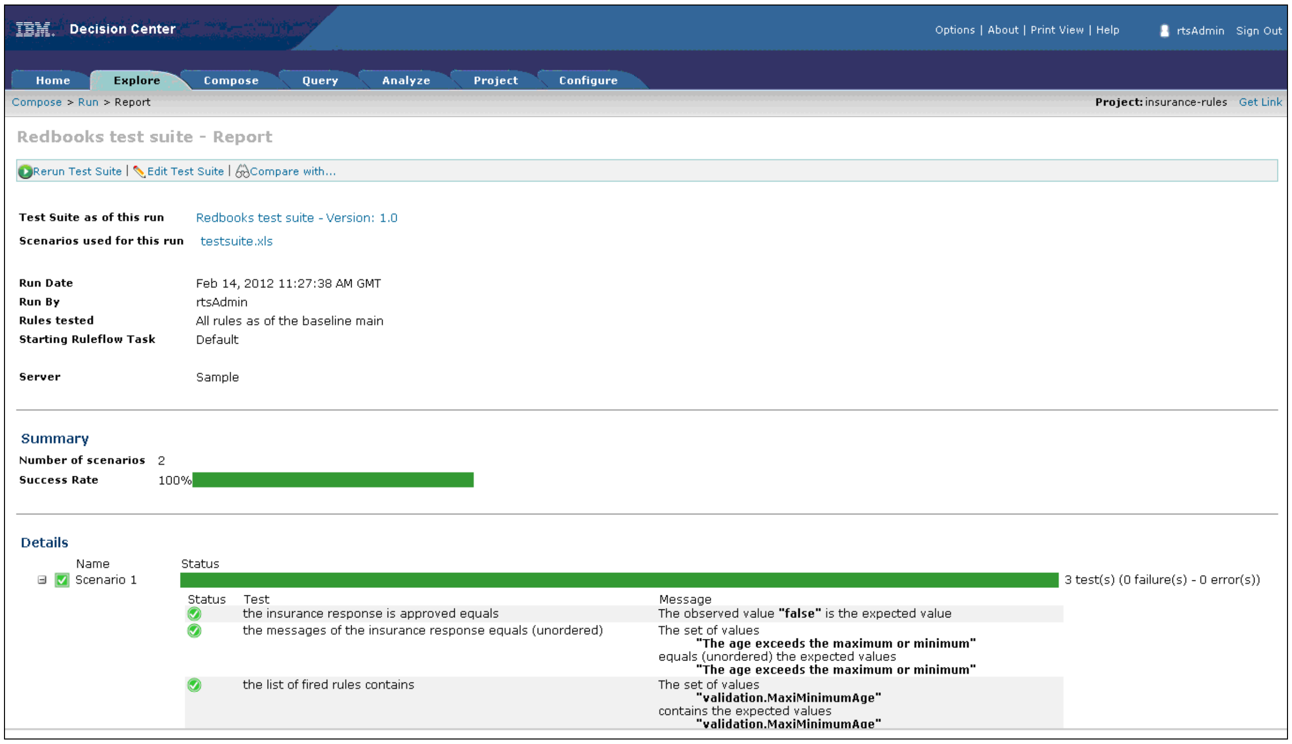
☐ Create an Excel file separate from the report to store the output values

In addition to any tests specified in the scenario file, you can add in the report:

☐ The list of rules fired
☐ The list of executed ruleflow tasks
☐ The duration (in ms) of execution

Figure 15-106 The Run Test suite page in Decision Center

7. To run the test suite, click **Run**, and wait until the test report is displayed (Figure 15-107).



Redbooks test suite - Report

Test Suite as of this run	Redbooks test suite - Version: 1.0
Scenarios used for this run	testsuite.xls
Run Date	Feb 14, 2012 11:27:38 AM GMT
Run By	rtsAdmin
Rules tested	All rules as of the baseline main
Starting Ruleflow Task	Default
Server	Sample

Summary

Number of scenarios 2

Success Rate 100%

Details

Name Status

Scenario 1 — 3 test(s) (0 failure(s) - 0 error(s))

Status	Test	Message
✓	the insurance response is approved equals	The observed value **"false"** is the expected value
✓	the messages of the insurance response equals (unordered)	The set of values **"The age exceeds the maximum or minimum"** equals (unordered) the expected values **"The age exceeds the maximum or minimum"**
✓	the list of fired rules contains	The set of values **"validation.MaxiMinimumAge"** contains the expected values **"validation.MaxiMinimumAge"**

Figure 15-107 Test suite execution report in Decision Center

16

Configuring the Rules Execution Servers for z/OS console with virtual IP addressing

This chapter describes the use of virtual IP addressing (VIPA) to allow zRules consoles to manage Rules Execution Servers for z/OS (zRES) on multiple logical partitions (LPARs) or multiple systems. This allows for uninterrupted rules deployment to continue during LPAR maintenance or downtime.

This chapter uses a scenario that consists of two LPARs, six rule execution environments, two zRES consoles, and one database, as shown in the diagram in Figure 16-3 on page 316.

The following topics are covered in this chapter:

© Copyright IBM Corp. 2013. All rights reserved.

16.1 Overview of a multiple LPAR environment

This environment is an Operational Decision Manager for z/OS system that consists of multiple zRES instances, which reside on multiple LPARs that are supported by a single zRules console that is connected to a single database. The ability to publish changed rules applications to a running zRES without having to restart the server, which is called *hot deployment* (16.1.1, “Hot deployment of rules in Operational Decision Manager” on page 312), from the zRules console is supported by using an internal asynchronous “publish/subscribe” (pub/sub) notification mechanism.

An example topology is a configuration that consists of two LPARs, LPAR A and LPAR B, which reside in PLEX 1**,** as shown in Figure 16-1.

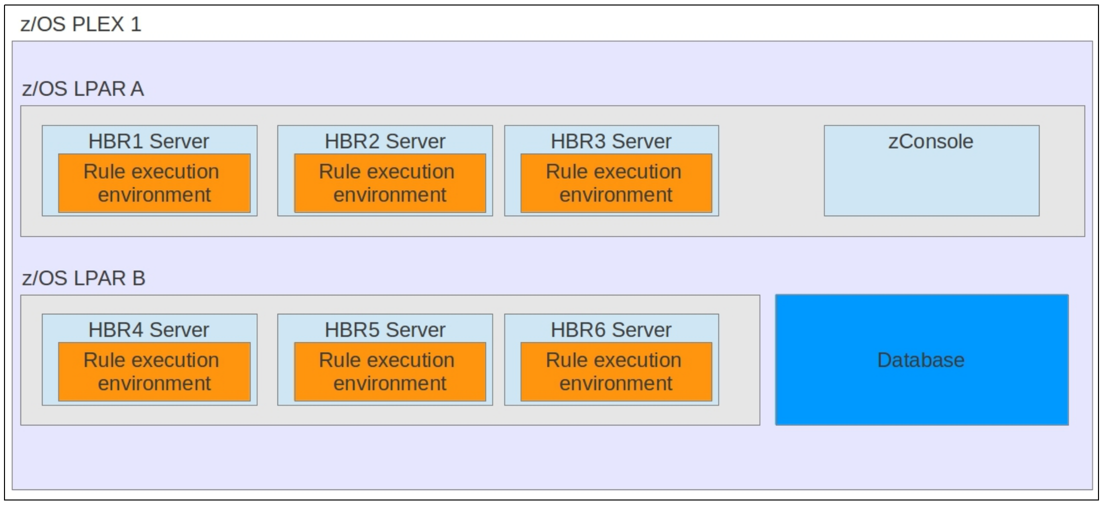

z/OS PLEX 1
z/OS LPAR A
HBR1 Server
Rule execution environment
HBR2 Server
Rule execution environment
HBR3 Server
Rule execution environment
zConsole
z/OS LPAR B
HBR4 Server
Rule execution environment
HBR5 Server
Rule execution environment
HBR6 Server
Rule execution environment
Database

Figure 16-1 Multiple LPAR Operational Decision Manager system

There are two styles of rules application deployment:

- Hot deployment of rules
- Cold deployment of rules

16.1.1 Hot deployment of rules in Operational Decision Manager

Hot deployment of rules in Operational Decision Manager is the ability to publish changed rules applications to a running rules execution server without having to restart the server. The deployment feature works by deploying the Rule Application using the zRules console.

Rule deployment can be performed directly from the zRules console, Rule Designer, Rule Team server, or Decision Center Business console. The zRules console uses asynchronous messages to notify all rule execution environments that are registered to that zRules console. The next invocation of that rule causes the zRES to go to the database to load the newest level of that rule application.

After the zRES reads the rules from the database, it continues to use that version of the rules until it is notified of another update. Figure 16-2 shows information flows that describe how deployment, storage, registration, and notification occur.

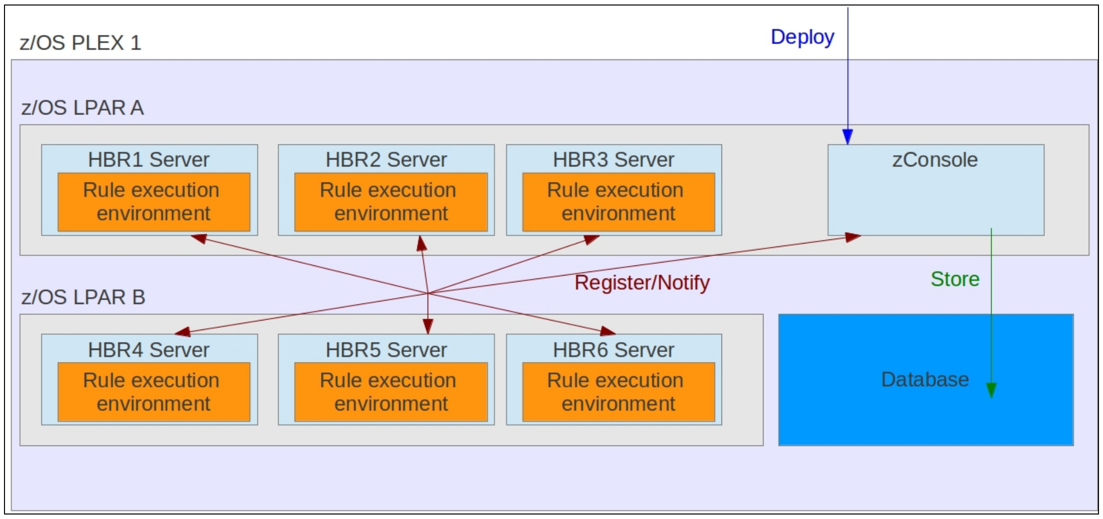

z/OS PLEX 1
Deploy
z/OS LPAR A
HBR1 Server
Rule execution environment
HBR2 Server
Rule execution environment
HBR3 Server
Rule execution environment
zConsole
Register/Notify
Store
z/OS LPAR B
HBR4 Server
Rule execution environment
HBR5 Server
Rule execution environment
HBR6 Server
Rule execution environment
Database

Figure 16-2 Hot deployment in a multiple LPAR environment

16.1.2 Cold deployment of rules in Operational Decision Manager

Cold deployment is performed by using a deployment method that *directly* updates the rules that are stored in the database. The rule execution environments load rules from the database, which means that on the first invocation of a ruleset, the rule execution environment goes to the database and retrieves the latest copy of the ruleset. On any subsequent invocations, it reuses the *same* ruleset *unless* it is notified of an updated version. To refresh rulesets in the rule execution environment to pick up ruleset changes, the rule execution server must be restarted.

16.2 Using Virtual IP addressing to allow more than one zRules console to be used

Any zRES that located on same LPAR as the zRules console can restart the zRules console, if the console fails. However, if the LPAR that hosts the zRules console fails, there is no means for the remaining zRESs to start (or restart) a zRules console. Consequently, hot deployment is not available until the LPAR that hosts the zRules console is brought back online.

This section explains the setup and behavior of a zRules console that uses VIPA to manage zRESs on multiple LPARs (or systems), therefore allowing continued hot deployment during LPAR maintenance or downtime.

The following sections describe the various elements that are associated with the loss of an LPAR that hosts a zRES and a ZRules console:

- 16.2.1, “What happens if the LPAR that hosts the zRules console fails” on page 314
- 16.2.2, “Using virtual IP addressing” on page 314
- 16.2.3, “How VIPA maintains hot deployment” on page 315

16.2.1 What happens if the LPAR that hosts the zRules console fails

If the LPAR that hosts the zRules console fails, for example, LPAR A in Figure 16-2 on page 313, there is no means for the remaining zRES, which is on LPAR B, to start (or restart) the zRules console (zConsole). Consequently, the hot deployment of rules is available until LPAR A is brought back online.

The rule execution environments on LPAR B still continue to execute the rules from the database. Rule deployment *is* still possible but only by using a cold deployment method because hot deployment is unavailable.

16.2.2 Using virtual IP addressing

Hot deployment can be more flexible in a production environment by using virtualization to allow the zRules console to be run on more than one LPAR. This situation is possible because all of the zRules console communication is performed using TCP/IP.

To use a virtual IP address (VIPA), define a virtual host that maps to LPAR A and LPAR B, which allows both LPARs to share the zRules console communication port and the zRules console HTTP port.

Note: Do not use a balancing algorithm on the port sharing because you are not sharing the load.

Send all requests to one server until it becomes unavailable. So, for the example that is used here, all traffic is sent to LPAR A until the connection to LPAR A is lost, and then all traffic be sent or redirected to LPAR B.

There are many ways in which VIPA can be set up for a configuration. The following IBM publication, *z/OS Communications Server: IP Configuration Guide,* SC31-8775-20, provides all the required information to decide how to set up VIPA:

http://publib.boulder.ibm.com/infocenter/zos/v1r13/topic/com.ibm.zos.r13.halz002/f1a1b3b1151.htm#wq464

Example 16-1 and Example 16-2 are extracts from the TCP PARMS dataset from each LPAR. You can use these extracts for an example setup, and they can help you with several default settings.

Example 16-1 TCP PARMS for LPAR A

```
VIPADYNAMIC
  VIPADEFINE MOVE IMMED 255.255.255.252 9.20.9.53
    VIPADISTRIBUTE 9.20.9.53 PORT
  24159 34159 44159
  DESTIP ALL
ENDVIPADYNAMIC
```

Example 16-2 TCP PARMS for LPAR B

```
VIPADYNAMIC
 VIPABACKUP 100 MOVE IMMEDIATE 255.255.255.252 9.20.9.53
ENDVIPADYNAMIC
```

In Example 16-1 on page 314 and Example 16-2 on page 314, the following entries are underlined twice:

- `9.20.9.53`: IP address for VIPA
- `24159`: `SSPPORT` port number (not required for VIPA support)
- `34159`: `CONSOLEPORT` port number
- `44159`: `CONSOLECOM` port number

Descriptions of the ports are in Table B-1 on page 328.

The "share port" setup is also explained in the IBM publication, *z/OS Communications Server: IP Configuration Guide*, SC31-8775-20. The configuration that is shown in Example 16-3 is created by adding the following configuration to the TCP PARMS dataset on both LPARs.

Example 16-3 TCP PARMS for LPAR A and LPAR B

```
IPCONFIG DYNAMICXCF 192.168.x.x 255.255.255.0 1
PORT
34159 TCP OMVS SHAREPORT ; zRules Console Port
44159 TCP OMVS SHAREPORT ; zRules Console ComPort
```

After the VIPA and share port are set up, the zRES needs to use the virtual host. You must modify the two zRES's parms members. In the ++HBRWDS++.SHBRPARM(HBRCMMN) member, modify the **`HBRCONSOLECOMPORT`** and **`HBRCONSOLECOMHOST`** parameters to use the virtual host and share port, as demonstrated in Example 16-4.

Example 16-4 HBRCMMN parms for each zRES

```
HBRCONSOLECOMPORT=44159
HBRCONSOLECOMHOST=zodm.hursley.ibm.com
```

In the ++HBRWDS++.SHBRPARM(HBRCNSL) member, modify the parameter **`HBRCONSOLEPORT`** to the shared port that is defined as shown in Example 16-5.

Example 16-5 HBRCNSL parm for each zRES

```
HBRCONSOLEPORT=34159
```

After these modifications, you must restart all zRESs.

16.2.3 How VIPA maintains hot deployment

This section describes the use of VIPA in the scenario that is presented in Figure 16-1 on page 312. The following modes of operation are described:

- "Normal operation"
- "Failure of LPAR A" on page 317
- "Restoration of LPAR A" on page 318
- "Return to the normal operating environment" on page 318

Normal operation

With all the servers started, using the VIPA and shared ports, the zRules console works normally. All traffic is routed to the zRules console on LPAR A, and therefore, all servers are registered to the one zRules console on LPAR A.

Share port enables the two zRules console address spaces to start and bind to the port. See Figure 16-3 on page 316, which depicts an Operational Decision Manager for z/OS configuration using VIPA.

The following four actions are performed for the hot deployment feature:

- Deployment
- Storage
- Registration
- Notification

The registration is performed during zRES startup. If the connection is lost, the zRES attempts a reconnection. If the connection attempt fails, the zRES attempts to reconnect every 10 seconds.

Deployment uses the *virtual host* and the *shared console port*. The zRules console, zConsole, stores rulesets in the database and notifies all connected rule execution environments, again, as though VIPA is not being used.

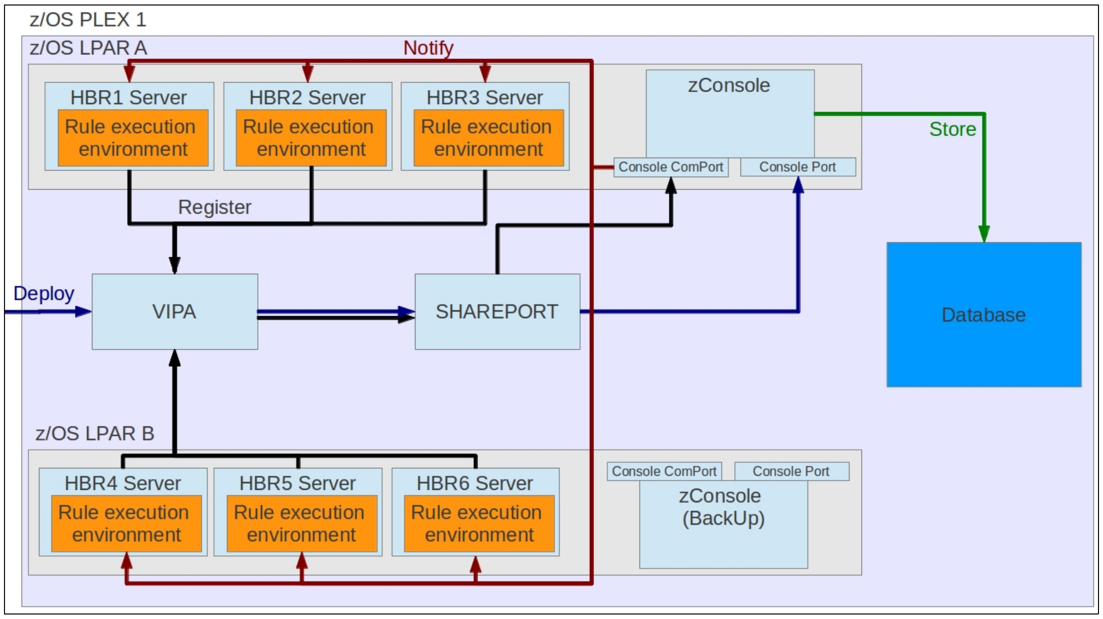


Figure 16-3 Configuration of VIPA using two zRules consoles

Failure of LPAR A

This section describes the sequence of events if a failure occurs that results in the loss of access to LPAR A. The scenario is depicted in Figure 16-4.

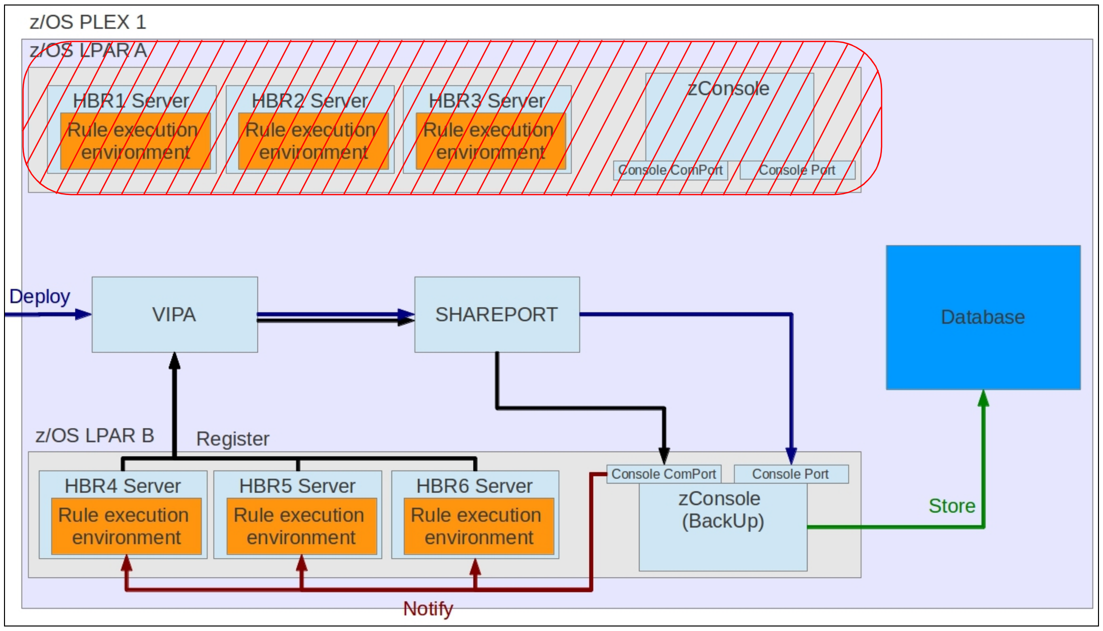


Figure 16-4 Failover to zConsole on LPAR B

The events occur in this sequence:

1. LPAR A fails.
2. VIPA routes all traffic aimed at the zRules console to the zRules console on LPAR B, which is zConsole (BackUp).
3. The rule execution environments, HBR4, HBR5, and HBR6, lose their connection to the LPAR A zRules console (zConsole) and try to reconnect.
4. The result of the reconnection attempt is an *almost instantaneous* connection to the zRules console that runs on LPAR B zConsole (BackUp), which ensures that the hot deployment can continue.

Standard rule execution and rule deployment continue from this point.

Important: There is no function to make the zRules console on LPAR B aware of any rule changes that are made by the zRules console that runs on LPAR A.

To work around the issue of this lack of awareness of rule changes made by another zRules console, you need to manually refresh the zRES console on LPAR B, which forces new rules to be visible.

To manually refresh the zRES console on LPAR B, click **Update RuleApps** on the Explorer tab in the zRules console, as shown in Figure 16-5.

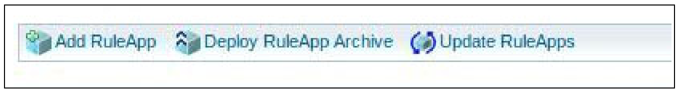


Figure 16-5 Update RuleApps on zRules console Explorer tab

Restoration of LPAR A

This section describes the scenario that is associated with the restoration of a failed LPAR (LPAR A in this example) that hosts the Operational Decision Manager execution environments and a zRES console.

The failed LPAR needs to be brought online normally, and Operational Decision Manager rule execution environments also need to be started normally.

Important: The configuration will *not* automatically return operation to the zRules console on LPAR A after LPAR A is restored.

Traffic continues to be routed to zConsole (BackUp) on LPAR B, uninterrupted, until an intervention occurs. See Figure 16-6.

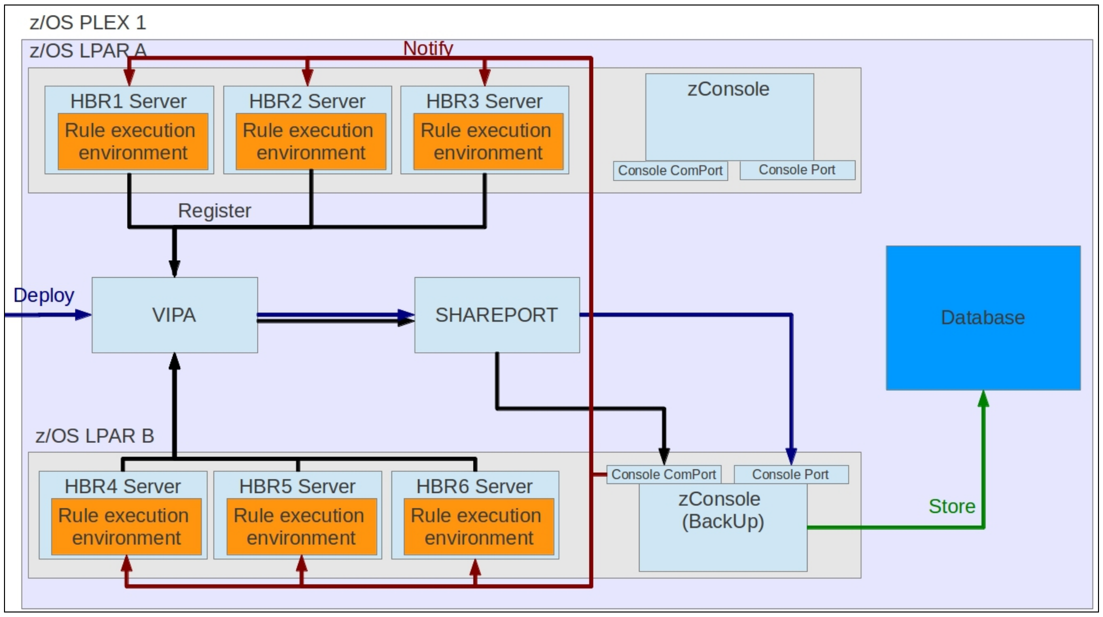


Figure 16-6 LPAR A restored

Return to the normal operating environment

To return the Operational Decision Manager system to the original mode of operation, which is routing traffic to LPAR A, the zRules console that runs on LPAR B, zConsole (BackUp), needs to be restarted.

Note: By using this example, *fail back* is a deliberate act after LPAR maintenance or recovery is completed successfully. The behavior can be tailored in the VIPA setup to meet other configuration requirements.

The result of restarting the LPAR B zRES console, zConsole (BackUp), is shown in Figure 16-6 on page 318.

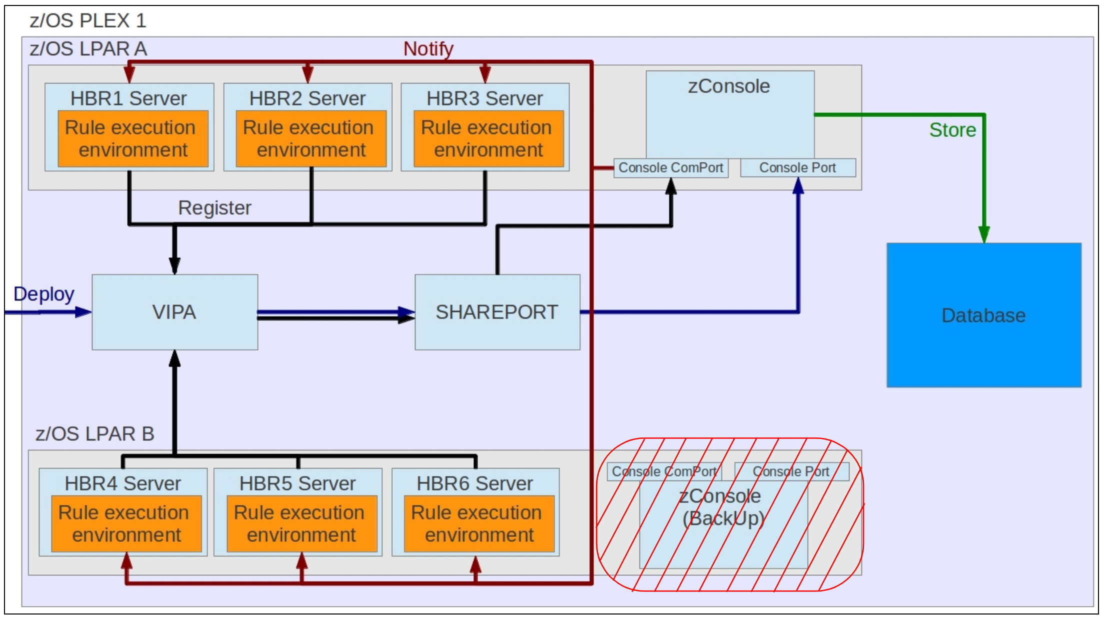


Figure 16-7 Operational Decision Manager system following a restart of zConsole (BackUp)

The result of restarting the zConsole (BackUp) is similar to the situation that is described in "Failure of LPAR A" on page 317:

1. VIPA routes all traffic that is aimed at a zRules console to the zRules console on LPAR A, zConsole.
2. The rule execution environments, HBR1, HBR2, and HBR3, lose their connection to the LPAR B zRules console, zConsole (BackUp) and try to reconnect.
3. The result of the attempt to reconnect is an almost instantaneous connection to the zRules console (originally used before the LPAR failure) that runs on LPAR A zConsole.

Hot deployment continues to be possible. A refresh might be required, as shown in Figure 16-5 on page 317. This time, you update RuleApps on the zRules console Explorer tab on the LPAR A zRES console, zConsole.

Part 3

Appendixes

Part 3 includes the following appendixes:

© Copyright IBM Corp. 2013. All rights reserved.

A

Calling out from a ruleset to a VSAM file to augment data

Occasionally, you are required to call out from a ruleset to augment the data that is required to make a decision. Only call out from a ruleset to augment data in the following circumstances:

- The data is not easily accessible to the application program.
- The decision cannot be made without the data.
- The data is required only in exceptional circumstances by the decision.

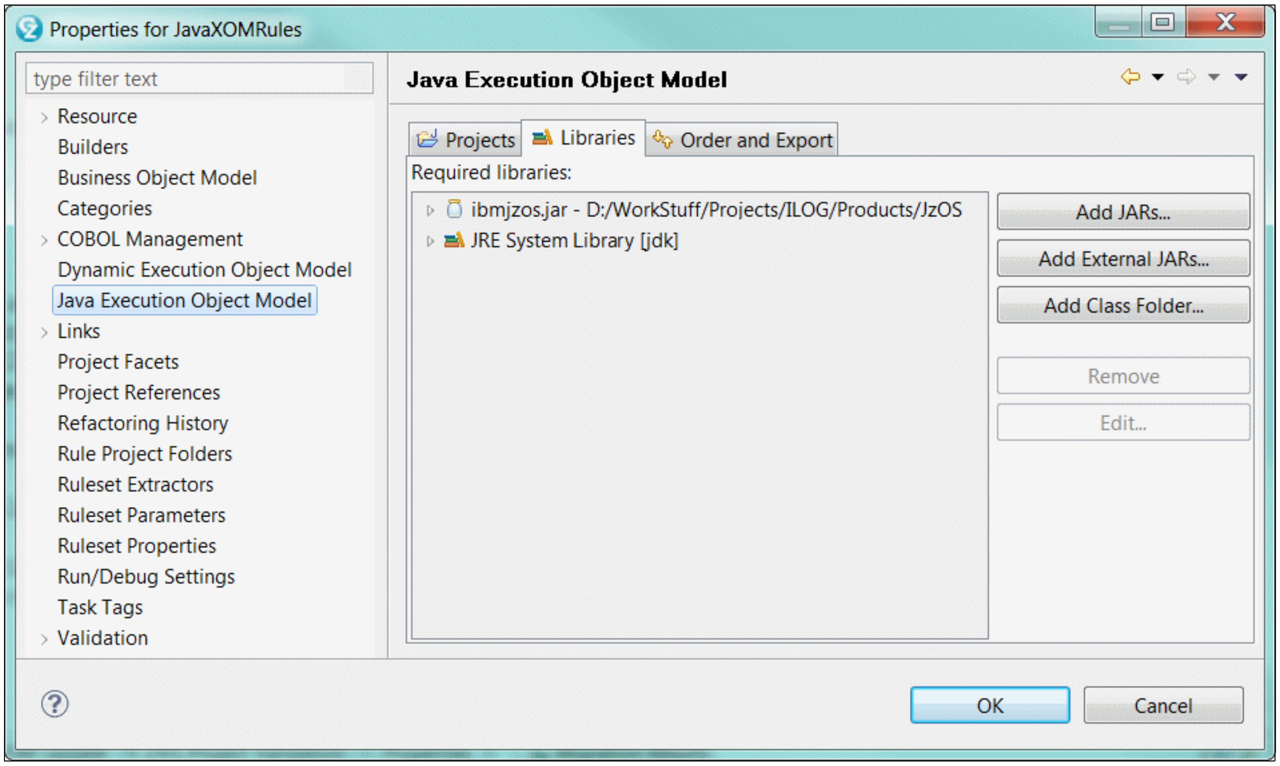
The JzOS libraries that come with IBM Java Runtime Environment 6 for z/OS provide a set of classes that can be used to access a record from a VSAM file. The following code shows an example of using the JzOS library classes to read a specified record from a VSAM file. You can use this code within a business object module (BOM) method to augment the data that is available to a decision. To use the JzOS classes within a BOM method, you must first copy the JzOS library to your computer to make it available to the rule project in which you want to author your rules.

The JzOS library is in your z/OS installation directory:

`<JAVA6_INSTALL_ROOT>/lib/ext/ibmjzos.jar`

© Copyright IBM Corp. 2013. All rights reserved.

To allow these classes to be available to your BOM authoring, you must add this JAR file as an external JAR to the Java Execution Model for your project. Use the project Properties panel, as shown in Figure A-1.



Required libraries:

- ibmjzos.jar - D:/WorkStuff/Projects/ILOG/Products/JzOS
- JRE System Library [jdk]

Figure A-1 Adding the JzOS JAR to the project class path

The class ReadKsdsVsam has a constructor and one method on it (Example A-1). This class is designed to read a single row from a key-sequenced data set (KSDS) format VSAM file, based on a supplied record key, and to return the corresponding row as a byte array. By knowing the format of the record structure, the required data can then be read out from the byte array and copied into local rule variables. The byte array can be read simply by using Java substringing. Or if the record is more complex, the byte array can be read by using the Java record framework tooling. The Java record framework tooling is available in IBM Rational Application Developer or IBM Rational Developer for System z.

Example A-1 ReadKsdsVsam.java

```
import java.io.UnsupportedEncodingException;
import com.ibm.jzos.ZFile;
import com.ibm.jzos.ZFileException;
public class ReadKsdsVsam
{
   private ZFile zFile;
   private String filename;
   private String options;
   private int lrecl;
   private int keyLen;
   private byte[] keyBytes;
public ReadKsdsVsam(String filenameInput, int lreclInput, int keyLenInput) throws
ReadKsdsVsamException
{
   // Set the options to open the file as VSAM type file, read only
   options = "rb,type=record";
   this.filename = filenameInput;
   this.lrecl = lreclInput;
   this.keyLen = keyLenInput;
   // Format the file name
   if(!filename.startsWith("//"))
   filename = "//" + filename;
   try { // Check the file exists
      if(!ZFile.exists(filename)) {
         throw new ReadKsdsVsamException("File "+filename+" does not exist");
      }
         // Open the file
         zFile = new ZFile(filename, options);
      }
      catch (ZFileException zfe)
      {
         throw new ReadKsdsVsamException(zfe);
      }
   }
}
public byte[] readRecord(String key) throws ReadKsdsVsamException
{
   byte[] record = new byte[lrecl];
try {
      keyBytes = key.getBytes(ZFile.DEFAULT_EBCDIC_CODE_PAGE);
      boolean located = zFile.locate(keyBytes, 0, keyLen, ZFile.LOCATE_KEY_EQ);
      if (!located) throw new ReadKsdsVsamException("Record: "+key+" cannot be
found");
      zFile.read(record);
   }
   catch (ZFileException zfe) {
      throw new ReadKsdsVsamException(zfe);
   }
      catch (UnsupportedEncodingException uee) {
      throw new ReadKsdsVsamException(uee);
   }
   return record;
}
public void closeFile() throws ReadKsdsVsamException
{
   try {
      zFile.close();
   }
      catch (ZFileException zfe) {
      throw new ReadKsdsVsamException(zfe);
   }
}
}
```

In Example A-1 on page 324, the constructor `public ReadKsdsVsam(String filenameInput, int lreclInput, int keyLenInput) throws ReadKsdsVsamException` takes the following arguments:

filenameInput	The fully qualified name of the file to read
lreclInput	The length of the record
keyLenInput	The length of the record key

The constructor checks to see whether the supplied file exists and then opens the file for reading. If any of these operations fail, it throws a `ReadKsdsVsamException` with the reason for the failure. This method can be verbalized and called in the initial actions section of the ruleflow.

To read a record from the VSAM file, you then use the following method:

```
public byte[] readRecord(String key) throws ReadKsdsVsamException
```

This method takes in the key of the record to read as a String and returns the contents of the record (including the key) as a byte[] array in the code page in which it was read.

Finally, the close() method is called to close the file. This method can be verbalized and called in the final actions section of the ruleset ruleflow.

B

Configuring runtime values by using variables defined in HBRINST

Operational Decision Manager can be customized in the HBRINST member of the ++HBRHLQ++.SHBRPARM dataset. Amending the variables in the member, when the customizing job is run, results in the values that are in HBRINST being placed into the customized members that are generated by the execution of the job.

Use the tables in the following sections, which group values into related areas, to amend the HBRINST member variables to your system environment:

- “Rules z/OS” on page 328
- “CICS JVM server” on page 330
- “IMS” on page 330
- “DB2 database” on page 331
- “WebSphere Application Server” on page 332
- “WebSphere Optimized Local Adapters (WOLA) script parameters” on page 333
- “WebSphere Application installation script parameters” on page 333
- “Subsystem ID used by COBOL management” on page 334

For more details about the HBRINST member and other members, see the *z/OS configuration and runtime variables* topic in the IBM Operational Decision Manager Information Center:

`http://pic.dhe.ibm.com/infocenter/dmanager/v8r0m1/topic/com.ibm.wodm.family.config.zos/topics/con_ds_jcl_and_runtime_vars.html`

Rules z/OS

Table B-1 lists the variables to create a new instance of Decision Server for z/OS.

Table B-1 Variables for rules on z/OS

Variable	Example value	Description
++HBRSSIDLIST++	`HBR1,HBR2`	A zRule Execution Server for z/OS server group consisting of a list of 1 - 32 subsystem IDs separated by commas, for example: `HBR1,HBR2,HBR3`. The first ID in the list is the primary server, from which you start the shared console. Rule execution is routed to the first available server in the list. Other servers execute rulesets only if rule execution is transferred to them. To route rule execution to a particular server, specify its ID first.
++HBRHLQ++	`HBR.V801`	High-level qualifier (HLQ) for Decision Server for z/OS data sets. This value is the installation target library from the product installation.
++HBRINSTPATH++	`/usr/lpp/zDM/V8R0M1`	This value is the root installation directory for the Operational Decision Manager product in z/OS UNIX System Services.
++HBRWORKPATH++	`/u/HBR1`	Working directory for the server instance. The value of this variable must differ from the value of the ++*CICSWORKPATH*++ variable.
++HBRWORKDS++	`HBR.WORKDS`	HLQ for the working data sets that contain customized JCL for creating an execution environment instance. For each new instance of the zRule Execution Server, this can remain the same. The HBRUUPTI job that appends the SSID to this value must be updated to a value that references the new instance. This value is the data set name for the changed output from the HBRUUPTI job.
++HBRJAVAHOME++	`/java/java/601_bit64_GA/J6.0.1_64`	This value is the *root* location of Java 6.0.1 on z/OS in UNIX System Services.
++HBRSSPPORT++	`24114`	Port number on which the Scenario Service Provider (SSP) service exists and is the port value for a zRule Execution Server that is set up in *TESTING* mode to use. This is used in Decision Validation Service (DVS) testing.
++HBRCONSOLEPORT++	`34114`	This value is the port that is used for the zRule Execution Server for the z/OS Execution Server Console. This value is the port that you use to deploy and view deployed artifacts.
++HBRCONSOLECOMPORT++	`44114`	This value is the port that is used by the zRes Console and zRule Execution Server for z/OS instance to communicate with each other.

Variable	Example value	Description
++HBRCONSOLECOMHOST++	`localhost`	Leaving this as `localhost` uses the default machine address. For a cross-LPAR setup (see a later section), this can be adjusted.
++HBRMODE++	`NORMAL`, `NORULE`, or `TEST`	The zRule Execution Server has three possible configurations: ► In `NORMAL` mode, the server accepts a COBOL workload and processes it using its internal Java virtual machine (JVM). The server settings can be changed using the console. This mode is used when executing and managing rules from a mainframe. ► In `NORULE` mode, the server uses an external JVM to process rules. This mode is used when executing rules on a JVM Server under CICS. ► In `TEST` mode, the server is used for testing and does not accept connections from local clients. This mode is used to start SSP and execute rules only from the console.
++HBRLANG++	`En_US`	Language used by the server. The list of supported languages is in the HBRCMMN data set member. The default value is `En_US`.
++HBRTRACELEVEL++	`ALL`, `FINE`, `INFO`, `WARNING`, `SEVERE`, or `OFF`	Trace level during execution: ► `ALL`: Logs all messages, including internal traces ► `FINE`: Logs debug messages, informational messages, errors, and warnings ► `INFO`: Logs informational messages, errors, and warnings ► `WARNING`: Logs errors and warnings ► `SEVERE`: Logs errors only ► `OFF`: No tracing
++HBRPERSISTENCETYPE++	`DB2` or `FILE`	Type of persistence layer that is used to store deployed artifacts. Set this variable to `DB2` or `FILE`.

CICS

If you are configuring CICS to execute rules on an instance of zRule Execution Server for z/OS, customize the variables that are listed in Table B-2.

Table B-2 Variables for CICS

Variable	Example value	Description
++CICSHLQ++	`CTS420.CICS`	This value is the HLQ for the CICS installation. Change this value to match the CICS installation HLQ.
++CICSCSDDSN++	`CTS420.APPLID.DFHCSD`	This value is the HLQ for the CICS region CICS system definition data set (CSD) file. For each new region into which a zRule Execution Server for z/OS is to be deployed, this value must be updated.
++CICSLIST++	`DFHLIST`	CICS startup group list that is specified for the **`GRPLIST`** parameter.

CICS JVM server

If you are configuring an instance of zRule Execution Server for z/OS running on a CICS JVM server, customize the variables that are listed in Table B-3. Note that some of these variables are repeated from Table B-2 on page 329.

Table B-3 Variables for CICS JVM server

Variable	Example value	Description
++CICSWORKPATH++	`/u/HBR1/CICS`	This value is the CICS UNIX System Services working directory path. Set up each server so that each new zRule Execution Server for z/OS instance has its own CICS work path. This way, problem determination is easier with separate logs for each zRule Execution Server for z/OS instance. The value of this variable must differ from the value of the ++*HBRWORKPATH*++ variable.
++CICSHLQ++	`CTS420.CICS`	This value is the HLQ for the CICS installation. Change this value to match the CICS installation HLQ.
++CICSCSDDSN++	`CTS420.APPLID.DFHCSD`	This value is the HLQ for the CICS region CICS system definition data set (CSD) file. For each new region into which a zRule Execution Server for z/OS is to be deployed, this value must be updated.
++*CICSINSTPATH*++	`/cics/cics670/lib`	Location of the CICS TS 4.1 JVM server JAR files.
++CICSLIST++	`DFHLIST`	CICS startup GRPLIST. List of groups containing the resource definitions that are created when you run the HBRCSD job.
++HBRJAVA31HOME++	`/java/java601_bit31_GA/J6.0.1`	If the system is installed in a CICS V4.1 region, the 31-bit JVM must be set here. If the CICS region is CICS V4.2 or higher, this is not used.
++JDBCPLAN++	`DSNJCC`	This value is the planned use for Java Database Connectivity (JDBC) connections and CICS.

IMS

If you are configuring IMS to execute rules on an instance of zRule Execution Server for z/OS, customize the variables that are listed in Table B-4.

Table B-4 Variables for IMS

Variable	Example value	Description
++IMSHLQ++	`IMS.V10.DBDC`	HLQ for the data sets of the IMS installation.
++IMSREGID++	`IMOA`	ID of the IMS instance to be used.
++IMSREGHLQ++	`IMSDATA.IMOA`	HLQ of the IMS region data sets.

DB2 database

If you are using a DB2 database as the persistence layer, customize the variables that are listed in Table B-5.

Table B-5 Variables for DB2

Variable	Example value	Description
++DB2HLQ++	`SYS2.DB2.V9R1`	HLQ of the DB2 installation.
++DB2RUNLIB++	`DSN910GP.RUNLIB.LOAD`	DB2 runtime library.
++DB2SUBSYSTEM++	`db2_subsystem_id`	DB2 subsystem name.
++DB2LOCATION++	`DSN910GP`	DB2 location name that is used to connect to this DB2 subsystem.
++DB2VCAT++	`DSN910GP`	DB2 integrated catalog facility (ICF) catalog.
++DB2ADMIN++	`DB2ADMINID`	User ID that is authorized to create Events DB2 databases.
++DB2CURRSQLID++	`ODMDBUSR`	Current SQL ID. The owner of a table space, database, or storage group. An authorization ID with the same name as a schema implicitly has CREATIN, ALTERIN, and DROPIN privileges for that schema.
++RESDATABASE++	`RESDB1`	Name of the database that is used by the zRule Execution Server for z/OS instance.
++RTSDATABASE++	`RTSDB1`	Name of the database that is used by the Decision Center instance.
++EVDATABASE++	`EVDB1`	Name of the database that is used by the Events runtime.
++RESSTOGROUP++	`RESSTG1`	Name of the storage group that is used by the zRule Execution Server for z/OS instance.
++RTSSTOGROUP++	`RTSSTG1`	Name of the storage group that is used by the Decision Center instance.
++EVSTOGROUP++	`EVSSTG1`	Name of the storage group that is used by the Events runtime instance
++DB2TABLEBP++	`BP1`	Buffer pool name for the tables.
++DB2INDEXBP++	`BP2`	Buffer pool name for the indexes.
++DB2LOBBP++	`BP3`	Buffer pool name for large objects.
++DB2SAMPLEPROGRAM++	`DSNTEP2`	DB2 program name.
++DB2SAMPLEPROGRAMPLAN++	`DSNTEP91`	DB2 plan name.
++DB2BP4K++	`BP4K`	Buffer pool name for 4 K objects.
++DB2BP8K++	`BP8K`	Buffer pool name for 8 K objects.
++DB2BP32K++	`BP32K`	Buffer pool name for 32 K objects.
++DB2USER++	`ODMDBUSR`	User ID for accessing the DB2 database.
++DB2PSWD++	*`<your password>`*	Password for accessing the DB2 database.

Variable	Example value	Description
++DB2CONSTR++	`host.db2.ipaddr.com:49100/DSN910GP`	Connection string for a Java Database Connectivity (JDBC) universal driver type 4 connection, with the format: *ipaddress:portnumber/database_subsystem_ID*
++DB2JARLOCN++	`/usr/lpp/db2910/classes`	Location of the DB2 classes in UNIX System Services.
++DB2NATIVELOC++	/usr/lpp/db2910/lib	Location of the DB2 native library files.

WebSphere Application Server

If you are configuring Operational Decision Manager on WebSphere Application Server for z/OS, customize the variables that are listed in Table B-6.

Table B-6 Variables for WebSphere Application Server

Variable	Example value	Description
++WASINSTPATH++	`/WebSphere/V8OIL2Z1/Appserver`	Installation directory of WebSphere Application Server.
++WAS_HOME++	`/WebSphere/AppServer/Profile`	WebSphere Application Server home directory. It is unique for each server instance.
++SECURITYTYPE++	`RACF`	Set to `RACF` if your WebSphere Application Server system is configured to use RACF. Set to `WebSphere Application Server` if your WebSphere Application Server system is configured to use federated security.
++DMGRPATH++	`/WebSphere/V8ILGDM`	The DManager path in an WebSphere Application Server Network Deployment environment. Important: After you run HBRUUPTI, check the following data set members to ensure that the DManager path length did not exceed the permitted length and get truncated: ► `HBRDSWAS` ► `HBRDCWAS` ► `HBRDSDVS`
++WASSERVERNAME++	`Serveril2Base`	WebSphere Application Server instance name.
++PROFILE++	`default`	WebSphere Application Server profile. Set to `default` on z/OS.
++CELLNAME++	`cell01`	WebSphere Application Server cell name.
++NODENAME++	`node1`	WebSphere Application Server node name.
++ADMINHOST++	*`<WAS server IP name>`*	Name of the host on which the WebSphere Application Server is running.
++WASBOOTSTRAPPORT++	`1234`	Boot strap port that is used by WebSphere Application Server.
++ADMINUSER++	`wasadmin`	Administration console user ID for the WebSphere Application Server administration console.

Variable	Example value	Description
++ADMINPSWD++	`<your password>`	Administration console user password for the WebSphere Application Server administration console for the above ID.
++EJBHLQ++	`CLID`	System Authorization Facility (SAF) prefix for EJBROLEs. This might be blank.

WebSphere Optimized Local Adapters (WOLA) script parameters

If you want your COBOL applications to connect to a Rule Execution Server on WebSphere Application Server through WebSphere Optimized Local Adapters (WOLA), customize the variables that are listed in Table B-7. For specific details about WOLA configuration, see Chapter 13, “Configuring WebSphere Optimized Local Adapters support” on page 225.

Table B-7 Variables for WOLA script parameters

Variable	Example value	Description
++HBRWOLALOADLIB++	`USER.WOLA.LOADLIB.WAS8`	Load library that is selected as part of setting up WOLA.
++HBRTARGETRES++	`WOLA`	Location for rules execution, in this case, `WOLA`.
++HBRWOLACELL++	`CILK`	Short name of the WebSphere Application Server cell to which to connect using WOLA.
++HBRWOLANODE++	`NILK`	Short name of the WebSphere Application Server node to which to connect using WOLA.
++HBRWOLASERVER++	`WSVR01`	Short name of the WebSphere Application Server for the connection.

WebSphere Application installation script parameters

If you are configuring Operational Decision Manager for z/OS on WebSphere Application Server using **`wsadmin`** scripts, customize the variables that are listed in Table B-8.

Table B-8 Variables for WebSphere Application installation script parameters

Variable	Example value	Description
++RESADMIN++	`resAdministrators`	Administrator user group for the Rule Execution Server.
++RESDEPLOY++	`resDeployers`	Deployment user group for the Rule Execution Server.
++RESMONITOR++	`resMonitors`	Monitor user group for the Rule Execution Server.
++RESADMINUSER++	`resAdmin`	Administration user for the Rule Execution Server.
++RESDEPLOYUSER++	`resDeployer`	Deployment user for the Rule Execution Server.
++RESMONITORUSER++	`resMonitor`	Monitor user for the Rule Execution Server.
++RTSADMIN++	`rtsAdministrator`	Administrator user group for Decision Center.
++RTSCONFIG++	`rtsConfigManager`	Configuration user group for Decision Center.

Variable	Example value	Description
++RTSUSER++	`rtsUser`	User group for Decision Center.
++RTSINSTALLER++	`rtsInstaller`	Installer user group for Decision Center.
++RTSADMINUSER++	`rtsAdmin`	Administration user for Decision Center.
++RTSCONFIGUSER++	`rtsConfig`	Configuration user for the Decision Center.
++RTSUSERUSER++	`rtsUser1`	User for Decision Center.
++RTSDBDRIVERTYPE++	`4`	JDBC universal driver type for the RTS data source connection.
++XOMDBDRIVERTYPE++	`4`	JDBC universal driver type for the execution object module (XOM) data source connection.
++RESDBDRIVERTYPE++	`4`	JDBC universal driver type for the RES data source connection.
++DB2SERVNAME++	*<host name>*	DB2 host name or IP address.
++DB2PORT++	49100	DB2 connection port.

Subsystem ID used by COBOL management

If you are configuring an execution environment to run COBOL rule subprograms, customize the variables that are listed in Table B-9.

Table B-9 Subsystem ID used by COBOL management

Variable	Example value	Description
++R4CSSID++	`SSID`	Variable to create a new subsystem ID for a COBOL rule subprogram.

C

Additional material

This book refers to additional material that can be downloaded from the Internet as described in the following sections.

Locating the web material

The web material that is associated with this book is available in softcopy on the Internet from the IBM Redbooks web server:

`ftp://www.redbooks.ibm.com/redbooks/SG248014`

Alternatively, you can go to the IBM Redbooks website:

`http://www.ibm.com/redbooks`

Select **Additional materials** and open the directory that corresponds with the IBM Redbooks form number, SG248014.

Downloading and extracting the web material

Create a subdirectory (folder) on your workstation, and extract the contents of the web material `.zip` file into this folder.

© Copyright IBM Corp. 2013. All rights reserved.

Abbreviations and acronyms

BAL	Business Action Language
BEP	Business Event Processing
BOM	Business object model
BPM	Business Process Management
DVS	Decision Validation Services
HLQ	High-level qualifier
IBM	International Business Machines Corporation
ITSO	International Technical Support Organization
IVP	Installation verification procedure
JMS	Java Message Service
JVM	Java virtual machine
KPI	Key performance indicator
RES	Rule Execution Server
SSP	Scenario service provider
XU	Execution unit
zFS	z/OS Distributed File Service
zRES	Rules Execution Server for z/OS

© Copyright IBM Corp. 2013. All rights reserved.

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 publications referenced in this list might be available in softcopy only.

- *Implementing Event Processing with CICS*, SG24-7792
- *Proven Practices for Enhancing Performance: A Q & A for IBM WebSphere ILOG BRMS 7.1*, REDP-4775
- *Making Better Decisions using WebSphere Operational Decision Management*, REDP-4836

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

ibm.com/redbooks

Other publications

These publications are also relevant as further information sources:

- Barbara von Halle, *Business Rules Applied*, Wiley, 2001, ISBN 978-0471412939

Online resources

These websites are also relevant as further information sources:

- IBM Operational Decision Manager product web site:

  http://www.ibm.com/software/decision-management/operational-decision-management/websphere-operational-decision-management/
- IBM WebSphere Operational Decision Management Version 8.0 Information Center:

  http://pic.dhe.ibm.com/infocenter/dmanager/v8r0/index.jsp
- Operational Decision Management Library:

  http://www.ibm.com/software/decision-management/operational-decision-management/odm-library/
- To enable history in the Decision Server Events run time:

  http://pic.dhe.ibm.com/infocenter/dmanager/v8r0/topic/com.ibm.wodm.dserver.events.config/topics/tsk_configuring_history_runtime.html

© Copyright IBM Corp. 2013. All rights reserved.

- WebSphere connectors:

  http://pic.dhe.ibm.com/infocenter/dmanager/v8r0/index.jsp?topic=%2Fcom.ibm.wodm.dserver.events.admin%2Ftopics%2Frunningconnectorsinthewasenvironment.html

- Stand-alone connectors:

  http://pic.dhe.ibm.com/infocenter/dmanager/v8r0/topic/com.ibm.wodm.dserver.events.admin/topics/zos_runningtechnologyconnectors.html

- The ilog.rules.dvs.client package extensive online documentation with code samples that explain how to use it:

  http://pic.dhe.ibm.com/infocenter/dmanager/v8r0/topic/com.ibm.wodm.dserver.rules.ref.designer/html/api/html/ilog/rules/dvs/client/package-summary.html

- CICS Explorer:

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

Help from IBM

IBM Support and downloads

ibm.com/support

IBM Global Services

ibm.com/services

IBM

Redbooks

Flexible Decision Automation for Your zEnterprise with Business Rules and Events

(0.5" spine)
0.475"<->0.873"
250 <-> 459 pages

Flexible Decision Automation for Your zEnterprise with Business Rules and Events

Understand the benefits of operational decision management

Build dynamic solutions with business events and business rules

Learn by example with practical scenarios

The IBM Operational Decision Manager product family provides value to organizations that want to improve the responsiveness and precision of automated decisions. This decision management platform on IBM z/OS provides comprehensive automation and governance of operational decisions that are made within mainframe applications. These decisions can be shared with other cross-platform applications, providing true enterprise decision management.

This IBM Redbooks publication makes the case for using Operational Decision Manager for z/OS and provides an overview of its components. It is aimed at IT architects, enterprise architects, and development managers looking to build rule-based and business event-based solutions. Step-by-step guidance is provided on getting started with business rules and creating business events by using a scenario-based approach. This book provides detailed guidelines for testing and simulation and describes advanced options for decision authoring. Finally, it describes and documents multiple runtime configuration options.

This second edition, SG24-8014-01, of this IBM Redbooks publication updated the information presented in this book to reflect function available in IBM Operational Decision Manager for z/OS Version 8.0.1. It is also important to note that the product name has changed from IBM WebSphere Operational Decision Management for z/OS to IBM Operational Decision Manager for z/OS.

INTERNATIONAL TECHNICAL SUPPORT ORGANIZATION

BUILDING TECHNICAL INFORMATION BASED ON PRACTICAL EXPERIENCE

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

For more information:
ibm.com/redbooks

SG24-8014-01 ISBN 0738438855