

Course Guide

# Developing Rule Solutions in IBM Operational Decision Manager V8.10

Course code WB404 / ZB404 ERC 1.2



**June 2019 edition**

## **Notices**

This information was developed for products and services offered in the US.

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, MD-NC119  
Armonk, NY 10504-1785  
United States of America*

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 jurisdictions 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 provide in any way it believes appropriate without incurring any obligation to you.

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 actual people or business enterprises is entirely coincidental.

## **Trademarks**

IBM, the IBM logo, and ibm.com 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" at [www.ibm.com/legal/copytrade.shtml](http://www.ibm.com/legal/copytrade.shtml).

**© Copyright International Business Machines Corporation 2019.**

**This document may not be reproduced in whole or in part without the prior written permission of IBM.**

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

# Contents

|   |             |
|---|-------------|
| <b>Trademarks .....</b>   | <b>xvii</b> |
| <b>Course description .....</b>                                   | <b>xix</b>  |
| <b>Agenda .....</b>   | <b>xxi</b>  |
| <b>Unit 1. Introducing IBM Operational Decision Manager .....</b> | <b>1-1</b>  |
| Unit objectives .....   | 1-2         |
| How to check online for course material updates .....             | 1-3         |
| Topics .....  | 1-4         |
| 1.1. What is a business rule? .....                               | 1-5         |
| What is a business rule? .....                                    | 1-6         |
| What is a business rule? .....                                    | 1-7         |
| From business policy to business rules .....                      | 1-8         |
| Discussion .....  | 1-9         |
| Discussion: What is a rule? .....                                 | 1-10        |
| 1.2. Why the need for decision management? .....                  | 1-11        |
| Why the need for decision management? .....                       | 1-12        |
| Why is there a need for decision management? .....                | 1-13        |
| What is decision management? .....                                | 1-15        |
| 1.3. What is operational decision management? .....               | 1-17        |
| What is operational decision management? .....                    | 1-18        |
| Challenges to decision automation challenges .....                | 1-19        |
| Using operational decision management .....                       | 1-20        |
| What is operational decision management? .....                    | 1-22        |
| Synchronized business and IT cycles .....                         | 1-23        |
| 1.4. Introducing IBM Operational Decision Manager .....           | 1-24        |
| Introducing IBM Operational Decision Manager .....                | 1-25        |
| Operational Decision Manager (1 of 2) .....                       | 1-26        |
| Operational Decision Manager (2 of 2) .....                       | 1-27        |
| Operational Decision Manager roles and activities .....           | 1-29        |
| Decision Center: Authoring and maintaining business rules .....   | 1-30        |
| Decision Server: Development and execution .....                  | 1-31        |
| Decision Server Insights .....                                    | 1-32        |
| 1.5. ODM packaging .....  | 1-34        |
| ODM packaging .....   | 1-35        |
| Options for using ODM .....                                       | 1-36        |
| On-premises editions .....  | 1-37        |
| IBM ODM on Cloud: Three runtime environments .....                | 1-38        |
| ODM on IBM Cloud Private .....                                    | 1-39        |
| ODM for Developers Docker .....                                   | 1-40        |
| 1.6. ODM roles .....  | 1-41        |
| ODM roles .....   | 1-42        |
| Operational Decision Manager user roles .....                     | 1-43        |
| Interaction between roles .....                                   | 1-44        |
| Business analyst .....  | 1-45        |
| Policy manager .....  | 1-46        |
| Rule author .....   | 1-47        |
| Architect .....   | 1-48        |
| Developer .....   | 1-49        |

|  |             |
|--|-------------|
| Administrator .....  | 1-50        |
| Discussion .....   | 1-51        |
| <b>1.7. Governance and decision management .....</b>           | <b>1-52</b> |
| Governance and decision management .....                       | 1-53        |
| Applying governance to decision management .....               | 1-54        |
| Governance in Operational Decision Manager .....               | 1-55        |
| Unit summary .....   | 1-56        |
| Review questions (1 of 2) .....                                | 1-57        |
| Review questions (2 of 2) .....                                | 1-58        |
| Review answers (1 of 2) .....                                  | 1-59        |
| Review answers (2 of 2) .....                                  | 1-60        |
| Exercise: Operational Decision Manager in action .....         | 1-61        |
| Exercise introduction .....                                    | 1-62        |
| Exercise workflow .....  | 1-63        |
| <b>Unit 2. Developing decision services .....</b>              | <b>2-1</b>  |
| Unit objectives .....  | 2-2         |
| Topics .....   | 2-3         |
| <b>2.1. Using an agile development approach .....</b>          | <b>2-4</b>  |
| Using an agile development approach .....                      | 2-5         |
| Lifecycle of business decisions .....                          | 2-6         |
| Agile Business Rule Development methodology .....              | 2-7         |
| Agile development approach .....                               | 2-8         |
| Discovery: Identifying decisions in the process (1 of 2) ..... | 2-9         |
| Discovery: Identifying decisions in the process (2 of 2) ..... | 2-11        |
| Discovery: Identifying the rules .....                         | 2-12        |
| Discovery: Identifying vocabulary .....                        | 2-13        |
| Implementing the model .....                                   | 2-14        |
| Using an agile development approach .....                      | 2-15        |
| <b>2.2. Designing the business rule application .....</b>      | <b>2-17</b> |
| Designing the business rule application .....                  | 2-18        |
| Overview of developer tasks .....                              | 2-19        |
| Rule projects (1 of 2) .....                                   | 2-21        |
| Rule projects (2 of 2) .....                                   | 2-22        |
| <b>2.3. Setting up decision services .....</b>                 | <b>2-24</b> |
| Setting up decision services .....                             | 2-25        |
| Decision service map .....                                     | 2-26        |
| Create a decision service rule project in Rule Designer .....  | 2-27        |
| Decision service rule project templates .....                  | 2-28        |
| Design: XOM and BOM .....                                      | 2-29        |
| Design: XOM .....  | 2-30        |
| Design: BOM .....  | 2-31        |
| Defining the decision operation .....                          | 2-32        |
| Using the Operation Map .....                                  | 2-33        |
| Designing the signature .....                                  | 2-34        |
| Designing the signature .....                                  | 2-35        |
| <b>2.4. Project properties .....</b>                           | <b>2-36</b> |
| Project properties .....                                       | 2-37        |
| Properties: Project hierarchy .....                            | 2-38        |
| Properties: Folders .....                                      | 2-39        |
| Properties: Project references .....                           | 2-40        |
| Properties: Rule engine type .....                             | 2-41        |
| <b>2.5. Using modular project organization .....</b>           | <b>2-42</b> |
| Using modular project organization .....                       | 2-43        |
| Modular structure (1 of 4) .....                               | 2-44        |
| Modular structure (2 of 4) .....                               | 2-45        |

|   |            |
|---|------------|
| Modular structure (3 of 4) .....                            | 2-46       |
| Modular structure (4 of 4) .....                            | 2-47       |
| 2.6. Sharing and synchronizing decision services.....       | 2-48       |
| Sharing and synchronizing decision services .....           | 2-49       |
| Synchronization between users .....                         | 2-50       |
| Synchronization tools .....                                 | 2-51       |
| Synchronization architecture .....                          | 2-52       |
| Connection entries .....                                    | 2-53       |
| Choosing how to synchronize .....                           | 2-54       |
| Synchronization commands .....                              | 2-55       |
| Choosing the master source .....                            | 2-56       |
| Unit summary .....  | 2-57       |
| Review questions .....                                      | 2-58       |
| Review answers .....  | 2-59       |
| Exercise: Setting up decision services .....                | 2-60       |
| Exercise introduction .....                                 | 2-61       |
| <b>Unit 3. Modeling decisions .....</b>                     | <b>3-1</b> |
| Unit objectives .....                                       | 3-2        |
| Topics .....  | 3-3        |
| 3.1. Introducing decision modeling in Decision Center ..... | 3-4        |
| Introducing decision modeling in Decision Center .....      | 3-5        |
| Recall: Using an agile development approach .....           | 3-6        |
| Modeling decisions in Business console .....                | 3-7        |
| Choosing the top-down decision model approach .....         | 3-8        |
| Decision model services versus decision services .....      | 3-9        |
| 3.2. Creating decision models .....                         | 3-10       |
| Creating decision models .....                              | 3-11       |
| Modeling in Business console .....                          | 3-12       |
| Creating the decision model diagram (1 of 2) .....          | 3-13       |
| Creating the decision model diagram (2 of 2) .....          | 3-14       |
| Modeling the node structure (1 of 2) .....                  | 3-15       |
| Modeling the node structure (2 of 2) .....                  | 3-16       |
| Authoring decision logic (1 of 2) .....                     | 3-17       |
| Authoring decision logic (2 of 2) .....                     | 3-18       |
| Defining custom types .....                                 | 3-19       |
| 3.3. Modeling with IBM Decision Composer .....              | 3-20       |
| Modeling with IBM Decision Composer .....                   | 3-21       |
| IBM Decision Composer on IBM Cloud .....                    | 3-22       |
| IBM Decision Composer online .....                          | 3-23       |
| Decision Composer online .....                              | 3-24       |
| For more information .....                                  | 3-25       |
| Unit summary .....  | 3-26       |
| Review questions .....                                      | 3-27       |
| Review answers (1 of 2) .....                               | 3-28       |
| Exercise: Modeling decisions .....                          | 3-29       |
| Exercise introduction .....                                 | 3-30       |
| Exercise overview .....                                     | 3-31       |
| <b>Unit 4. Programming with business rules .....</b>        | <b>4-1</b> |
| Unit objectives .....                                       | 4-2        |
| Topics .....  | 4-3        |
| 4.1. Introducing ruleset integration and execution.....     | 4-4        |
| Introducing ruleset integration and execution .....         | 4-5        |
| Integrating Operational Decision Manager .....              | 4-6        |
| Integration steps .....                                     | 4-7        |

|   |            |
|---|------------|
| Execution environments . . . . .                                | 4-8        |
| 4.2. What is a decision engine? . . . . .                       | 4-9        |
| What is a decision engine? . . . . .                            | 4-10       |
| Engine . . . . .  | 4-11       |
| Rule execution by the rule engine . . . . .                     | 4-12       |
| Decision engine API . . . . .                                   | 4-13       |
| 4.3. Understanding rule execution modes . . . . .               | 4-14       |
| Understanding rule execution modes . . . . .                    | 4-15       |
| Execution modes . . . . .                                       | 4-16       |
| Execution modes: Fastpath (1 of 2) . . . . .                    | 4-17       |
| Execution modes: Fastpath (2 of 2) . . . . .                    | 4-18       |
| Execution modes: Sequential (1 of 2) . . . . .                  | 4-19       |
| Execution modes: Sequential (2 of 2) . . . . .                  | 4-20       |
| Execution modes: RetePlus (1 of 3) . . . . .                    | 4-21       |
| Execution modes: RetePlus (2 of 3) . . . . .                    | 4-23       |
| Execution modes: RetePlus (3 of 3) . . . . .                    | 4-24       |
| RetePlus: Rule order and selection . . . . .                    | 4-25       |
| Refraction (1 of 2) . . . . .                                   | 4-26       |
| Refraction (2 of 2) . . . . .                                   | 4-27       |
| Recency (1 of 2) . . . . .                                      | 4-28       |
| Recency (2 of 2) . . . . .                                      | 4-29       |
| Choosing an execution mode . . . . .                            | 4-30       |
| Choosing execution mode according to application type . . . . . | 4-31       |
| 4.4. RetePlus example: Trucks and drivers . . . . .             | 4-32       |
| RetePlus example: Trucks and drivers . . . . .                  | 4-33       |
| RetePlus in action . . . . .                                    | 4-34       |
| Input objects in working memory . . . . .                       | 4-35       |
| Evaluating rules with objects . . . . .                         | 4-36       |
| Rule instances in the agenda for execution (1 of 2) . . . . .   | 4-37       |
| Rule instances in the agenda for execution (2 of 2) . . . . .   | 4-38       |
| ChangeTire executes . . . . .                                   | 4-39       |
| Side effects: Updated objects create matches . . . . .          | 4-40       |
| AssignDriver executes . . . . .                                 | 4-41       |
| Side effects: Updated objects no longer match . . . . .         | 4-42       |
| AssignDriver executes again . . . . .                           | 4-43       |
| Side effects of AssignDriver . . . . .                          | 4-44       |
| Rule execution completes when agenda is empty . . . . .         | 4-45       |
| Unit summary . . . . .  | 4-46       |
| Review questions . . . . .                                      | 4-47       |
| Review answers (1 of 2) . . . . .                               | 4-48       |
| <b>Unit 5. Developing object models . . . . .</b>               | <b>5-1</b> |
| Unit objectives . . . . .                                       | 5-2        |
| Topics . . . . .  | 5-3        |
| 5.1. Designing the models . . . . .                             | 5-4        |
| Designing the models . . . . .                                  | 5-5        |
| Introduction to the models . . . . .                            | 5-6        |
| BOM and XOM at run time . . . . .                               | 5-7        |
| Designing the BOM and the XOM . . . . .                         | 5-8        |
| Example: Car insurance class diagram . . . . .                  | 5-9        |
| Example: Implementation of insurance model . . . . .            | 5-10       |
| Example: Implementation of insurance model . . . . .            | 5-11       |
| 5.2. Defining business and execution object models . . . . .    | 5-12       |
| Defining business and execution object models . . . . .         | 5-13       |
| Business object model (BOM) . . . . .                           | 5-14       |
| BOM entries . . . . .   | 5-15       |

|  |            |
|--|------------|
| BOM path . . . . .   | 5-16       |
| Execution object model (XOM) . . . . .                                     | 5-17       |
| Types of XOM . . . . .   | 5-18       |
| Rule project and XOM . . . . .   | 5-19       |
| 5.3. Editing the business object model . . . . .                           | 5-20       |
| Editing the business object model . . . . .                                | 5-21       |
| Introduction . . . . .   | 5-22       |
| BOM editor (1 of 2) . . . . .  | 5-23       |
| BOM editor (2 of 2) . . . . .  | 5-24       |
| General information for methods . . . . .                                  | 5-25       |
| General information for attributes . . . . .                               | 5-26       |
| 5.4. Mapping the BOM to the XOM . . . . .                                  | 5-28       |
| Mapping the BOM to the XOM . . . . .                                       | 5-29       |
| What is BOM-to-XOM mapping? . . . . .                                      | 5-30       |
| Rule language mapping . . . . .  | 5-31       |
| Use of explicit mappings . . . . .   | 5-32       |
| Create a method using ARL mapping code . . . . .                           | 5-33       |
| Testing instances of an execution class (1 of 2) . . . . .                 | 5-34       |
| Testing instances of an execution class (2 of 2) . . . . .                 | 5-35       |
| Example: Mapping a business class to an execution class (1 of 2) . . . . . | 5-36       |
| Example: Mapping a business class to an execution class (2 of 2) . . . . . | 5-37       |
| Explicit extender mapping . . . . .  | 5-38       |
| 5.5. Working with the vocabulary . . . . .                                 | 5-39       |
| Working with the vocabulary . . . . .                                      | 5-40       |
| Rule vocabulary . . . . .  | 5-41       |
| Vocabulary and verbalization . . . . .                                     | 5-42       |
| Verbalization in the BOM editor . . . . .                                  | 5-43       |
| Verbalization (1 of 2) . . . . .   | 5-44       |
| Verbalization (2 of 2) . . . . .   | 5-45       |
| Placeholders . . . . .   | 5-46       |
| Verbalizing BOM members . . . . .  | 5-47       |
| 5.6. Refactoring . . . . .   | 5-49       |
| Refactoring . . . . .  | 5-50       |
| Refactoring . . . . .  | 5-51       |
| From the XOM to the rules . . . . .  | 5-52       |
| BOM and XOM evolution . . . . .  | 5-53       |
| XOM changes (1 of 3) . . . . .   | 5-54       |
| XOM changes (2 of 3) . . . . .   | 5-55       |
| XOM changes (3 of 3) . . . . .   | 5-56       |
| BOM changes . . . . .  | 5-57       |
| Vocabulary changes . . . . .   | 5-58       |
| Unit summary . . . . .   | 5-59       |
| Review questions . . . . .   | 5-60       |
| Review answers (1 of 2) . . . . .  | 5-61       |
| Review answers (2 of 2) . . . . .  | 5-62       |
| Exercise: Working with the BOM . . . . .                                   | 5-63       |
| Exercise introduction . . . . .  | 5-64       |
| Exercise: Refactoring . . . . .  | 5-65       |
| Exercise introduction . . . . .  | 5-66       |
| <b>Unit 6. Orchestrating ruleset execution . . . . .</b>                   | <b>6-1</b> |
| Unit objectives . . . . .  | 6-2        |
| Topics . . . . .   | 6-3        |
| 6.1. Controlling rule execution: Overview . . . . .                        | 6-4        |
| Controlling rule execution: Overview . . . . .                             | 6-5        |
| What is orchestration? . . . . .   | 6-6        |

|  |            |
|--|------------|
| Key elements for orchestration . . . . .                         | 6-7        |
| 6.2. Designing ruleflows. . . . .                                | 6-8        |
| Designing ruleflows . . . . .                                    | 6-9        |
| Ruleflows . . . . .  | 6-10       |
| Ruleflow elements . . . . .                                      | 6-11       |
| Rule tasks . . . . .   | 6-12       |
| Initial and final actions . . . . .                              | 6-13       |
| Transitions . . . . .  | 6-14       |
| Forks and joins . . . . .  | 6-15       |
| Expression of initial or final actions, and conditions . . . . . | 6-16       |
| Main ruleflow . . . . .  | 6-17       |
| 6.3. Controlling rule selection for execution. . . . .           | 6-18       |
| Controlling rule selection for execution . . . . .               | 6-19       |
| Rule selection pipe . . . . .                                    | 6-20       |
| Ruleflow scope selection . . . . .                               | 6-21       |
| Runtime rule selection . . . . .                                 | 6-22       |
| Rule hierarchies . . . . .                                       | 6-23       |
| Rule overriding . . . . .  | 6-24       |
| Rule condition evaluation . . . . .                              | 6-25       |
| 6.4. Controlling rule order during rule execution . . . . .      | 6-26       |
| Controlling rule order during rule execution . . . . .           | 6-27       |
| Introduction . . . . .   | 6-28       |
| Rule priority . . . . .  | 6-29       |
| Rule task execution order (1 of 2) . . . . .                     | 6-30       |
| Rule task execution order (2 of 2) . . . . .                     | 6-31       |
| Execution modes . . . . .  | 6-32       |
| Unit summary . . . . .   | 6-33       |
| Review questions . . . . .                                       | 6-34       |
| Review answers . . . . .   | 6-35       |
| Exercise: Working with ruleflows . . . . .                       | 6-36       |
| Exercise introduction . . . . .                                  | 6-37       |
| <b>Unit 7. Authoring rules. . . . .</b>                          | <b>7-1</b> |
| Unit objectives . . . . .  | 7-2        |
| Topics . . . . .   | 7-3        |
| 7.1. From business policy to business rules . . . . .            | 7-4        |
| From business policy to business rules . . . . .                 | 7-5        |
| Rule language evolves during a project (1 of 2) . . . . .        | 7-6        |
| Rule language evolves during a project (1 of 2) . . . . .        | 7-7        |
| When rule authoring occurs . . . . .                             | 7-8        |
| Who writes rules? . . . . .                                      | 7-9        |
| Where are rules written? . . . . .                               | 7-10       |
| 7.2. Authoring action rules with BAL . . . . .                   | 7-11       |
| Authoring action rules with BAL . . . . .                        | 7-12       |
| Business Action Language (BAL) . . . . .                         | 7-13       |
| Rule structure . . . . .   | 7-14       |
| Example of action rule . . . . .                                 | 7-15       |
| Rule definitions: Overview . . . . .                             | 7-16       |
| Rule definitions: Values . . . . .                               | 7-17       |
| Rule definitions: Constraints . . . . .                          | 7-18       |
| Multiple variables in BAL . . . . .                              | 7-19       |
| Rule conditions: Introduction . . . . .                          | 7-20       |
| Rule conditions: Comparison test . . . . .                       | 7-21       |
| Rule conditions: Membership test . . . . .                       | 7-22       |
| Rule conditions: Existence test . . . . .                        | 7-23       |
| Rule conditions: Count test . . . . .                            | 7-24       |

|  |            |
|--|------------|
| Multiple conditions . . . . .  | 7-25       |
| Avoiding ambiguity with parentheses (1 of 2) . . . . .                           | 7-26       |
| Avoiding ambiguity with parentheses (2 of 2) . . . . .                           | 7-27       |
| Use of commas (,) in rule conditions . . . . .                                   | 7-28       |
| Rule actions . . . . .   | 7-29       |
| Examples of rule actions . . . . .   | 7-31       |
| BAL number operators . . . . .   | 7-32       |
| BAL arithmetic operators . . . . .   | 7-33       |
| 7.3. Authoring decision tables . . . . .   | 7-34       |
| Authoring decision tables . . . . .  | 7-35       |
| Decision tables . . . . .  | 7-36       |
| Preconditions . . . . .  | 7-37       |
| Error checking in the editors . . . . .  | 7-38       |
| About decision tables . . . . .  | 7-39       |
| Example: Symmetrical rules . . . . .   | 7-40       |
| Example: Symmetrical rules in a decision table . . . . .                         | 7-41       |
| Structure of a decision table . . . . .  | 7-42       |
| Columns for conditions and actions . . . . .                                     | 7-43       |
| Locking facilities . . . . .   | 7-44       |
| Decision table size . . . . .  | 7-45       |
| 7.4. Advanced Rule Language (ARL) . . . . .                                      | 7-46       |
| Advanced Rule Language (ARL) . . . . .   | 7-47       |
| Translation from BAL to ARL . . . . .  | 7-48       |
| Viewing ARL . . . . .  | 7-49       |
| 7.5. Technical rules . . . . .   | 7-50       |
| Technical rules . . . . .  | 7-51       |
| Writing technical rules in IRL . . . . .   | 7-52       |
| IRL syntax . . . . .   | 7-53       |
| Technical rule structure . . . . .   | 7-55       |
| Multiple variables in IRL . . . . .  | 7-56       |
| Create a technical rule . . . . .  | 7-57       |
| 7.6. Defining objects that are used in rules . . . . .                           | 7-58       |
| Defining objects that are used in rules . . . . .                                | 7-59       |
| Introduction . . . . .   | 7-60       |
| Parameters . . . . .   | 7-61       |
| Ruleset variables . . . . .  | 7-62       |
| Objects in working memory . . . . .  | 7-63       |
| Working with objects in rule artifacts . . . . .                                 | 7-64       |
| Using IRL . . . . .  | 7-65       |
| Using verbalized elements in BAL . . . . .                                       | 7-66       |
| Rule variables (1 of 2) . . . . .  | 7-67       |
| Rule variables (2 of 2) . . . . .  | 7-68       |
| Using automatic variables . . . . .  | 7-69       |
| Unit summary . . . . .   | 7-70       |
| Review questions . . . . .   | 7-71       |
| Review answers . . . . .   | 7-72       |
| Exercise: Exploring action rules . . . . .                                       | 7-73       |
| Exercise introduction . . . . .  | 7-74       |
| Exercise: Authoring action rules . . . . .                                       | 7-75       |
| Exercise introduction . . . . .  | 7-76       |
| Exercise: Authoring decision tables . . . . .                                    | 7-77       |
| Exercise introduction . . . . .  | 7-78       |
| <b>Unit 8. Customizing rule vocabulary with categories and domains . . . . .</b> | <b>8-1</b> |
| Unit objectives . . . . .  | 8-2        |
| Topics . . . . .   | 8-3        |

|                |   |            |
|----------------|---|------------|
| 8.1.           | Simplifying vocabulary with categories . . . . .      | 8-4        |
|                | Simplifying vocabulary with categories . . . . .      | 8-5        |
|                | Categories in the BOM editor . . . . .                | 8-6        |
|                | Categories . . . . .                                  | 8-7        |
|                | Category semantics . . . . .                          | 8-8        |
| 8.2.           | Defining domains . . . . .                            | 8-9        |
|                | Defining domains . . . . .                            | 8-10       |
|                | Domains section . . . . .                             | 8-11       |
| 8.3.           | Static domains . . . . .                              | 8-12       |
|                | Static domains . . . . .                              | 8-13       |
|                | Domains: Literals . . . . .                           | 8-14       |
|                | Domains: Static references . . . . .                  | 8-15       |
|                | Domains: Bounded . . . . .                            | 8-16       |
|                | Domains: Collection . . . . .                         | 8-17       |
|                | Domains: Other . . . . .                              | 8-18       |
| 8.4.           | Dynamic domains . . . . .                             | 8-19       |
|                | Dynamic domains . . . . .                             | 8-20       |
|                | Domains: Dynamic (1 of 2) . . . . .                   | 8-21       |
|                | Domains: Dynamic (2 of 2) . . . . .                   | 8-22       |
|                | Dynamic domains: Microsoft Excel source (1 of 4)      | 8-23       |
|                | Dynamic domains: Microsoft Excel source (2 of 4)      | 8-24       |
|                | Dynamic domains: Microsoft Excel source (3 of 4)      | 8-25       |
|                | Dynamic domains: Microsoft Excel source (4 of 4)      | 8-26       |
|                | Enumerated versus complex domains . . . . .           | 8-27       |
|                | Automatic creation of domains . . . . .               | 8-28       |
| 8.5.           | Updating dynamic domains in Decision Center . . . . . | 8-29       |
|                | Updating dynamic domains in Decision Center . . . . . | 8-30       |
|                | Dynamic domains in Decision Center . . . . .          | 8-31       |
|                | Updating the dynamic domain (1 of 4) . . . . .        | 8-33       |
|                | Updating the dynamic domain (2 of 4) . . . . .        | 8-34       |
|                | Updating the dynamic domain (3 of 4) . . . . .        | 8-35       |
|                | Updating the dynamic domain (4 of 4) . . . . .        | 8-36       |
|                | Unit summary . . . . .                                | 8-37       |
|                | Review questions . . . . .                            | 8-38       |
|                | Review answers . . . . .                              | 8-39       |
|                | Exercise: Working with static domains . . . . .       | 8-40       |
|                | Exercise introduction . . . . .                       | 8-41       |
|                | Exercise: Working with dynamic domains . . . . .      | 8-42       |
|                | Exercise introduction . . . . .                       | 8-43       |
| <b>Unit 9.</b> | <b>Working with queries . . . . .</b>                 | <b>9-1</b> |
|                | Unit objectives . . . . .                             | 9-2        |
|                | Topics . . . . .                                      | 9-3        |
| 9.1.           | Searching for rule artifacts . . . . .                | 9-4        |
|                | Searching for rule artifacts . . . . .                | 9-5        |
|                | Search . . . . .                                      | 9-6        |
|                | Search for rule dependencies . . . . .                | 9-7        |
| 9.2.           | Querying rules . . . . .                              | 9-8        |
|                | Querying rules . . . . .                              | 9-9        |
|                | Introducing queries . . . . .                         | 9-10       |
|                | When queries are useful (1 of 2) . . . . .            | 9-11       |
|                | When queries are useful (2 of 2) . . . . .            | 9-12       |
|                | Business Query Language (BQL) . . . . .               | 9-13       |
|                | Query conditions . . . . .                            | 9-14       |
|                | Example of rule query conditions . . . . .            | 9-15       |
|                | Filter on properties . . . . .                        | 9-16       |

|  |              |
|--|--------------|
| Filter on definitions . . . . .                              | 9-17         |
| Filter on behavior . . . . .                                 | 9-18         |
| Query actions . . . . .                                      | 9-19         |
| Queries and ruleset extractor . . . . .                      | 9-20         |
| <b>9.3. Extracting rulesets . . . . .</b>                    | <b>9-21</b>  |
| Extracting rulesets . . . . .                                | 9-22         |
| Ruleset archive . . . . .                                    | 9-23         |
| Queries in Decision Center . . . . .                         | 9-24         |
| Queries in Decision Center . . . . .                         | 9-25         |
| Working with queries: Queries tab overview . . . . .         | 9-26         |
| Working with queries: Queries List . . . . .                 | 9-27         |
| Working with queries: Query Results . . . . .                | 9-28         |
| Unit summary . . . . .                                       | 9-29         |
| Review questions . . . . .                                   | 9-30         |
| Review answers . . . . .                                     | 9-31         |
| Exercise: Working with queries . . . . .                     | 9-32         |
| Exercise introduction . . . . .                              | 9-33         |
| <b>Unit 10. Running and debugging . . . . .</b>              | <b>10-1</b>  |
| Unit objectives . . . . .                                    | 10-2         |
| Topics . . . . .   | 10-3         |
| <b>10.1. Working with launch configurations . . . . .</b>    | <b>10-4</b>  |
| Working with launch configurations . . . . .                 | 10-5         |
| Running and debugging decision services . . . . .            | 10-6         |
| Defining launch configuration (1 of 2) . . . . .             | 10-7         |
| Defining launch configuration (2 of 2) . . . . .             | 10-8         |
| Run configuration . . . . .                                  | 10-9         |
| Debug configuration . . . . .                                | 10-10        |
| Parameters and arguments . . . . .                           | 10-11        |
| <b>10.2. Automatic exception handling . . . . .</b>          | <b>10-12</b> |
| Automatic exception handling . . . . .                       | 10-13        |
| Handling exceptions . . . . .                                | 10-14        |
| Enabling automatic exception handling . . . . .              | 10-15        |
| Capturing trace logs in Rule Designer . . . . .              | 10-16        |
| <b>10.3. Using Rule Designer debugging tools . . . . .</b>   | <b>10-17</b> |
| Using Rule Designer debugging tools . . . . .                | 10-18        |
| Rule Designer debugging tools . . . . .                      | 10-19        |
| Debug perspective: Debug view . . . . .                      | 10-20        |
| Debug perspective: Variables view . . . . .                  | 10-21        |
| Debug perspective: Breakpoints view . . . . .                | 10-22        |
| Debug perspective: Working memory and Agenda views . . . . . | 10-23        |
| Unit summary . . . . .                                       | 10-24        |
| Review questions . . . . .                                   | 10-25        |
| Review answers (1 of 2) . . . . .                            | 10-26        |
| Review answers (2 of 2) . . . . .                            | 10-27        |
| Exercise: Executing rules locally . . . . .                  | 10-28        |
| Exercise introduction . . . . .                              | 10-29        |
| Exercise: Debugging a ruleset . . . . .                      | 10-30        |
| <b>Unit 11. Enabling tests and simulations . . . . .</b>     | <b>11-1</b>  |
| Unit objectives . . . . .                                    | 11-2         |
| Topics . . . . .   | 11-3         |
| <b>11.1. Overview of testing and simulation . . . . .</b>    | <b>11-4</b>  |
| Overview of testing and simulation . . . . .                 | 11-5         |
| What is testing and simulation? . . . . .                    | 11-6         |
| What are scenarios? . . . . .                                | 11-7         |

|   |             |
|---|-------------|
| Example scenario: Loan application . . . . .                          | 11-8        |
| Decision Runner . . . . .   | 11-9        |
| Reports . . . . .   | 11-10       |
| Testing and simulation in the decision lifecycle . . . . .            | 11-11       |
| 11.2. Setting up testing and simulation . . . . .                     | 11-12       |
| Setting up testing and simulation . . . . .                           | 11-13       |
| Enabling remote testing in Business console . . . . .                 | 11-14       |
| Supporting business users for testing and simulation . . . . .        | 11-15       |
| 11.3. Working with scenarios . . . . .                                | 11-17       |
| Working with scenarios . . . . .                                      | 11-18       |
| Scenario files and the BOM . . . . .                                  | 11-19       |
| Relationships in the BOM (1 of 2) . . . . .                           | 11-20       |
| Relationships in the BOM (2 of 2) . . . . .                           | 11-21       |
| Structure of the scenario files (1 of 3) . . . . .                    | 11-22       |
| Structure of the scenario files in Microsoft Excel (2 of 3) . . . . . | 11-23       |
| Structure of the scenario files in Microsoft Excel (3 of 3) . . . . . | 11-24       |
| Expected results (1 of 2) . . . . .                                   | 11-25       |
| Expected results (2 of 2) . . . . .                                   | 11-26       |
| Adding and removing columns in the Microsoft Excel file . . . . .     | 11-27       |
| Adding columns with virtual attributes . . . . .                      | 11-28       |
| Removing columns in the scenario file . . . . .                       | 11-29       |
| Unit summary . . . . .  | 11-30       |
| Review questions . . . . .  | 11-31       |
| Review answers (1 of 2) . . . . .                                     | 11-32       |
| Review answers (2 of 2) . . . . .                                     | 11-33       |
| Exercise: Enabling rule validation . . . . .                          | 11-34       |
| Exercise introduction . . . . .                                       | 11-35       |
| <b>Unit 12. Managing deployment . . . . .</b>                         | <b>12-1</b> |
| Unit objectives . . . . .   | 12-2        |
| Topics . . . . .  | 12-3        |
| 12.1. Introducing managed execution . . . . .                         | 12-4        |
| Introducing managed execution . . . . .                               | 12-5        |
| Introduction to Rule Execution Server . . . . .                       | 12-6        |
| Key functions of Rule Execution Server . . . . .                      | 12-7        |
| Setting up a Rule Execution Server . . . . .                          | 12-8        |
| 12.2. Deploying decision services . . . . .                           | 12-10       |
| Deploying decision services . . . . .                                 | 12-11       |
| Deployment principles . . . . .                                       | 12-12       |
| RuleApp . . . . .   | 12-13       |
| Ruleset path . . . . .  | 12-14       |
| Deployed RuleApp and ruleset names . . . . .                          | 12-15       |
| RuleApp archive . . . . .   | 12-17       |
| RuleApp properties and ruleset properties . . . . .                   | 12-18       |
| Deployment configurations (1 of 4) . . . . .                          | 12-20       |
| Deployment configurations (2 of 4) . . . . .                          | 12-21       |
| Deployment configurations (3 of 4) . . . . .                          | 12-22       |
| Deployment configurations (4 of 4) . . . . .                          | 12-23       |
| Deployment from Decision Center . . . . .                             | 12-24       |
| Deployment from Rule Execution Server console . . . . .               | 12-25       |
| 12.3. Managing XOMs . . . . .   | 12-26       |
| Managing XOMs . . . . .   | 12-27       |
| XOM deployment with decision services . . . . .                       | 12-28       |
| Modifying the XOM . . . . .   | 12-29       |
| Managed Java XOM elements . . . . .                                   | 12-30       |
| Link from a ruleset to a managed Java XOM element . . . . .           | 12-31       |

|   |              |
|---|--------------|
| Java XOM deployment . . . . .   | 12-32        |
| XOM deployment from Decision Center . . . . .                                 | 12-33        |
| Managed XOM and Decision Center . . . . .                                     | 12-34        |
| Managed XOMs in Rule Designer and Decision Center . . . . .                   | 12-35        |
| Embedded managed Java XOMs . . . . .  | 12-37        |
| Managed XOM storage . . . . .   | 12-38        |
| <b>12.4. Managing resources with Ant tasks . . . . .</b>                      | <b>12-39</b> |
| Managing resources with Ant tasks . . . . .                                   | 12-40        |
| Deployment and management with Ant . . . . .                                  | 12-41        |
| Setting up the Ant environment . . . . .                                      | 12-42        |
| <b>12.5. Managing resources through the REST API . . . . .</b>                | <b>12-43</b> |
| Managing resources through the REST API . . . . .                             | 12-44        |
| REST API tool in Rule Execution Server console . . . . .                      | 12-45        |
| <b>12.6. Building RuleApps with the Build Command Maven plug-in . . . . .</b> | <b>12-46</b> |
| Building RuleApps with the Build Command Maven plug-in . . . . .              | 12-47        |
| Building RuleApps with Build Command . . . . .                                | 12-48        |
| Build and deploy . . . . .  | 12-49        |
| Project Object Model (POM) files . . . . .                                    | 12-50        |
| Example decision service POM file (1 of 2) . . . . .                          | 12-51        |
| Example decision service POM file (2 of 2) . . . . .                          | 12-52        |
| Build results . . . . .   | 12-53        |
| Unit summary . . . . .  | 12-54        |
| Review questions . . . . .  | 12-55        |
| Review answers . . . . .  | 12-56        |
| Exercise: . . . . .   | 12-57        |
| Exercise introduction . . . . .   | 12-58        |
| Exercise: . . . . .   | 12-59        |
| Exercise introduction . . . . .   | 12-60        |
| <b>Unit 13. Executing rules with Rule Execution Server . . . . .</b>          | <b>13-1</b>  |
| Unit objectives . . . . .   | 13-2         |
| Topics . . . . .  | 13-3         |
| <b>13.1. Managed execution with Rule Execution Server . . . . .</b>           | <b>13-4</b>  |
| Managed execution with Rule Execution Server . . . . .                        | 13-5         |
| Rule Execution Server in a production enterprise application . . . . .        | 13-6         |
| Introducing managed execution with Rule Execution Server . . . . .            | 13-7         |
| Example of possible platforms . . . . .                                       | 13-8         |
| <b>13.2. Rule Execution Server modular architecture . . . . .</b>             | <b>13-9</b>  |
| Rule Execution Server modular architecture . . . . .                          | 13-10        |
| Rule Execution Server architecture . . . . .                                  | 13-11        |
| Execution stack: Execution components . . . . .                               | 13-13        |
| Execution Unit . . . . .  | 13-15        |
| JMX management and execution model . . . . .                                  | 13-16        |
| Management and monitoring stack: Console . . . . .                            | 13-18        |
| Persistence layer . . . . .   | 13-19        |
| Ant tasks . . . . .   | 13-21        |
| <b>13.3. Managed execution in action . . . . .</b>                            | <b>13-22</b> |
| Managed execution in action . . . . .   | 13-23        |
| Elements of a rule-based solution . . . . .                                   | 13-24        |
| Ruleset parsing . . . . .   | 13-25        |
| Ruleset execution . . . . .   | 13-27        |
| Ruleset update at run time . . . . .  | 13-28        |
| Runtime class loading . . . . .   | 13-29        |
| <b>13.4. Platforms for Rule Execution Server . . . . .</b>                    | <b>13-30</b> |
| Platforms for Rule Execution Server . . . . .                                 | 13-31        |
| Architecture questions . . . . .  | 13-32        |

|   |             |
|---|-------------|
| Possible choices: Introduction . . . . .                            | 13-33       |
| Java SE . . . . .   | 13-35       |
| Java EE . . . . .   | 13-36       |
| Java SE and Java EE: A summary . . . . .                            | 13-38       |
| Transparent decision services . . . . .                             | 13-39       |
| 13.5. Introducing the Rule Execution Server API . . . . .           | 13-40       |
| Introducing the Rule Execution Server API . . . . .                 | 13-41       |
| Introduction . . . . .  | 13-42       |
| Execution patterns: Synchronous . . . . .                           | 13-43       |
| Execution patterns: Asynchronous . . . . .                          | 13-44       |
| Rule session API overview (1 of 2) . . . . .                        | 13-45       |
| Rule session API overview (2 of 2) . . . . .                        | 13-46       |
| Rule session factories . . . . .                                    | 13-47       |
| Rule sessions . . . . .   | 13-48       |
| Stateless rule sessions . . . . .                                   | 13-49       |
| Stateful rule sessions . . . . .                                    | 13-50       |
| Rule session requests . . . . .                                     | 13-51       |
| Rule session requests: Decision ID . . . . .                        | 13-52       |
| Rule session responses . . . . .                                    | 13-53       |
| Ruleset path for execution (1 of 2) . . . . .                       | 13-54       |
| Ruleset path for execution (2 of 2) . . . . .                       | 13-55       |
| Traces . . . . .  | 13-56       |
| Decision traces: How to . . . . .                                   | 13-58       |
| Decision traces: Complement . . . . .                               | 13-60       |
| 13.6. Executing rules in Java SE . . . . .                          | 13-61       |
| Executing rules in Java SE . . . . .                                | 13-62       |
| Rule execution in Java SE . . . . .                                 | 13-63       |
| Deployed Rule Execution Server artifacts in Java SE . . . . .       | 13-64       |
| Required Rule Execution Server API in Java SE . . . . .             | 13-65       |
| Example with a stateless rule session in Java SE . . . . .          | 13-66       |
| 13.7. Executing rules in Java EE . . . . .                          | 13-68       |
| Executing rules in Java EE . . . . .                                | 13-69       |
| Rule execution in Java EE . . . . .                                 | 13-70       |
| Deployed Rule Execution Server artifacts in Java EE . . . . .       | 13-71       |
| Required API in Java EE . . . . .                                   | 13-72       |
| Java EE POJO rule session . . . . .                                 | 13-73       |
| Java EE EJB rule session . . . . .                                  | 13-74       |
| Java EE message-driven bean (MDB) . . . . .                         | 13-75       |
| 13.8. Executing rules as transparent decision services . . . . .    | 13-76       |
| Executing rules as transparent decision services . . . . .          | 13-77       |
| Rules as transparent decision services . . . . .                    | 13-78       |
| Hosted transparent decision service . . . . .                       | 13-79       |
| Unit summary . . . . .  | 13-80       |
| Review questions . . . . .  | 13-81       |
| Review answers . . . . .  | 13-82       |
| Exercise: . . . . .   | 13-83       |
| Exercise introduction . . . . .                                     | 13-84       |
| <b>Unit 14. Auditing and monitoring ruleset execution . . . . .</b> | <b>14-1</b> |
| Unit objectives . . . . .   | 14-2        |
| Topics . . . . .  | 14-3        |
| 14.1. Auditing ruleset execution . . . . .                          | 14-4        |
| Auditing ruleset execution . . . . .                                | 14-5        |
| Auditing ruleset execution . . . . .                                | 14-6        |
| Decision Warehouse . . . . .  | 14-7        |
| Default use of Decision Warehouse . . . . .                         | 14-8        |

|   |              |
|---|--------------|
| Step 1: Install and create database resource .....                    | 14-9         |
| Step 2: Enable rule execution monitoring .....                        | 14-10        |
| Step 3: Execute the ruleset .....                                     | 14-11        |
| Step 4-a: View stored decision traces .....                           | 14-12        |
| Step 4-b: Filter stored decision traces .....                         | 14-13        |
| Step 4-c: View decision details and fired rules .....                 | 14-14        |
| <b>14.2. Monitoring ruleset execution.....</b>                        | <b>14-15</b> |
| Monitoring ruleset execution .....                                    | 14-16        |
| Monitoring ruleset execution .....                                    | 14-17        |
| Debug mode .....  | 14-18        |
| Ruleset monitoring options (1 of 5) .....                             | 14-19        |
| Ruleset monitoring options (2 of 5) .....                             | 14-20        |
| Ruleset monitoring options (3 of 5) .....                             | 14-21        |
| Ruleset monitoring options (4 of 5) .....                             | 14-22        |
| Ruleset monitoring options (5 of 5) .....                             | 14-23        |
| Ruleset statistics (1 of 3) .....                                     | 14-24        |
| Ruleset statistics (2 of 3) .....                                     | 14-25        |
| Ruleset statistics (3 of 3) .....                                     | 14-26        |
| Test of ruleset execution .....                                       | 14-27        |
| Logged events on Execution Units .....                                | 14-28        |
| Unit summary .....  | 14-29        |
| Review questions .....  | 14-30        |
| Review answers .....  | 14-31        |
| Exercise: Auditing ruleset execution through Decision Warehouse ..... | 14-32        |
| Exercise introduction .....   | 14-33        |
| <b>Unit 15. Working with the REST API.....</b>                        | <b>15-1</b>  |
| Unit objectives .....   | 15-2         |
| Topics .....  | 15-3         |
| <b>15.1. Overview of the ruleset execution REST API.....</b>          | <b>15-4</b>  |
| Overview of the ruleset execution REST API .....                      | 15-5         |
| REST API for ruleset execution .....                                  | 15-6         |
| Endpoint URIs .....   | 15-7         |
| HTTP methods .....  | 15-8         |
| Request and response schema .....                                     | 15-9         |
| XML serialization of the XOM .....                                    | 15-11        |
| WADL representation .....   | 15-12        |
| Using the REST execution API for ruleset execution .....              | 15-13        |
| <b>15.2. Testing ruleset execution with the REST API.....</b>         | <b>15-14</b> |
| Testing ruleset execution with the REST API .....                     | 15-15        |
| Executing a ruleset (1 of 2) .....                                    | 15-16        |
| Executing a ruleset (2 of 2) .....                                    | 15-17        |
| Validating the XML execution request .....                            | 15-18        |
| Validation response in JSON .....                                     | 15-19        |
| Execution response .....  | 15-20        |
| <b>15.3. Working with OpenAPI and API Connect.....</b>                | <b>15-21</b> |
| Working with OpenAPI and API Connect .....                            | 15-22        |
| OpenAPI (1 of 2) .....  | 15-23        |
| OpenAPI (2 of 2) .....  | 15-24        |
| Swagger description file .....  | 15-25        |
| Defining parameters (1 of 2) .....                                    | 15-26        |
| Defining parameters (2 of 2) .....                                    | 15-27        |
| Response .....  | 15-28        |
| Using the Swagger file with IBM API Connect .....                     | 15-29        |
| Unit summary .....  | 15-30        |
| Review questions .....  | 15-31        |

|   |             |
|---|-------------|
| Review answers .....  | 15-32       |
| Exercise: Executing rules as a hosted transparent decision service (HTDS) ..... | 15-33       |
| Exercise introduction .....   | 15-34       |
| <b>Unit 16. Introducing decision governance.....</b>                            | <b>16-1</b> |
| Unit objectives .....   | 16-2        |
| Topics .....  | 16-3        |
| 16.1. What is decision governance? .....  | 16-4        |
| What is decision governance? .....  | 16-5        |
| What is decision governance? .....  | 16-6        |
| Why decision governance is required .....                                       | 16-7        |
| Traditional approach to maintenance .....                                       | 16-8        |
| Decision management increases communication .....                               | 16-9        |
| Decision governance goal .....  | 16-11       |
| Definition of project governance .....  | 16-12       |
| Business Rule Management group .....  | 16-13       |
| Governance and agile development .....  | 16-14       |
| Implementing governance .....   | 16-15       |
| 16.2. Operational Decision Manager support for governance.....                  | 16-17       |
| Operational Decision Manager support for governance .....                       | 16-18       |
| Decision lifecycle in Operational Decision Manager (1 of 2) .....               | 16-19       |
| Decision lifecycle in Operational Decision Manager (2 of 2) .....               | 16-20       |
| Discussion .....  | 16-21       |
| 16.3. Decision governance framework .....                                       | 16-23       |
| Decision governance framework .....   | 16-24       |
| Decision governance framework overview .....                                    | 16-25       |
| Recall: Operational Decision Manager tools .....                                | 16-26       |
| Publishing decision services from Rule Designer .....                           | 16-28       |
| Decision Center: Governance in Business console .....                           | 16-29       |
| States and user roles .....   | 16-30       |
| Releases .....  | 16-31       |
| Release governance .....  | 16-32       |
| Release sequence .....  | 16-33       |
| Change activity governance .....  | 16-34       |
| Validation activity governance .....  | 16-35       |
| Release deployment .....  | 16-36       |
| Unit summary .....  | 16-37       |
| Review questions .....  | 16-38       |
| Review answers (1 of 2) .....   | 16-39       |
| Review answers (2 of 2) .....   | 16-40       |
| <b>Unit 17. Course summary .....</b>  | <b>17-1</b> |
| Unit objectives .....   | 17-2        |
| Course objectives .....   | 17-3        |
| Course objectives .....   | 17-4        |
| To learn more on the subject .....  | 17-5        |
| Earn an IBM Badge .....   | 17-6        |
| Enhance your learning with IBM resources .....                                  | 17-7        |
| Unit summary .....  | 17-8        |
| Course completion .....   | 17-9        |
| <b>Appendix A. List of abbreviations .....</b>                                  | <b>A-1</b>  |
| <b>Appendix B. Appendix: IBM ODM on Cloud .....</b>                             | <b>B-1</b>  |
| <b>Appendix C. Appendix: ODM on IBM Cloud Private (ICP) .....</b>               | <b>C-1</b>  |

---

# Trademarks

The reader should recognize that the following terms, which appear in the content of this training document, are official trademarks of IBM or other companies:

IBM, the IBM logo, and ibm.com are trademarks or registered trademarks of International Business Machines Corp., registered in many jurisdictions worldwide.

The following are trademarks of International Business Machines Corporation, registered in many jurisdictions worldwide:

|                  |            |              |
|------------------|------------|--------------|
| Bluemix®         | CICS®      | Express®     |
| IBM API Connect® | IBM Cloud™ | ILOG®        |
| Insight®         | Notes®     | Orchestrate® |
| Redbooks®        | SPSS®      | Tivoli®      |
| WebSphere®       | z/OS®      |              |

Intel, Intel Xeon and Xeon are trademarks or registered trademarks of Intel Corporation or its subsidiaries in the United States and other countries.

Microsoft and Windows 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.

VMware is a registered trademark or trademark of VMware, Inc. or its subsidiaries in the United States and/or other jurisdictions.

Social® is a trademark or registered trademark of TWC Product and Technology, LLC, an IBM Company.

Other product and service names might be trademarks of IBM or other companies.

---

## **For IBM ODM on Cloud students**

This course is designed for business analysts who work with the on-premises version of IBM Operational Decision Manager.

This course was created with IBM Operational Decision Manager V8.10, but the tools and concepts that are covered in this class also apply to IBM ODM on Cloud. Most of the tools that you work with in this course are also available in ODM on Cloud.

Relevant differences between the on-premises version of ODM and ODM on Cloud are noted throughout the course.

---

# Course description

## Developing Rule Solutions in IBM Operational Decision Manager V8.10

**Duration: 5 days**

### Purpose

This course introduces developers to IBM Operational Decision Manager V8.10. It teaches you the concepts and skills that you need to design, develop, and integrate a business rule solution with Operational Decision Manager.

The course begins with an overview of Operational Decision Manager, which is composed of two main environments: Decision Server for technical users and Decision Center for business users. The course outlines the collaboration between development and business teams during project development.

Through instructor-led presentations and hands-on lab exercises, you learn about the core features of Decision Server, which is the primary working environment for developers. You design decision services and work with the object models that are required to author and execute rule artifacts. You gain experience with deployment and execution, and work extensively with Rule Execution Server. In addition, you become familiar with rule authoring so that you can support business users to set up and customize the rule authoring and validation environments. You also learn how to use Operational Decision Manager features to support decision governance.

The lab environment for this course uses Windows Server 2012 R2 Standard Edition.

### Audience

This course is designed for application developers.

### Prerequisites

- Experience with the Java programming language and object-oriented concepts
- Knowledge of Java Platform, Enterprise Edition (Java EE)
- Basic knowledge of Extensible Markup Language (XML)
- Basic knowledge of the REST API and RESTful architecture

If you do not meet all of the requirements, you can still complete the lab exercises for this class by following the step-by-step instructions that are provided.

### Objectives

- Describe the benefits of implementing a decision management solution with Operational Decision Manager

- Identify the key user roles that are involved in designing and developing a decision management solution, and the tasks that are associated with each role
- Describe the development process of building a business rule application and the collaboration between business and development teams
- Set up and customize the Business Object Model (BOM) and vocabulary for rule authoring
- Implement the Execution Object Model (XOM) that enables rule execution
- Orchestrate rule execution through ruleflows
- Author rule artifacts to implement business policies
- Debug business rule applications to ensure that the implemented business logic is error-free
- Set up and customize testing and simulation for business users
- Package and deploy decision services to test and production environments
- Integrate decision services for managed execution within an enterprise environment
- Monitor and audit execution of decision services
- Work with Operational Decision Manager features that support decision governance

---

# Agenda

---



## Note

The following unit and exercise durations are estimates, and might not reflect every class experience.

---

## Day 1

- (00:30) Course introduction
- (01:30) Unit 1. Introducing IBM Operational Decision Manager
- (01:30) Exercise 1. Operational Decision Manager in action
- (01:30) Unit 2. Developing decision services
- (01:30) Exercise 2. Setting up decision services
- (00:30) Unit 3. Modeling decisions
- (01:00) Exercise 3. Modeling decisions

## Day 2

- (00:45) Unit 4. Programming with business rules
- (01:00) Unit 5. Developing object models
- (00:30) Exercise 4. Working with the BOM
- (00:45) Exercise 5. Refactoring
- (00:45) Unit 6. Orchestrating ruleset execution
- (00:30) Exercise 6. Working with ruleflows
- (02:00) Unit 7. Authoring rules
- (00:30) Exercise 7. Exploring action rules
- (00:45) Exercise 8. Authoring action rules

## Day 3

- (00:45) Exercise 9. Authoring decision tables
- (01:00) Unit 8. Customizing rule vocabulary with categories and domains
- (00:45) Exercise 10. Working with static domains
- (01:30) Exercise 11. Working with dynamic domains
- (00:45) Unit 9. Working with queries
- (00:45) Exercise 12. Working with queries
- (00:45) Unit 10. Running and debugging
- (00:30) Exercise 13. Executing rules locally
- (00:30) Exercise 14. Debugging a ruleset

## Day 4

- (01:15) Unit 11. Enabling tests and simulations
- (1:00) Exercise 15. Enabling rule validation
- (01:00) Unit 12. Managing deployment
- (00:45) Exercise 16. Managing deployment
- (00:30) Exercise 17. Using Build Command to build RuleApps
- (02:15) Unit 13. Executing rules with Rule Execution Server
- (00:30) Exercise 18. Exploring the Rule Execution Server console

## Day 5

- (01:00) Unit 14. Auditing and monitoring ruleset execution
- (00:45) Exercise 19. Auditing ruleset execution through Decision Warehouse
- (00:45) Unit 15. Working with the REST API
- (01:00) Exercise 20. Executing rules as a hosted transparent decision service (HTDS)
- (01:30) Unit 16. Introducing decision governance
- (00:30) Unit 17. Course summary

---

# Unit 1. Introducing IBM Operational Decision Manager

## Estimated time

01:30

## Overview

This unit introduces IBM Operational Decision Manager and describes the advantages of implementing a decision management solution in your organization.

## How you will check your progress

- Review
- Exercise

## Unit objectives

- Explain the benefits of using Operational Decision Manager
- Identify the need for governance
- Map the various roles that are involved in a decision management solution to roles in your organization
- Identify the tasks that are performed on various Operational Decision Manager modules, and which user roles perform them

Figure 1-1. Unit objectives



## How to check online for course material updates



**Note:** If your classroom does not have internet access, ask your instructor for more information.

### Instructions

1. Enter this URL in your browser:  
[ibm.biz/CloudEduCourses](http://ibm.biz/CloudEduCourses)
2. Find the product category for your course, and click the link to view all products and courses.
3. Find your course in the course list and then click the link.
4. The wiki page displays information for the course. If the course has a course corrections document, this page is where it is found.
5. If you want to download an attachment, such as a course corrections document, click the **Attachments** tab on the page.  


The screenshot shows a horizontal navigation bar with four tabs: "Comments (0)", "Versions (1)", "Attachments (1)" (which is highlighted in blue), and "About".
6. To save the file to your computer, click the document link and follow the prompts.

Figure 1-2. How to check online for course material updates

## Topics

- What is a business rule?
- Why the need for decision management?
- What is operational decision management?
- Introducing IBM Operational Decision Manager
- ODM packaging
- ODM roles
- Governance and decision management

## 1.1. What is a business rule?

## What is a business rule?

Introducing IBM Operational Decision Manager

© Copyright IBM Corporation 2019

*Figure 1-4. What is a business rule?*

## What is a business rule?

- A business rule is a statement of business logic that:
  - Business users can author and understand
  - Applications can invoke for execution
- From the business perspective:
  - A business rule is a precise statement that describes, constrains, or controls some aspect of your business
- From the IT perspective:
  - Business rules are a package of executable business policy statements that can be invoked from an application

Figure 1-5. What is a business rule?

How you define a business rule can depend on your perspective. Generally, a business rule is a statement of business logic that can be understood both by the business users who author the rules, and by the application that invokes the rules for execution.

From the business perspective, rules relate to the expression of decisions that are needed to respond to a business request. If you are a business user, you might define a business rule as a precise statement that describes, constrains, or controls some aspect of your business.

From the development or IT perspective, rules relate to the implementation of a decision system, and integration into an existing infrastructure, such as application-based or service-oriented architecture. A developer sees business rules as a package of executable business policy statements that can be invoked from an application.

For more information about business rule definitions, see the Business Rules Group website ([www.businessrulesgroup.org/home-brg.shtml](http://www.businessrulesgroup.org/home-brg.shtml)):

*“Because these perspectives are distinct, we have a definition of ‘business rule’ for each of these perspectives.”*

## From business policy to business rules

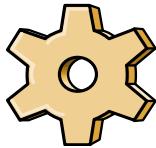
- Rules formalize business policy as “if-then” statements
- Example:

### Business policy



*When customers spend more than \$1500 in a single transaction, they should be upgraded*

### Formal rule



**If** the customer's category is Gold and the value of the customer's shopping cart is more than \$1500  
**Then** change the customer's category to Platinum

Figure 1-6. From business policy to business rules

Business policies are statements that are used for decisions, such as pricing for insurance or loan underwriting, eligibility approvals for social or health services, or product recommendations for online purchases. A single policy can require hundreds of rules to implement it.

Business rules formalize a business policy into a series of “if-then” statements.

For example, a business policy that reads as:

“When customers spend more than \$1500 in a single transaction, they should be upgraded.”

That policy might be expressed formally as the following business rule:

If the customer's category is Gold and the value of the customer's shopping cart is more than \$1500

Then change the customer's category to Platinum

In the form of business rules, the business logic can be packaged and called from the application code as a decision service. Therefore, changes to the business policy do not require changes to the application or process code.

## Discussion

What is a rule?

Introducing IBM Operational Decision Manager

© Copyright IBM Corporation 2019

*Figure 1-7. Discussion*

Now that you have the IBM definition of a rule, take a minute to write down your own definition, along with some examples of rules that are used in your organization.

This discussion is an opportunity to discover more about the work experience and expertise of your fellow students. After writing your answers, be prepared to share your ideas with the class.

## Discussion: What is a rule?

1. Write your definition of a business rule.
  
2. Provide examples of business rules, from either your domain or from the following application types:
  - Insurance: Online quotation
  - Financial services: Loan application
  - Telecommunications: Choosing a rate plan

Figure 1-8. Discussion: What is a rule?

Keeping in mind the IBM definition of a rule, take a minute to write down your own definition, along with some examples of rules that are used in your organization.

For classroom and online students: When you are done with your notes, discuss your ideas with the class.

For self-paced students: Take some time to consider these questions and write your notes here.

## 1.2. Why the need for decision management?

## Why the need for decision management?

Introducing IBM Operational Decision Manager

© Copyright IBM Corporation 2019

*Figure 1-9. Why the need for decision management?*

# IBM Training



## Why is there a need for decision management?

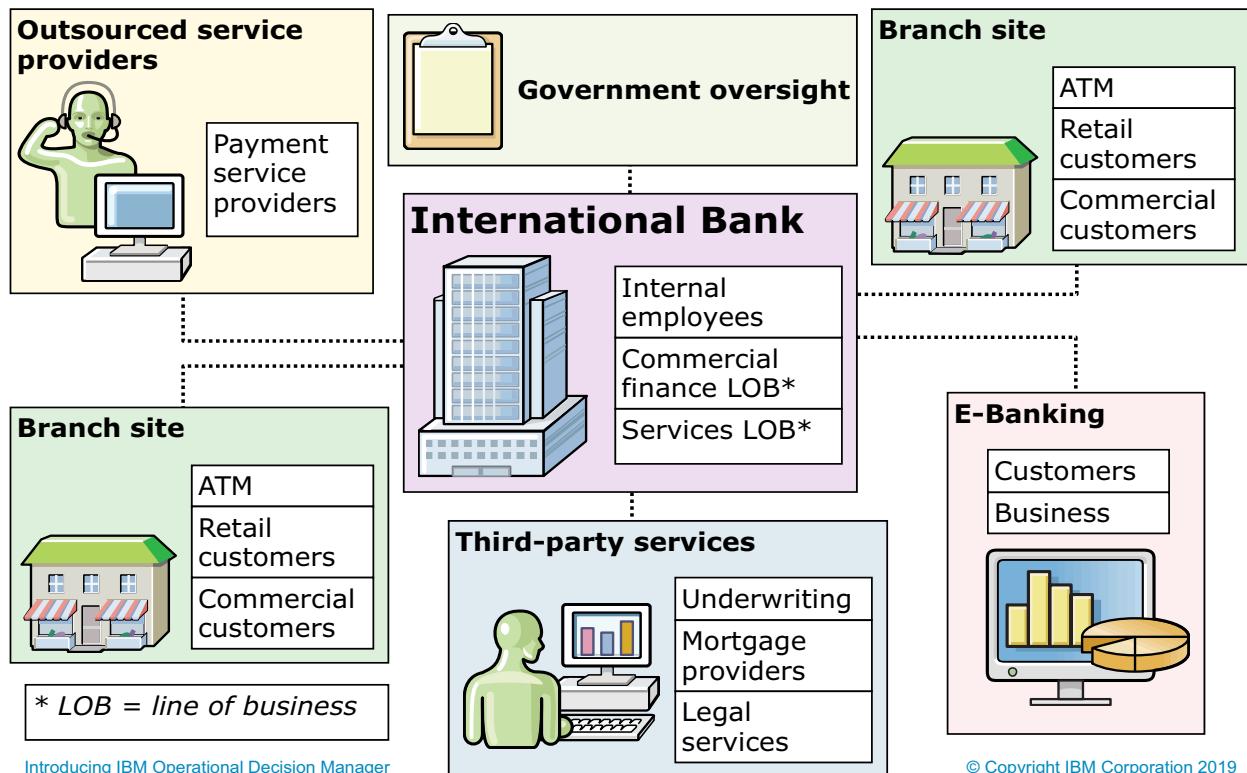


Figure 1-10. Why is there a need for decision management?

**Every** business, government, and industry is challenged with **thousands** of daily operational decisions.

Business rules have been discussed for years, but **decision management is more than just rules**.

**Decision management** combines business expertise with technology, and thus becomes an **extension** of the decisions **business people would make if they had unlimited time** to consider those decisions.

As illustrated on this slide, the financial industry is a **typical example of the current business environment** for many domains, which includes:

- Multiple channels and platforms
- Government regulations that vary according to geography
- A broad network of relationships and interactions between employees, customers, suppliers, and partners, along with internal and external processes and systems

Within this network, businesses make thousands, even millions of **repeated** and **repeatable** decisions. Those decisions need to be consistent across the organization, but might vary according to the context. The dynamic nature of this environment requires business agility to respond to change by making better decisions in real time.

For example, as illustrated on this slide, consider the financial industry. Whether the decision is to underwrite a mortgage, extend credit, or provide a loan, every day this organization makes thousands, even millions, of decisions that are repeated and repeatable.

Regardless of the business domain, the right decision must be based on the *complete* business context. The decision might differ from one situation to the next. For each business situation, you want a decision that leads to the next best course of action within the business process.

**Decision management** focuses on how to *improve repeatable* decisions through automation. Obviously, *not every business decision can be automated*. But, for *repeatable* decisions, the technology makes it possible to **record the rules** you use in your business every day, **codify those** rules in a user interface, and make them easy to change and manage.

---



## Questions

How does this picture relate to your industry and organizational structure?

---

## What is decision management?

- A business discipline that combines business expertise with software
  - Automate, optimize, and govern *repeatable* business decisions
  - Capture, change, and govern decision logic in a controlled and scalable way
  - Automate decisions so that they can be called in real time by processes, applications, and other business solutions

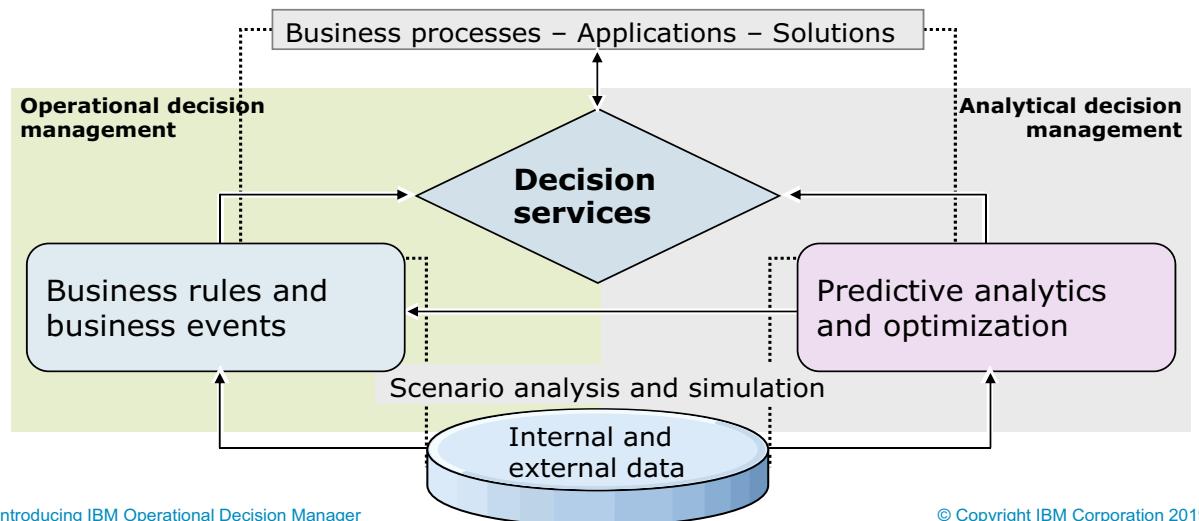


Figure 1-11. What is decision management?

Decision management is an extension of the decisions business people would make if they had unlimited time to consider those decisions. Not every business decision can be automated. However, for repeatable decisions, the technology makes it possible to record the rules you use in your business every day. You can also codify those rules in a user interface, and easily change and manage those rules so that the system can make the decision for you.

As shown in this diagram, decision services encapsulate the behavior of the automated decision. Automation of the decision logic means that those decisions can be called in real time by processes, applications, and other business solutions.

The key enablers of decision management are generally viewed to be business rules and predictive analytics. Integration between these two entry points provides a complete approach to decision management.

- (Right side of screen) In ***analytical decision management***, decision history is analyzed to build a model that can be used to predict the best decision response for the future. Decisions can be optimized for the specific contexts in which they are being made to ensure the best possible decision at the current moment and situational context. You can also use data to discover insights and continuously improve decisions over time.
- (Left side of screen) In ***operational decision management***, policy, suggested practices, and business experience are used to write rules that describe decision making and identify

situations that require a response. In many cases, the results from analytical decision management can enhance the operational decisions.

To measure the effectiveness of decision management, you define key performance indicators (KPIs) that relate to the overall business goals. By using KPIs with scenario analysis and simulation, you can assess how decision changes might affect the behavior of business systems.



### Note

Analytical decision management is not the focus of this course, but you get a brief introduction to it to see what it covers.

---

## 1.3. What is operational decision management?

## What is operational decision management?

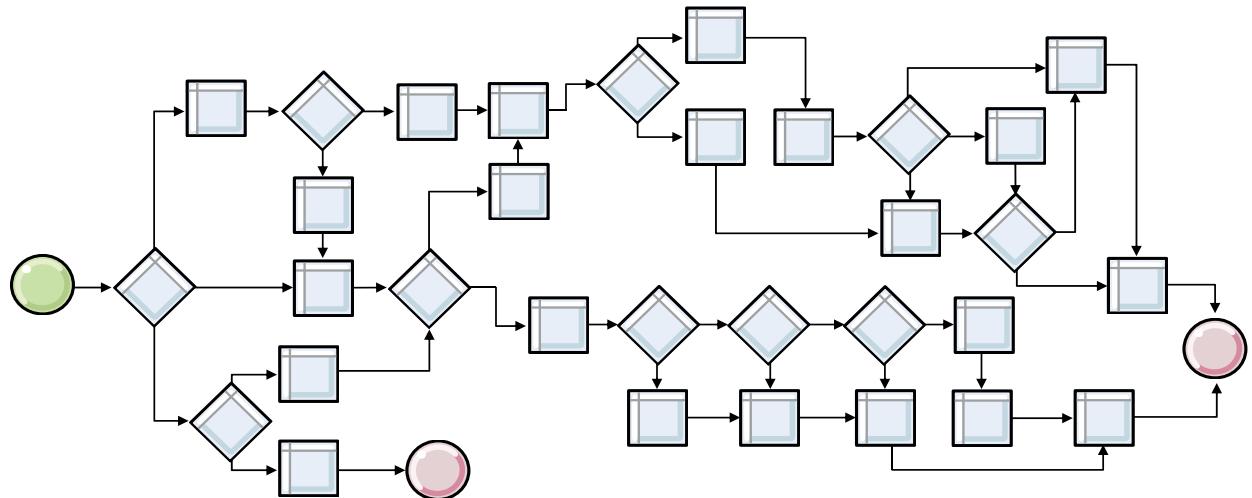
Introducing IBM Operational Decision Manager

© Copyright IBM Corporation 2019

*Figure 1-12. What is operational decision management?*

## Challenges to decision automation challenges

- Decisions are locked in processes and applications
- Programming skills are required to create and modify decision logic
- IT bandwidth limits the speed of business change
- Manual intervention increases costs and reduces customer satisfaction



Introducing IBM Operational Decision Manager

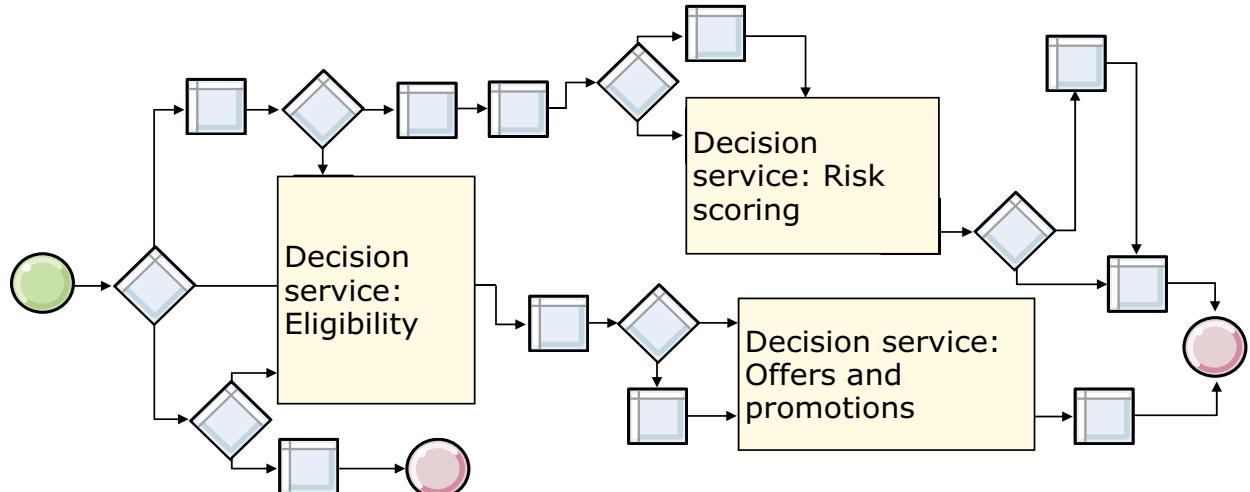
© Copyright IBM Corporation 2019

*Figure 1-13. Challenges to decision automation challenges*

Business agility depends on responsive, intelligent decision automation. However, organizations might have millions of rules that live in spreadsheets. Or, the automation of business policies might be hardcoded in the application. If so, rules might be invisible, incorrect, and unmanageable without help from the development team. Changes to policy might take months to implement, which might severely limit an organization's ability to be flexible.

## Using operational decision management

- Facilitates reuse of decision assets across processes
- Empowers business users to own, author, and update decision logic
- Increases response time to changing market conditions
- Maximizes automation and straight-through processing



Introducing IBM Operational Decision Manager

© Copyright IBM Corporation 2019

*Figure 1-14. Using operational decision management*

With operational decision management, the decision logic is externalized from processes and applications. This externalization of decision logic provides visibility for business users and enables reuse across business systems.

As depicted on this slide, many of the tasks in a process can be automated and replaced with decision services.

The separation of decision logic from application code puts business experts in control of the business logic instead of the IT team. Business experts can define and manage the business logic themselves. This separation reduces the amount of time and effort that are required to update that business logic in production systems. It also increases the ability of an organization to respond to changes in the business environment.

Decision automation also reduces the load on people, which frees them to use their skills more effectively and maximizes straight-through processing. Automation provides faster, consistent, predictable, and traceable decision making.

To manage decisions, you must first recognize the context in which decisions are made. Modeling is really the key. Decisions are always made within the context of a business process, so you start with process modeling and management. Within the process model, you can then identify the decision points and capture the individual rules that support the decision.

You might wonder about the need to manage rules separately from processes. Strong interplay exists between business process management and decision management. However, even when a *process* is not fully automated, you can still identify tasks within the process that would be better implemented as a decision point, and improve business performance.

## What is operational decision management?

- Systematic use of technology to manage the process of making operational decisions across critical business systems
- Externalizes decision logic from application code
  - Decision logic is managed independently from the application
  - Changes to business policy do not affect application code
- Empowers business users to maintain business rules directly with limited dependence on the IT department

Introducing IBM Operational Decision Manager

© Copyright IBM Corporation 2019

Figure 1-15. What is operational decision management?

Operational decision management typically involves high-volume, repeatable decisions that are based on hundreds or thousands of business rules.

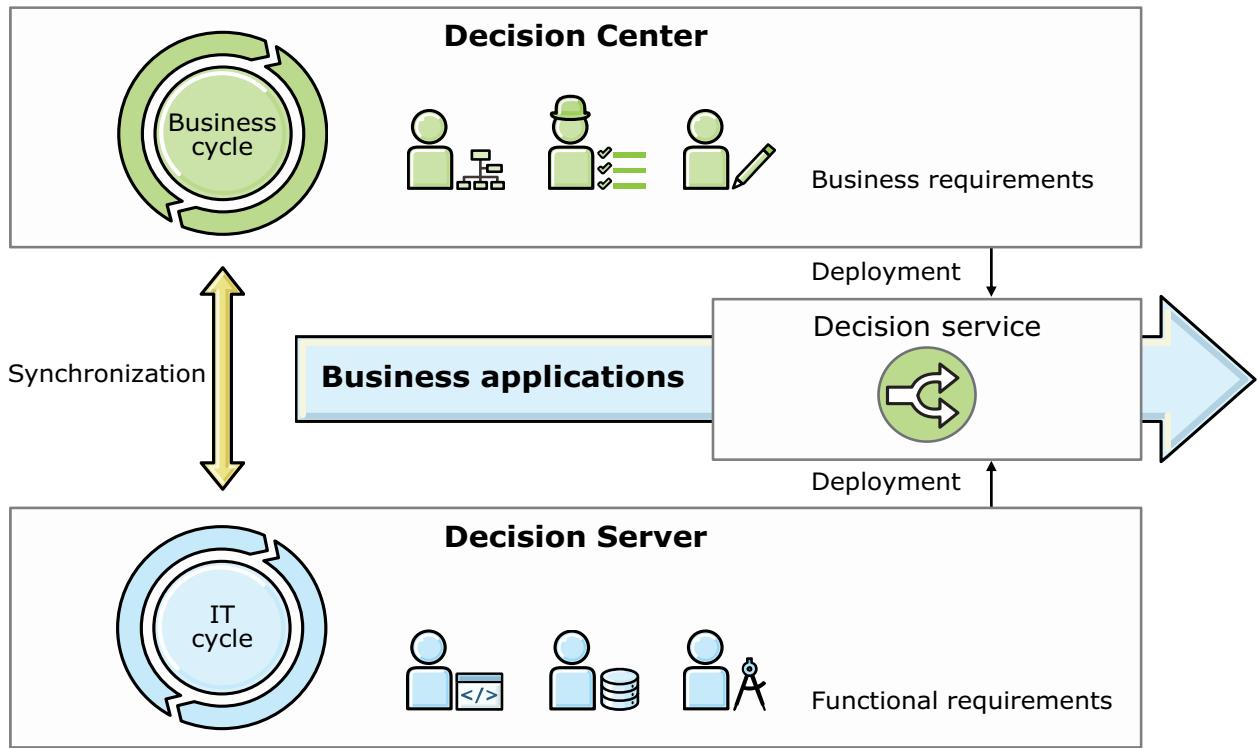
Operational decisions rely on known business expertise, such as corporate policies and external regulations. Operational decisions also rely on enterprise knowledge:

- Corporate suggested practices (explicit expertise that defines how the business is run)
- “Know-how” (implicit knowledge that is used in running the business that is not codified)

The ability to deal with change in operational systems is directly related to the decisions that those systems can make. Every transaction, order, customer interaction, or process depends on decisions, which depend, in turn, on particular internal or external requirements and situational contexts. Any change to those requirements or situations can affect decisions, including those decisions that are handled automatically within business systems.

With Operational Decision Manager, you manage decisions separately from business applications.

## Synchronized business and IT cycles



Introducing IBM Operational Decision Manager

© Copyright IBM Corporation 2019

Figure 1-16. Synchronized business and IT cycles

Decision management and application development lifecycles can evolve in parallel.

The lifecycles for decision management are much shorter than application development lifecycles. The separation of decisions from processes and applications allows these lifecycles to be managed asynchronously. Decisions can evolve as the business context requires, without putting an extra load on the application development team.

For example, application developers might work in a semi-annual cycle: a new application version is developed every six months in response to changing application infrastructure and other core business requirements. At the same time, policy managers might work on a weekly cycle to deliver decision updates in response to an evolving market or changing regulatory environment. Changes to rules do not affect code, which means that policy managers can review and even modify policies without first needing to contact the development team. Each time the application evolves, the decision management environment is synchronized with the application.

Within your organization, you can negotiate the degree of dependence between business and development teams. Dependency can range from limited review by business users of the decisions that developers implement, to giving business users complete control over the specification, creation, testing, and deployment of the rules.

## 1.4. Introducing IBM Operational Decision Manager



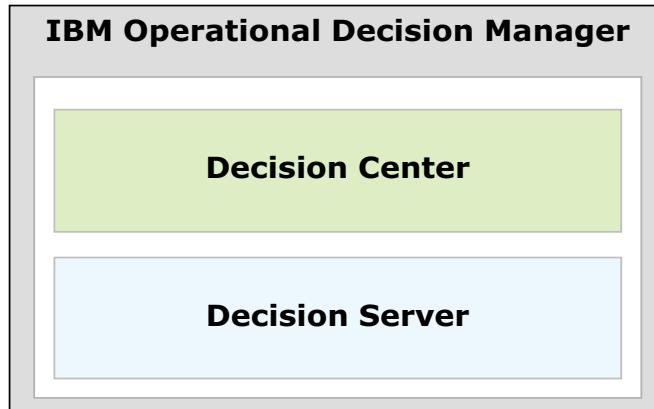
Introducing IBM Operational Decision Manager

© Copyright IBM Corporation 2019

*Figure 1-17. Introducing IBM Operational Decision Manager*

## Operational Decision Manager (1 of 2)

- Provides two main environments:
  - Decision Center for business users
  - Decision Server for technical users



[Introducing IBM Operational Decision Manager](#)

© Copyright IBM Corporation 2019

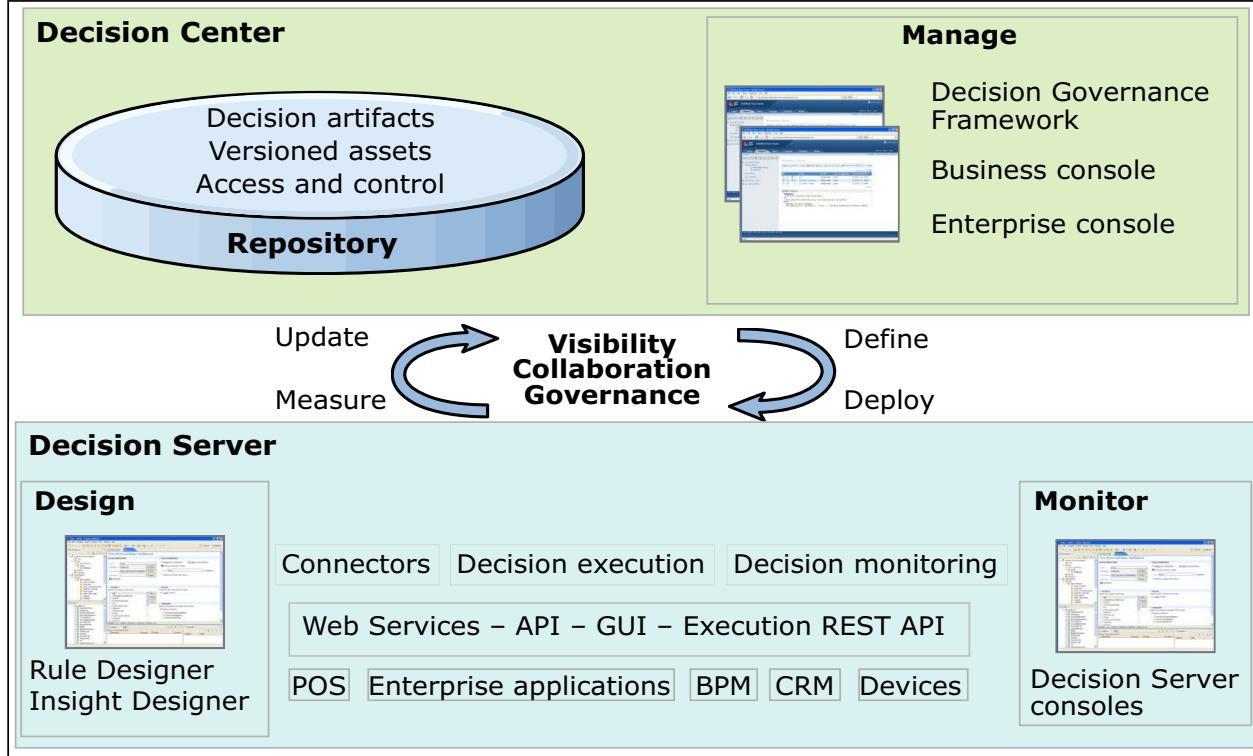
*Figure 1-18. Operational Decision Manager (1 of 2)*

Operational Decision Manager provides comprehensive capabilities to intelligently automate a wide range of decisions across business processes and applications.

- Provides organizations the ability to build highly flexible, adaptable solutions that can detect and react to data patterns as they occur within a specified time period
- Provides the appropriate decision response to transactional and process-oriented business systems
- Improves the quality of transaction and process-related decisions that are made repeatedly
- Determines the appropriate course of action, according to the specific context of each situation
- Provides an environment for business experts to author and maintain decision logic in partnership with IT
- Provides an integrated management repository for rule-based decisions and event-based decisions; this repository supports governance and change management
- Executes real-time decisions precisely and reliably based on the context of specific interactions



## Operational Decision Manager (2 of 2)



Introducing IBM Operational Decision Manager

© Copyright IBM Corporation 2019

Figure 1-19. Operational Decision Manager (2 of 2)

As you can see in this diagram, Operational Decision Manager consists of a set of modules that operate in different environments. However, these modules also work together to provide a comprehensive platform for managing and executing business rules.

**Decision Center** provides all the tools for business users to define and govern business rule and business event-based decision logic.

Through the capabilities of Decision Center, the entire organization is aligned in the implementation of automated decision services.

Decision artifacts are stored in a centralized **repository** with version control, release management, and secure access.

Decision Center also comes with these user interfaces:

- Business console, which provides a social collaboration environment for rule authoring and validation, deployment, and release management.
- Enterprise console, which is a web console that provides a full set of authoring and management capabilities.

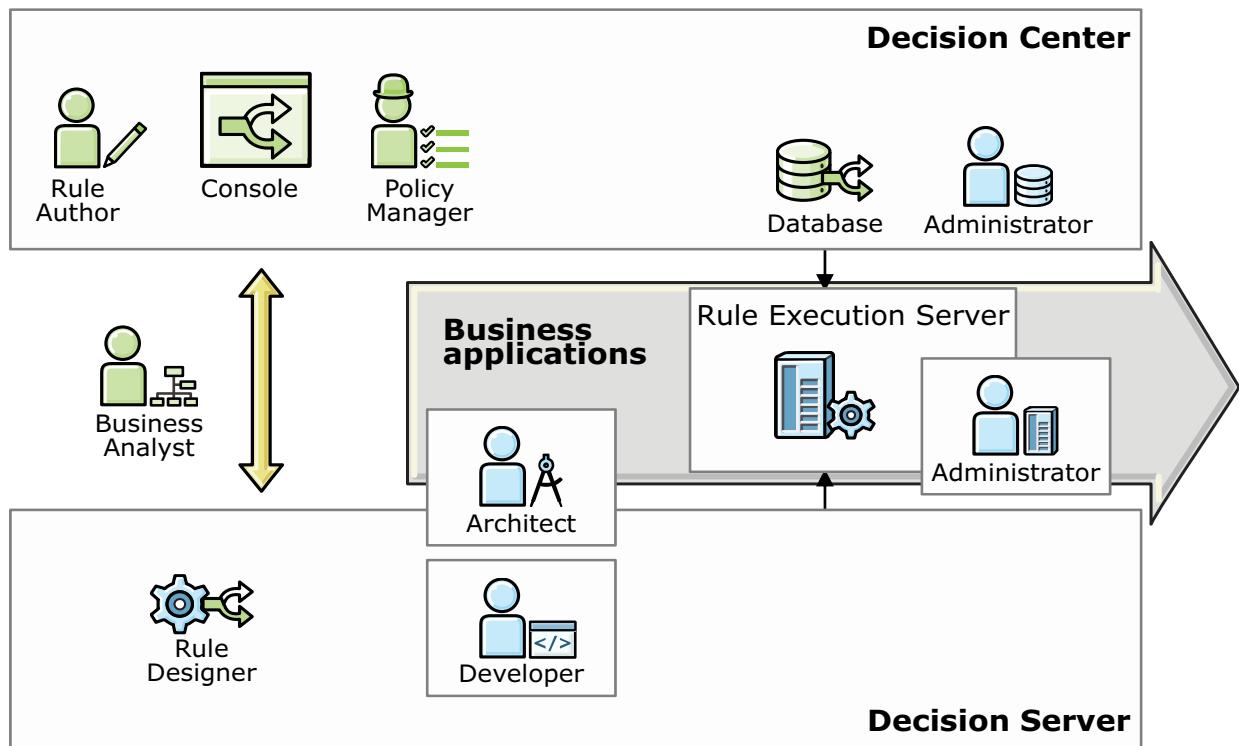
**Decision Server** is for technical users and it contains all runtime components and Eclipse-based development tools.

- Rule Designer and Insight Designer are integrated as Eclipse plug-ins so that you can design, develop, and synchronize with the business environment.
- Decision Server is also the execution environment for business rules, and it provides execution management and monitoring to detect event-based patterns, process hundreds or even thousands of business rules, and determine how to respond.

# IBM Training



## Operational Decision Manager roles and activities



Introducing IBM Operational Decision Manager

© Copyright IBM Corporation 2019

Figure 1-20. Operational Decision Manager roles and activities

The Operational Decision Manager modules are aimed at specific user roles, which are based on their varied skill sets. Synchronization mechanisms allow developers and business users to collaborate on the same project, while working in their own environments and at their own pace.

This graphic shows an overview of the different modules that are used for business rules. You see the environment in which they are used, and how they work together through synchronization and deployment.

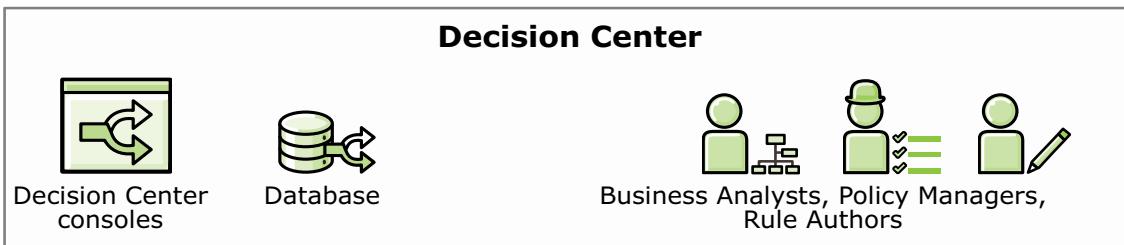
Operational Decision Manager has three main environments:

- The development environment in Designer
- The business rule management and authoring environment, in Decision Center
- The rule execution environment, which can be managed through Rule Execution Server

Next, you learn more about the modules.

## Decision Center: Authoring and maintaining business rules

- Decision Center is the main authoring and management environment for business users
  - Decision Center Business console
  - Decision Center Enterprise console
  - Decision Center database
- Compatibility with IBM Decision Composer for decision modeling
  - Online decision modeling tool: <https://decision-composer.ibm.com>
  - Models can be created directly in Business console or imported from Decision Composer



[Introducing IBM Operational Decision Manager](#)

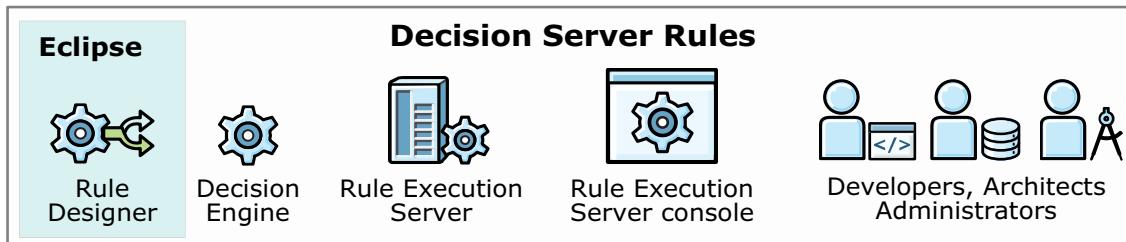
© Copyright IBM Corporation 2019

*Figure 1-21. Decision Center: Authoring and maintaining business rules*

Decision Center encompasses the modules that are shown here. Decision Center Business console includes modeling tools that allow business users to model their requirements in a DMN-inspired notation. The decision models are compatible with IBM Decision Composer, which is described later in this unit.

## Decision Server: Development and execution

- Decision Server is the technical environment for development, deployment, and execution of decision services
  - Rule Designer
  - Decision Engine
  - Rule Execution Server
  - Rule Execution Server console



Introducing IBM Operational Decision Manager

© Copyright IBM Corporation 2019

Figure 1-22. Decision Server: Development and execution

Decision Server encompasses the tools for development, deployment, integration, and managed execution of decision services.

## Decision Server Insights

- Decision Server Insights provides context awareness and situation detection through complex event processing
  - Insights Designer
  - Insight Server
  - Insight Inspector



[Introducing IBM Operational Decision Manager](#)

© Copyright IBM Corporation 2019

Figure 1-23. Decision Server Insights

Decision Server Insights has a structure similar to ODM Decision Server Rules. It includes:

- Insights Designer: A development environment in Eclipse
- Insight Server: A runtime environment that handles complex event processing and agent execution

Monitoring tools are also available, including Insight Inspector, which is a browser-based tool for visualizing event activity.

Insight Designer offers the following features:

- Eclipse interface to develop rule-based, event-driven solutions
- Develop solutions that capture business models and logic through natural-language editors
- Solutions route events to entities through agents or services and use business rules to process responses
- Solutions include model definition, business rules, and analytics
- Connectivity definitions determine inbound and outbound endpoints to receive and send events between solution and external systems

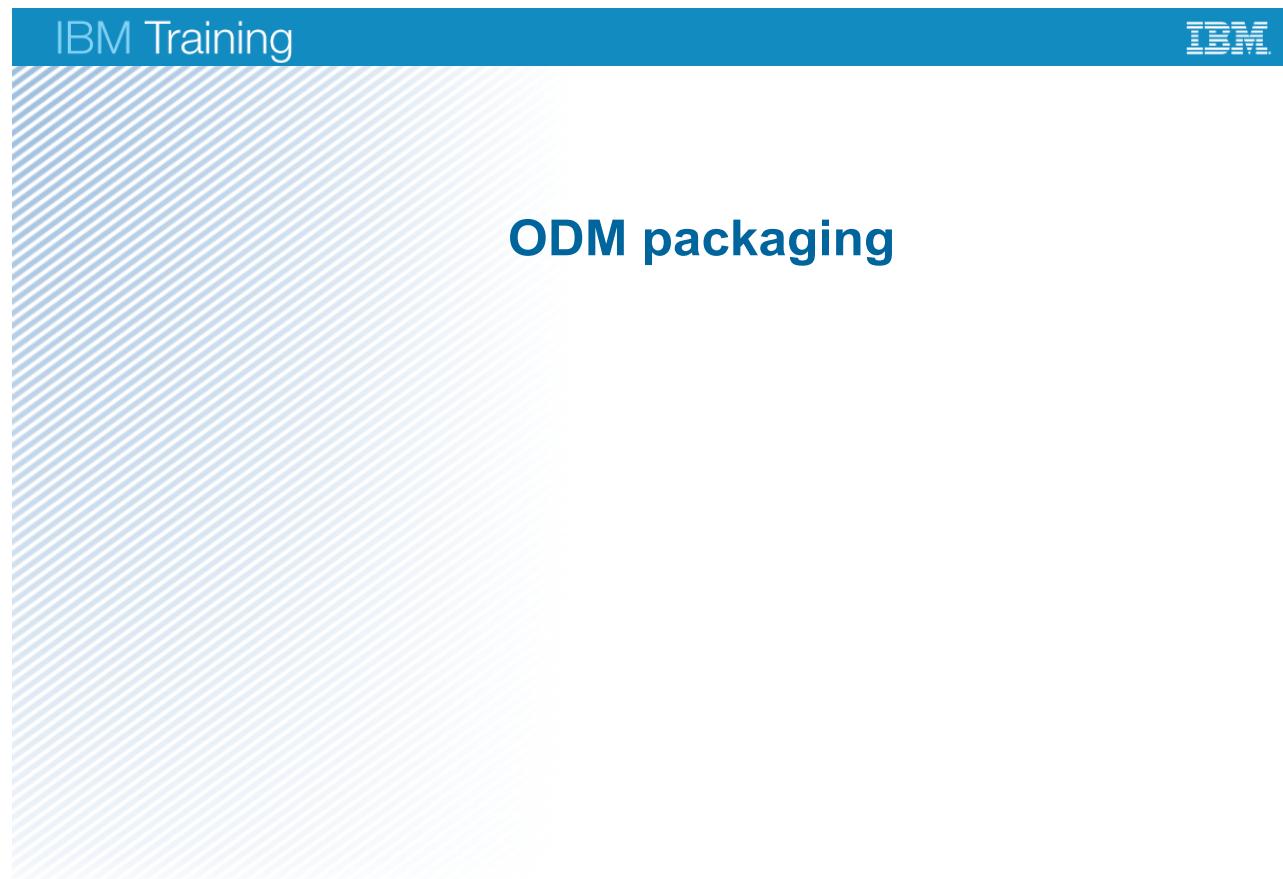
Insight Server Runtime is based on WebSphere Liberty and Extreme Scale:

- Elastic and scalable in-memory compute and data grid
- Maintains a stateful context of business entities
- Applies event-processing logic at the time of interaction

Insights Inspector provides the capability to:

- Visualize timeline of event activity for entities

## 1.5. ODM packaging



Introducing IBM Operational Decision Manager

© Copyright IBM Corporation 2019

*Figure 1-24. ODM packaging*

## Options for using ODM

ODM is available in these environments:

- On premises
  - Complete installation in your environment
- ODM on Cloud
  - Hosted environment for Decision Server and Decision Center
- ODM on IBM Cloud Private
  - Cloud platform behind your firewall
- ODM for Developers Docker
  - Free service to get started with developing rule applications
  - [hub.docker.com/r/ibmcom/odm](https://hub.docker.com/r/ibmcom/odm)

Figure 1-25. Options for using ODM

You can use ODM in a variety of environments, as you see on this slide.

This course is based on the on-premises version of ODM. However, the information that you learn here can be applied to all environments.

## On-premises editions

### ODM Server

- Decision Server Rules
- Decision Server Insights
- Decision Center

### ODM Server Express

- Decision Server Rules
- Decision Center
- *Same functionality as Standard, but with license restrictions*

Introducing IBM Operational Decision Manager

© Copyright IBM Corporation 2019

Figure 1-26. On-premises editions

Operational Decision Manager (ODM) has these on-premise editions to meet client needs:

- **Operational Decision Manager Server** provides a full-featured decision management platform that includes Decision Server Insights. You can address the entire decision automation scope, which includes business rules and decision insights capabilities.
- **Operational Decision Manager Express** provides business rule management with some licensing restrictions that make it an affordable entry point into rules-based decision management capabilities.

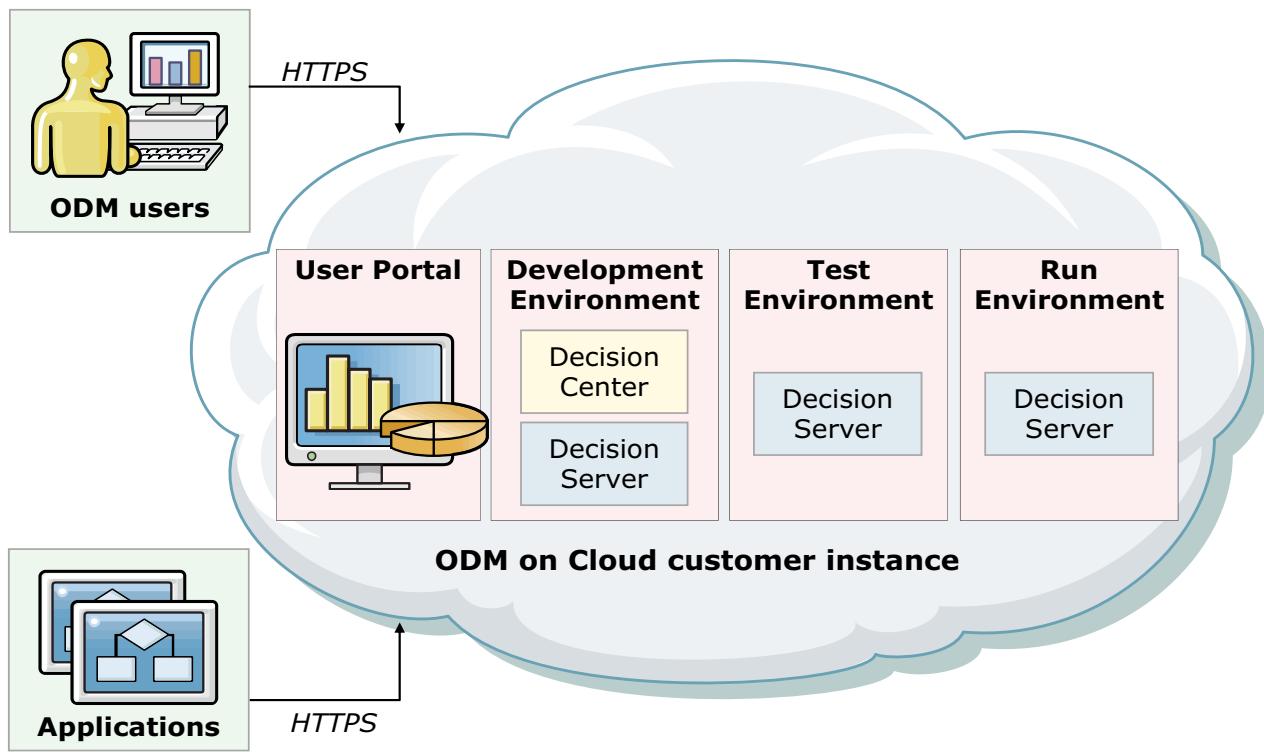
This course uses ODM Server.

For more information about the IBM Operational Decision Manager packaging, see the product documentation or visit:

[www.ibm.com/us-en/marketplace/operational-decision-manager](http://www.ibm.com/us-en/marketplace/operational-decision-manager)



## IBM ODM on Cloud: Three runtime environments



Introducing IBM Operational Decision Manager

© Copyright IBM Corporation 2019

Figure 1-27. IBM ODM on Cloud: Three runtime environments

In addition to the on-premises set of Operational Decision Manager product offerings, IBM offers IBM ODM on Cloud.

IBM ODM on Cloud is a subscription ODM service that offers a cloud-based, collaborative, and role-based environment.

With cloud computing, users can access applications or computing resources as services from anywhere through their connected devices by using a simplified user interface. Data and services can then be accessed from the cloud through connected devices over the internet.

IBM ODM on Cloud offers three runtime environments for decision management: development, testing, and running.

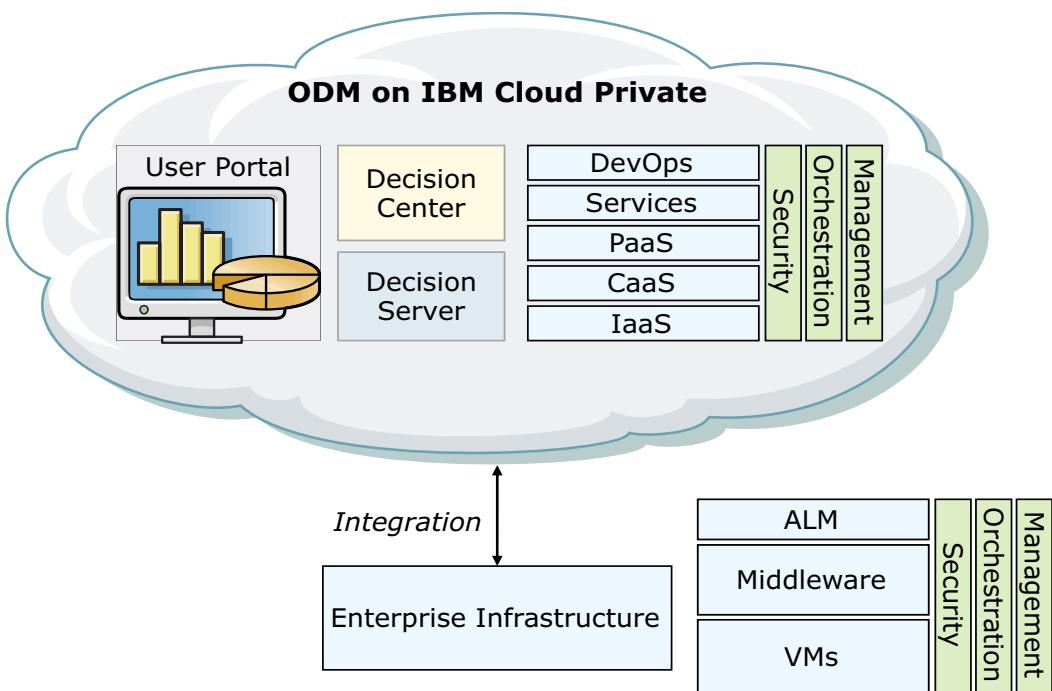
Business users work mainly in the Decision Center consoles that are available in the cloud.

For more information about IBM ODM on Cloud, see Appendix B: IBM ODM on Cloud.

IBM Training



## ODM on IBM Cloud Private



Introducing IBM Operational Decision Manager

© Copyright IBM Corporation 2019

Figure 1-28. ODM on IBM Cloud Private

IBM Cloud Private (ICP) is an integrated platform that consists of the IaaS, PaaS, and developer services necessary to create, run, and manage cloud applications.

With IBM Cloud Private, you use your own infrastructure to host services for a limited number of users behind a firewall. IBM Cloud Private, by virtue of being private, gives you control and flexibility in the way you set up, configure, and run your services.

Using ODM on IBM Cloud Private you create and provision multiple machines, change computing resources on-demand, scale applications, and take advantage of a self-service platform.

For more information about IBM ODM on IBM Cloud Private, see Appendix B: IBM ODM on IBM Cloud Private (ICP).

## ODM for Developers Docker

- ODM topology with Decision Server Rules and Decision Center on a single Docker image
- Free service to get started with developing rule applications
- Samples and tutorials provided to help you evaluate product
- See:
  - [hub.docker.com/r/ibmcom/odm](https://hub.docker.com/r/ibmcom/odm)
  - <https://odmdev.github.io/odm-ondocker>

Figure 1-29. ODM for Developers Docker

You can find more information about ODM for Developers at:

<https://hub.docker.com/r/ibmcom/odm>

## 1.6. ODM roles



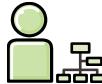
Introducing IBM Operational Decision Manager

© Copyright IBM Corporation 2019

*Figure 1-30. ODM roles*

## Operational Decision Manager user roles

### Business users



Business analyst

Bridges between the business side and technical side of a business rule application



Policy manager

Business expert and owner of business policy



Rule author

Business domain expert who updates and reviews rules

### Technical users



Architect

Manages overall deployment, organization of rules, and optimization of rule execution



Developer

Develops, tests, and deploys business rule applications and event applications



Administrator

Installs and configures rule management and execution environments

[Introducing IBM Operational Decision Manager](#)

© Copyright IBM Corporation 2019

Figure 1-31. Operational Decision Manager user roles

During the development of a decision management solution, various skills are required at different stages of the lifecycle. In ODM, these skills are grouped into a set of business and technical roles.

Developing and maintaining a decision management solution involves various skills that are grouped into two categories:

- Business users: Business analysts, policy managers, and rule authors develop and maintain the decision logic.
- Technical users: Architects, developers, and administrators develop and maintain the rule application.

These roles are explained in more detail in the upcoming slides.

## Interaction between roles

- Roles do not correspond to individuals, but to activities and responsibilities
- Tasks might not correspond to a single position in your organization
  - A business expert might be involved in the technical side of things
  - A developer might also be the person who authors and manages the rules
- Communication between the business and technical roles is vital

Figure 1-32. *Interaction between roles*

Roles refer to tasks and responsibilities rather than individuals.

Roles might not correspond to a single position in your organization, so crossover between departments might make it difficult to discern who fits into a particular role. For example, a business policy expert might be involved in the technical side of things, and a developer might also be the person who writes and manages the rules.

Expect extensive interaction between business roles and technical roles, particularly during the early stages of developing an application. Having this expectation can ensure that the implementation meets the business view.

## Business analyst

- Responsibilities:
  - Designing a formal specification for the rules, with validation from both developers and policy managers
  - Defining the vocabulary that is used in rules
  - Writing and organizing business rules so that rule authors can maintain them
  - Validating that rule execution yields the expected results
- Tools: Designer and Decision Center



Introducing IBM Operational Decision Manager

© Copyright IBM Corporation 2019

Figure 1-33. Business analyst

Business analysts act as a bridge between the business and technical sides of a business rule application.

Business analysts are involved in rule discovery tasks, including process modeling, writing use cases, and defining the vocabulary that is used in rules. They also work with developers to ensure that the implementation matches the business requirements.

Depending on their level of technical knowledge, business analysts can do tasks that are currently described as developer tasks. However, business analysts generally do not write code.

## Policy manager

- Responsibilities:
  - Participating in the design of a formal specification for the rules
  - Defining vocabulary elements with the help of business analysts
  - Creating and updating rules
  - Reviewing how the execution of rules is orchestrated
  - Reporting on the status of the business policy
  - Testing rules to ensure that they are written correctly
  - Running simulations to ensure that the rules give the intended business outcome
  - Managing multiple releases
- Tools: Decision Center Business and Enterprise Consoles



Introducing IBM Operational Decision Manager

© Copyright IBM Corporation 2019

*Figure 1-34. Policy manager*

Policy managers are owners of the decisions within an organization, and are involved in the review, validation, and authoring activities of the decision lifecycle. Their responsibilities are listed on the slide.

Policy managers can work in Decision Center or Rule Solutions for Office.

A policy manager is a business expert who is responsible for defining business policy definitions, participating in rule discovery and validation of results, and reviewing how the execution of rules is organized.

Examples of policy managers include actuaries, underwriters, and compliance officers for insurance companies or those personnel who are in charge of underwriting or pricing in mortgage providers.

Policy managers work with business analysts during the initial discovery phase to validate that business requirements are captured accurately.

## Rule author

- Responsibilities:
  - Updating and sometimes creating rules
  - Reviewing the business rules
- Tools: Decision Center Business and Enterprise Consoles



Introducing IBM Operational Decision Manager

© Copyright IBM Corporation 2019

Figure 1-35. Rule author

A rule author is a business domain expert who formulates policies into business rules. Rule authors can work in Decision Center or Rule Solutions for Office to update and create rules. They also work with queries and reports to review business rules and event rules.

## Architect

- Responsibilities:
  - Managing the overall deployment organization of the rules and making sure that their execution is optimized
  - Defining the project organization so that it is convenient for developers and business users alike
  - Defining the granularity of the rule applications and how they fit into the wider business process
- Tools: Designer



Introducing IBM Operational Decision Manager

© Copyright IBM Corporation 2019

Figure 1-36. Architect

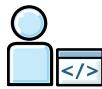
Architects work in Designer. Their responsibilities are listed on the slide.

An architect ensures overall deployment, organization of rules, and optimization of rule execution.

The architect defines the types of rules that are used, and orchestrates their execution in a business rule application. The architect also ensures coherent rule deployment across several business rule applications.

## Developer

- **Responsibilities:**
  - Developing, testing, debugging, and deploying business rule applications
  - Writing the code for rule execution
- **Tools:** Rule Designer



Introducing IBM Operational Decision Manager

© Copyright IBM Corporation 2019

Figure 1-37. Developer

Developers are familiar with object models, APIs, and the development environment (Java EE application servers, Java SE, and z/OS platforms). Developers are involved in the design, author, test, integration, and deployment activities of the decision lifecycle.

Developers work in Designer to do these tasks:

- Work with business analysts to implement business rule vocabulary
- Set up the authoring environment for rule authors
- Write the invocation code for rule execution
- Write complex rules that business users cannot write
- Implement customizations to meet specific needs

## Administrator

- Responsibilities:
  - Deploying and configuring the server and database for Decision Center and Rule Execution Server
  - Managing user access to Decision Center and Rule Execution Server
  - Configuring trace data sources for testing purposes
  - Deploying applications
  - Redeploying rulesets and event assets as changes are made
  - Generating detailed execution reports
  - Tracking and monitoring rule execution
  - Restoring a particular application state
- Tools: Servers for Decision Center or runtime environments



Introducing IBM Operational Decision Manager

© Copyright IBM Corporation 2019

Figure 1-38. Administrator

Administrators install and configure rule management and execution environments. Their responsibilities are listed on the slide.

System administrators work on the servers to ensure that they run smoothly. These servers can be for Decision Center or runtime environments.

## Discussion

Who is in charge of what?

Introducing IBM Operational Decision Manager

© Copyright IBM Corporation 2019

*Figure 1-39. Discussion*

Take a moment to consider the following questions and map the people from your organization to the role descriptions.

If your organization is involved in an active decision management project, the results from this exercise can be useful to the project manager. It is important to identify, early in a project, which people to include on the rule management team after the solution goes live.

1. Who is in charge of capturing rules from the business policies?

---

2. Who is in charge of authoring rules?

---

3. Who is in charge of deploying rules?

---

4. How does your current role relate to the BRMS roles?

## 1.7. Governance and decision management

## Governance and decision management

Introducing IBM Operational Decision Manager

© Copyright IBM Corporation 2019

*Figure 1-40. Governance and decision management*

## Applying governance to decision management

- Governance is management of the decision logic lifecycle
  - Govern lifecycle from initial development to deployment and maintenance
- Provides an organizational framework to prevent problems:
  - Defines expectations
  - Assigns roles and responsibilities
  - Verifies performance
- Goals:
  - Efficient collaboration between business and IT teams
  - Ability to demonstrate that an organization accomplished what it said it would accomplish

[Introducing IBM Operational Decision Manager](#)

© Copyright IBM Corporation 2019

*Figure 1-41. Applying governance to decision management*

The advantages of implementing a decision management solution can be lost when governance is not included. Governance is the management of the decision logic lifecycle, from initial development to deployment and maintenance.

Implementing decision management requires that development teams collaborate regularly with business users, starting with the initial design phases of a project. The business team knows which requirements form the basis of decisions. Modeling makes it easier for the business team to understand the business logic, and how it can be implemented, which makes the system more maintainable.

Governance processes, change management, and testing features also alleviate fear of the potential side effects of rule changes. Agile development involves interaction between these teams daily. Regardless of the development methodology that your organization uses (such as waterfall), both the development and business teams must interact regularly. This regular interaction helps to ensure that developers understand “what was meant” by business users, not just “what was said.”

Applying governance to decision management encompasses people, processes, and goals across your organization. By defining expectations, assigning responsibilities, and verifying performance, governance provides an organizational framework that prevents potential problems. It facilitates efficient collaboration between business and IT teams, and makes it possible for an organization to demonstrate that it accomplished what it said it would accomplish.

## Governance in Operational Decision Manager

- Operational Decision Manager supports governance and change management through the decision governance framework
  - The decision governance framework is a ready-to-use method for applying governance principles to a BRMS
- Decision governance is based on the states of releases and activities within a decision service lifecycle, and the roles of users who work on these releases and activities

[Introducing IBM Operational Decision Manager](#)

© Copyright IBM Corporation 2019

*Figure 1-42. Governance in Operational Decision Manager*

In Operational Decision Manager, the decision governance framework is a built-in framework to help your organization apply governance and change management principles.

This framework is defined around the change management activities and releases of a decision service lifecycle. The decision service includes all the rules and projects that are required to produce a decision.

The decision service and the decision governance framework are specially designed to support rule authors who work in the Decision Center Business console.

You learn more about decision services and the decision governance framework later in this course.

## Unit summary

- Explain the benefits of using Operational Decision Manager
- Identify the need for governance
- Map the various roles that are involved in a decision management solution to roles in your organization
- Identify the tasks that are performed on various Operational Decision Manager modules, and which user roles perform them

Figure 1-43. Unit summary

## Review questions (1 of 2)

1. True or False: Operational decision management combines business expertise with technology to automate repeated and repeatable decisions.
2. True or False: Because the length of application lifecycles and decision lifecycles usually take about the same amount of time, developers can use Operational Decision Manager to update rules every six months, when the application infrastructure is updated.



[Introducing IBM Operational Decision Manager](#)

© Copyright IBM Corporation 2019

*Figure 1-44. Review questions (1 of 2)*

Write your answers here:

- 1.
- 2.

## Review questions (2 of 2)

3. Operational Decision Manager Server includes which components:
  - A. Decision Server Rules
  - B. Decision Server Insights
  - C. Decision Center
  - D. All of the above
  
4. Which of the following roles are involved during the early stages of a business rule application development project? Select all that apply.
  - A. Policy manager
  - B. Business analyst
  - C. Architect
  - D. Developer



Introducing IBM Operational Decision Manager

© Copyright IBM Corporation 2019

Figure 1-45. Review questions (2 of 2)

Write your answers here:

3.

4.

## Review answers (1 of 2)

1. True or False: Operational decision management combines business expertise with technology to automate repeated and repeatable decisions.  
The answer is True.
  
2. True or False: Because the length of application lifecycles and decision lifecycles usually take about the same amount of time, developers can use Operational Decision Manager to update rules every six months, when the application infrastructure is updated.  
The answer is False. Business policies evolve more rapidly than the application infrastructure. By using Operational Decision Manager, you can manage the decision lifecycle and the application infrastructure lifecycle asynchronously.



Figure 1-46. Review answers (1 of 2)

## Review answers (2 of 2)



3. Operational Decision Manager Server includes which components:

- A. [Decision Server Rules](#)
- B. Decision Server Insights
- C. [Decision Center](#)
- D. All of the above

The answer is A and C.

4. Which of the following roles are involved during the early stages of a business rule application development project?  
Select all that apply.

- A. [Policy manager](#)
- B. [Business analyst](#)
- C. [Architect](#)
- D. [Developer](#)

The answer is A, B, C, and D.

Figure 1-47. Review answers (2 of 2)

## Exercise: Operational Decision Manager in action

Introducing IBM Operational Decision Manager

© Copyright IBM Corporation 2019

*Figure 1-48. Exercise: Operational Decision Manager in action*

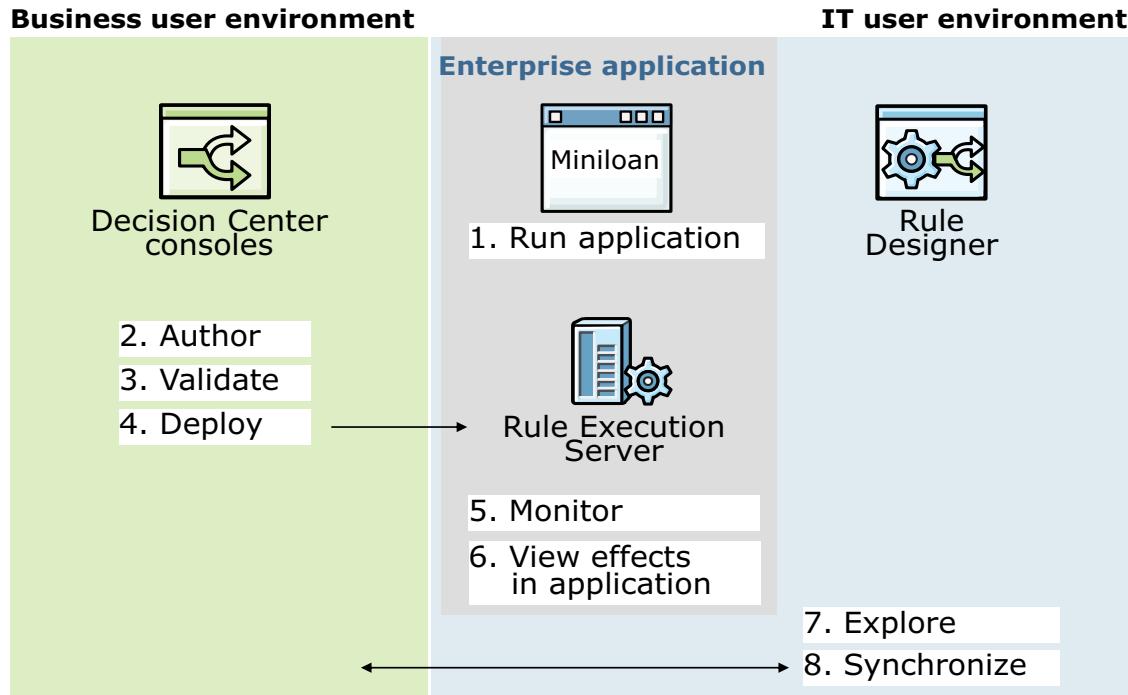
## Exercise introduction

- Explain the general workflow in Operational Decision Manager for working with business rule projects
- Identify the Operational Decision Manager tasks that apply to your role in your organization



*Figure 1-49. Exercise introduction*

## Exercise workflow



Introducing IBM Operational Decision Manager

© Copyright IBM Corporation 2019

Figure 1-50. Exercise workflow

This diagram depicts the workflow that you follow during the exercise. After you run the application that calls business rules, you modify the rules and redeploy them to see how easily change can be applied.

Take a moment to read through the steps that are outlined here, which describe the workflow that you see in the exercise.

# Unit 2. Developing decision services

## Estimated time

01:30

## Overview

This unit teaches you how to get started with development of decision services.

## How you will check your progress

- Review
- Exercise

## Unit objectives

- Identify the development tasks in building a decision management application
- Describe how to set up a decision service in Rule Designer
- Share and synchronize decision services between the business and development environments

Figure 2-1. Unit objectives

## Topics

- Using an agile development approach
- Designing the business rule application
- Setting up decision services
- Project properties
- Using modular project organization
- Sharing and synchronizing decision services

Developing decision services

© Copyright IBM Corporation 2019

*Figure 2-2. Topics*

## 2.1. Using an agile development approach

## Using an agile development approach

Developing decision services

© Copyright IBM Corporation 2019

*Figure 2-3. Using an agile development approach*



## Lifecycle of business decisions

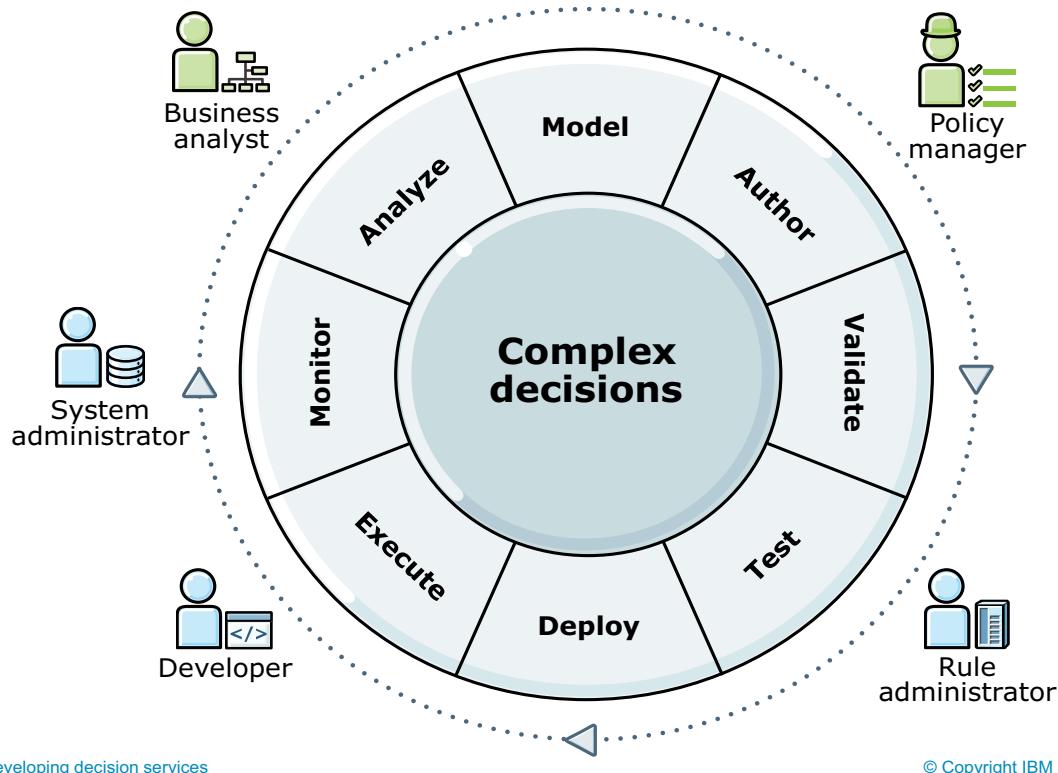
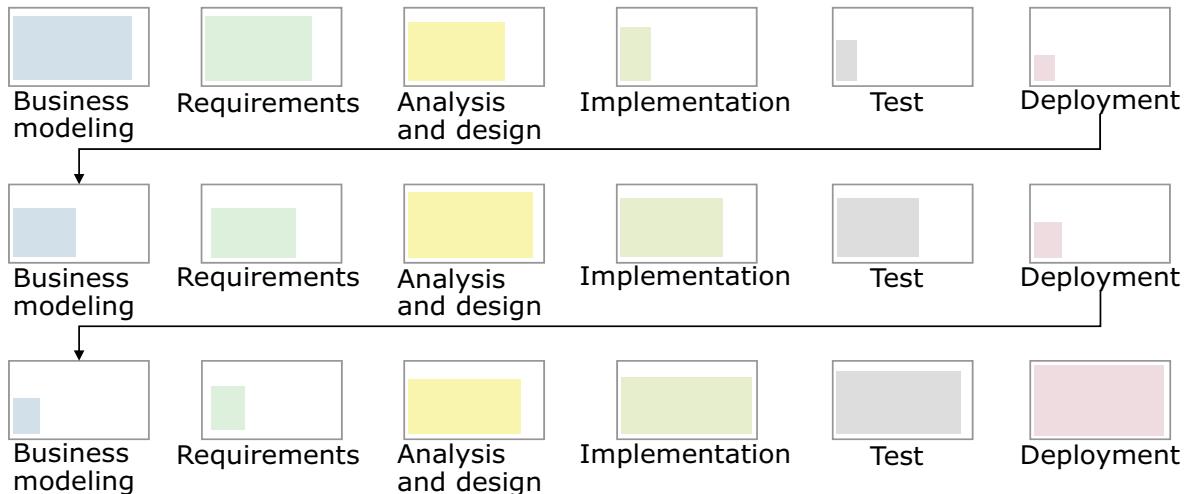


Figure 2-4. Lifecycle of business decisions

The full lifecycle of business decisions includes the phases that you see here.

## Agile Business Rule Development methodology

- Use an agile, or iterative, approach to implement decision management incrementally
- Agile Business Rule Development (ABRD) includes a set of phases and activities



Developing decision services

© Copyright IBM Corporation 2019

Figure 2-5. Agile Business Rule Development methodology

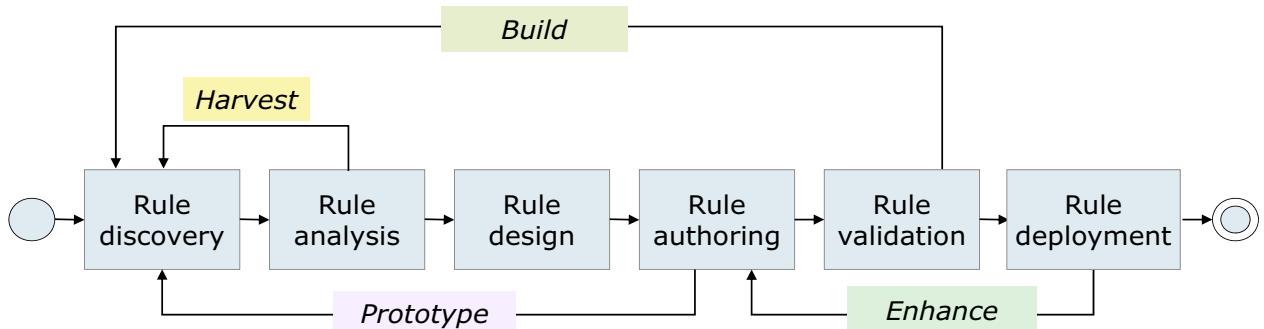
The overall implementation of decision management should follow an agile and iterative approach that responds easily to change.

The Agile Business Rule Development methodology, or ABRD, includes a sequence of phases. Each phase includes iterations on a set of activities.

Initial activities concentrate on business modeling, requirements gathering, analysis, and design. Early results from these activities can be implemented and tested to ensure that requirements were captured correctly. As the project evolves and requirements stabilize, more effort can be spent on implementation, testing, and deployment activities.

# Agile development approach

- Use an iterative, incremental approach to implement decision management
    - Architects and business users collaborate to model requirements
    - Developers design the rule project structure, implement, test, and deploy



## Developing decision services

© Copyright IBM Corporation 2019

*Figure 2-6. Agile development approach*

Each ABRD phase includes iterations on a set of activities. The goal is to deliver a workable set of rules at the end of each phase: first on paper, then as a prototype in Designer.

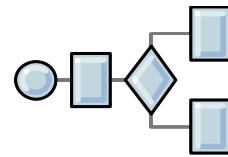
ABRD includes rule discovery and rule analysis as the first phases of approach. Before implementing decisions, you first must find, or *discover*, them from the various rule sources. Business analysts and architects usually work together during the discover phase to ensure that the logic is expressed correctly “on paper” before moving to the tools. The logic is also analyzed for overlaps, consistency, and gaps. The “paper” results from these phases become the application requirements that get passed to the development team.

During the discovery and analysis phases, developers can already start working on the *design* of the application, which includes writing code to implement business object models. Developers start setting up the rule authoring environment in Rule Designer according to the early results of discovery, while business users continue with discovery tasks.

ABRD supports frequent requirement and model changes during early phases of the project as business owners provide early feedback on implementation.

## Discovery: Identifying decisions in the process (1 of 2)

- Start with models to understand the business context
  - Process models to provide the decision context
  - Use cases to identify individual rules and vocabulary
  - Class diagrams and object models to represent the vocabulary



- Roles: Architects, business analysts, and policy managers



- Developers begin implementation according to these requirements
  - Clear, unambiguous requirements can reduce back-and-forth validation between development and business users



Figure 2-7. Discovery: Identifying decisions in the process (1 of 2)

The first phase of a decision management project is discovery. Architects work with business analysts, policy managers, and other business experts to identify where rules can be used within your business process.

Models are the starting point to define the scope of what the business rule application should do and establish the context for decisions. Without this context, you might end up implementing the wrong rules.

Rule discovery includes identifying rule sources, extracting the rules from those sources, and documenting them. Discovery can involve interviewing the business experts or policy managers, reading documentation, or analyzing the source code. Usually, the best source for business policies is people, which means that capturing those rules requires extensive interviews with the business policy experts. If rule discovery is not done thoroughly, the knowledge gap between business experts and IT can be an issue when implementing policy requirements in core business applications.

The main purpose of modeling is communication. Models help ensure that business context and rules are captured in a clear and unambiguous way so that the business policy managers can validate them and the development team can implement them.

When preparing requirements, the business users should keep in mind these goals:

- Formalize business process flow. Identifying the business objectives, the activities of the process, and the actors or roles that perform those activities provides the business context for the rules. The rules will be geared towards attaining the objectives.
- Identify the decision points. Decisions that are made within the process can usually be broken down into subdecisions, which are often the underlying rules.
- Define the underlying object model. Since business rules are written about “things” or data, modeling the business helps identify the data that will make up the underlying object model, which is the basis for the rule vocabulary.
- Detail the rule execution logic. Rules influence the business process flow. Wherever decisions appear in the process, the results can provide the basis for taking alternative paths through the process flow.

## Discovery: Identifying decisions in the process (2 of 2)

- Determine where decisions are made within the scope of the process:
  - Which decisions require human intervention?
  - Which decisions can the system make?
- Focus on automated decision points
  - Example: Online insurance application

| Process                       | Decision point   |
|-------------------------------|--|
| Browse to website             | -  |
| Choose insurance type         | <b>Manual</b>  |
| Give personal information     | -  |
| System verifies information   | <b>Automated:</b> <i>Is the information valid?</i>             |
| System determines eligibility | <b>Automated:</b> <i>Is the client eligible?</i>               |
| System determines price       | <b>Automated:</b> <i>What price applies based on criteria?</i> |
| System returns quotation      | -  |
| Accept or reject quotation    | <b>Manual</b>  |

Figure 2-8. Discovery: Identifying decisions in the process (2 of 2)

Before you can *implement* your rules, you must determine where the system uses your rules.

Within the scope of a business process, some points require a decision before proceeding. Some decisions require people to make them, but others can be automated.

- Decisions that people make are called manual decision points
- Decisions that can be automated are called automated decision points

You focus on the automated decision points for implementation.

The example here for an online insurance application includes multiple decision points. The process is outlined step-by-step from the point of view of a user who browses to a website to get an insurance quotation.

When the user sees the initial web page that lists insurance options, who makes the selection? Obviously, it is the user, so this step is a manual decision point.

Next, after the user enters personal information, who validates that information? Who determines whether the user is eligible for insurance? Who calculates the price? These steps are all automated decision points, where you can implement business rules.

By identifying all decision points in a process, you can determine which decisions require people versus which decisions can be automated to use rules.

## Discovery: Identifying the rules

- Based on decision points, determine underlying rules that are required to implement the business decision
  - A single business decision might require firing several, even hundreds of business rules
- Example: Processing a loan request
  - Business decision: Can the loan be approved?
- Series of smaller decisions determine result for main business decision
  - Is the personal information valid?
  - How much is the client eligible to borrow?
  - Is the client able to repay the loan?
  - Each of these smaller decisions might also require several rules to implement the business logic

*Figure 2-9. Discovery: Identifying the rules*

After identifying the automated decision points, you then determine the underlying rules that are required to express the business policy. A single business decision might require several, even hundreds of business rules to express it. Business logic usually requires a series of smaller, intermediate decisions before the overall decision can be determined.

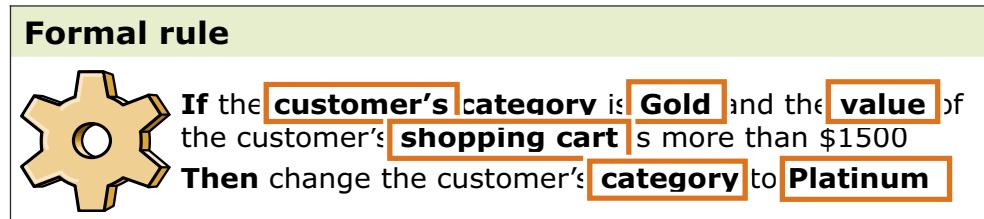
For example, when processing a loan request, the main business decision is whether to grant a loan to an applicant. When you break down that decision, you discover a series of other smaller decisions, such as:

- Did the person provide all the required personal information?
- Is the information valid?
- How much is the client eligible to borrow?
- Is the client able to repay the loan?

Each of these intermediate decisions can again be further broken down into several underlying rules to implement the logic.

## Discovery: Identifying vocabulary

- Identify and document the vocabulary that is required to formulate the rules



- Formalize vocabulary as a conceptual object model



1..\*

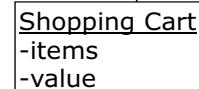


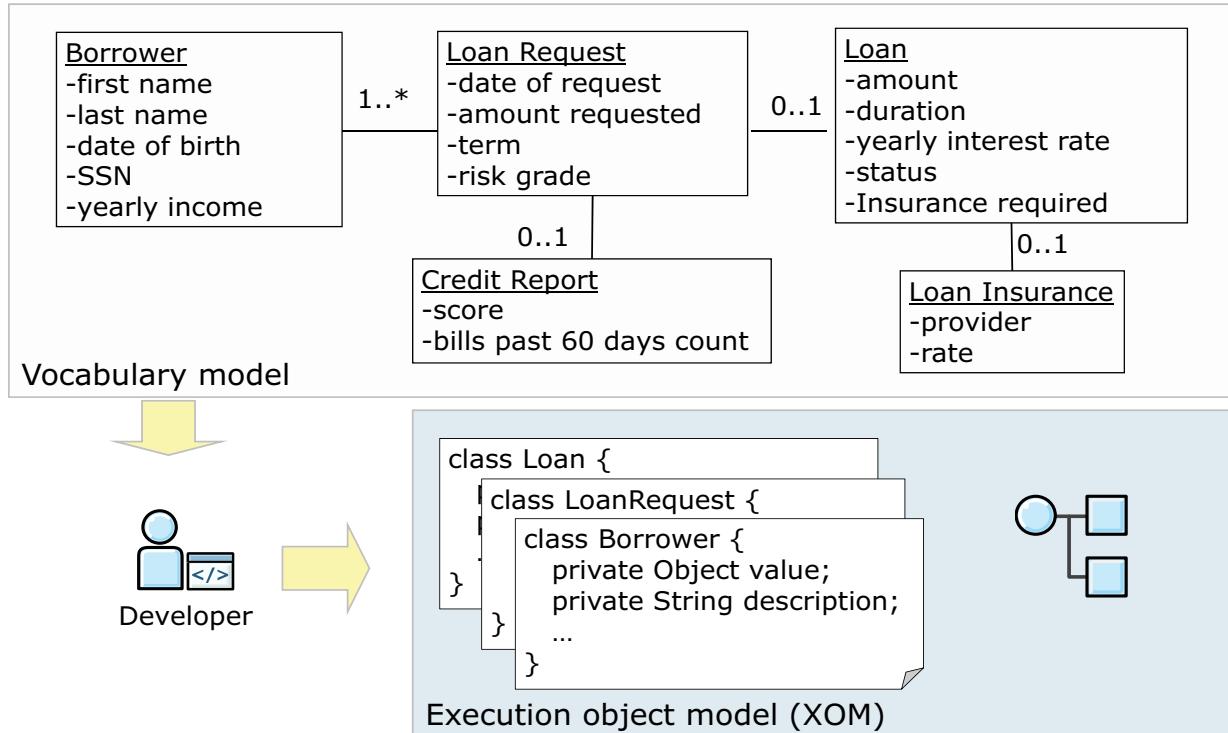
Figure 2-10. Discovery: Identifying vocabulary

Discovering rules for implementation involves not only documenting the business rules, but also identifying the vocabulary that is required to formulate the rules, and then formalizing that vocabulary as a conceptual object model.

For example, if the rules talk about customers, then the object model must include a Customer object. If you have rules about reward programs and customer categories, then the object model must include Reward and Category objects.

Because rule discovery is intertwined with defining the object model, these activities are iterative. The discovery phase can require several interviews between business analysts, policy managers, and developers, so be prepared to meet often for discovery workshops.

## Implementing the model



Developing decision services

© Copyright IBM Corporation 2019

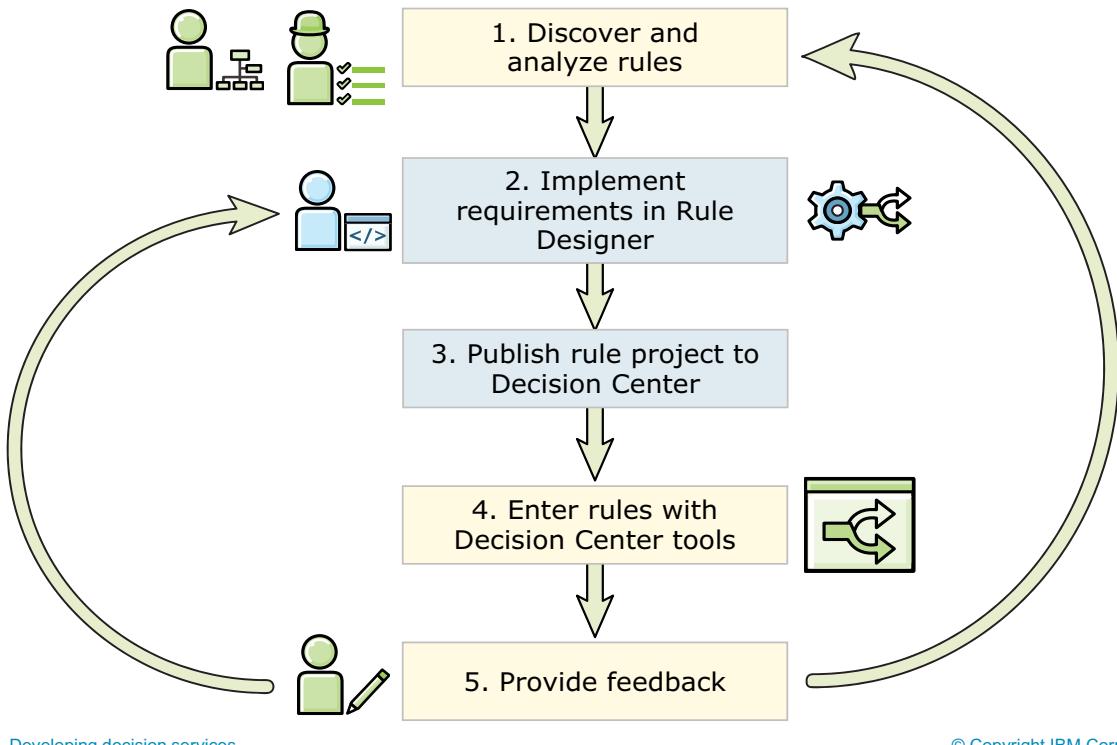
Figure 2-11. Implementing the model

The vocabulary that the business users capture during the discovery and analysis phase becomes the requirements for development of the object models. You can implement these requirements as Java classes or XML data, which becomes the execution object model (XOM).

The XOM is the code that allows the rules to execute. You learn more about the XOM later in this course.



## Using an agile development approach



Developing decision services

© Copyright IBM Corporation 2019

Figure 2-12. Using an agile development approach

Decision management projects require extensive collaboration between business and technical stakeholders in the project. To ensure that the rules were discovered and analyzed correctly, the business users must start entering rules into the tool as soon as possible. This need requires that developers prepare the rule authoring environment early in the development cycle, even before all the rules are discovered.

The following development tasks are illustrated on this slide:

1. Early results from rule discovery, including vocabulary models and initial rules, are passed to the developers as requirements.
2. Based on these requirements and discussion with the business analyst, developers work in Rule Designer to design and set up the rule authoring environment. As a first step, developers prepare a *rule project*, which is used as a container for the newly discovered rules and vocabulary.
3. When the rule project is ready, it is published to Decision Center so that business users can access it through the Decision Center consoles. Both developers and business users can continue to work simultaneously on the same project but in their separate environments on separate tasks.
4. As rule authors begin entering rules in Decision Center, they might find problems or discover new requirements.

5. The rules might look good on paper, but trying to implement them in the tool can uncover possible design or analysis issues.

By using an agile or iterative approach, the business users can provide early feedback on the project implementation, which allows the developers to respond more easily to requirement or model changes. The project might go through several cycles of implementation and feedback loops, but these iterations become less frequent as the project stabilizes.

## 2.2. Designing the business rule application

## Designing the business rule application

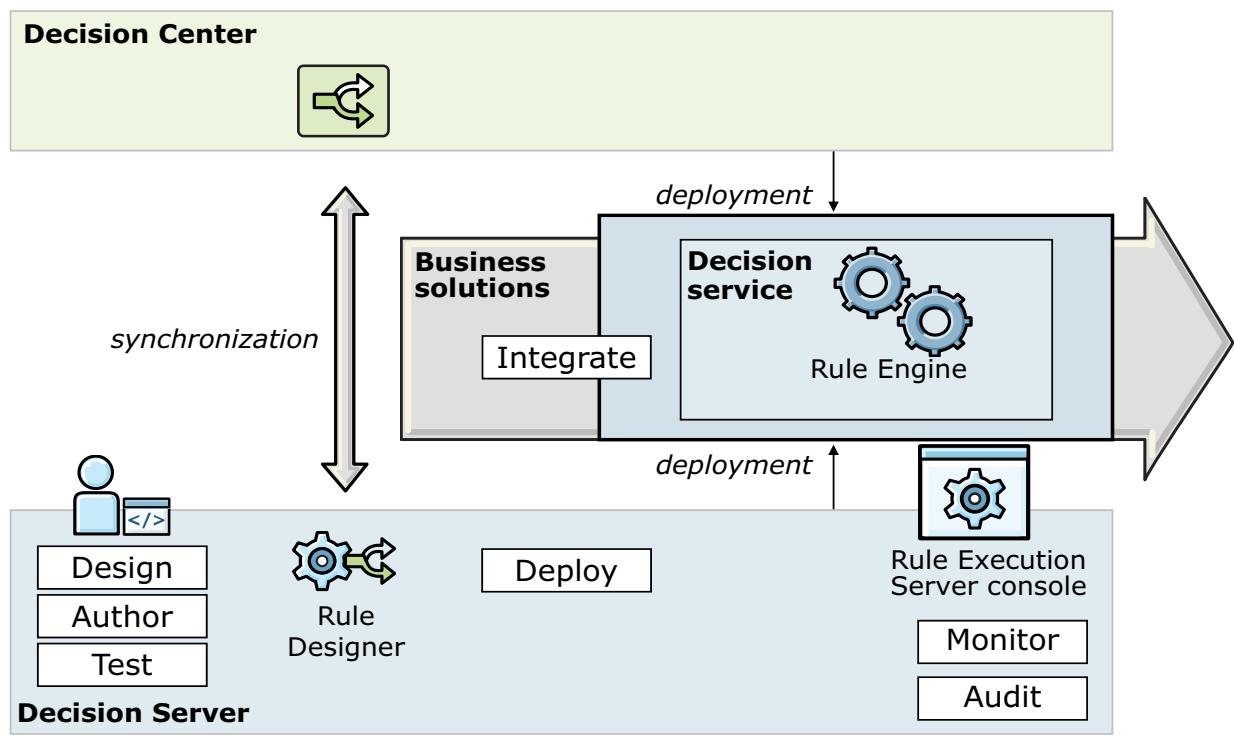
Developing decision services

© Copyright IBM Corporation 2019

*Figure 2-13. Designing the business rule application*



## Overview of developer tasks



Developing decision services

© Copyright IBM Corporation 2019

Figure 2-14. Overview of developer tasks

This slide presents an overview of the various tasks that are involved when you, as a developer, work with business rules. This unit focuses on the first task: **Design**.

Before delving into the details of design, review the following summary of all the other tasks, which are covered in detail during this course.

As a developer on a business rule application development project, you look at the project from these angles:

- Implementation of the business logic as rules
- Implementation of the rule vocabulary in Java or XML
- Implementation of the application to use the business logic

As you saw earlier, business rules are developed and maintained independently from the application code.

The first step in the development of a business rule application is to **design** the rule projects in Rule Designer. Design involves setting up the rule authoring and management infrastructure for business users.

In Rule Designer, you design the granularity of your decision services and the contract between client applications and the decision service. You also design the model and vocabulary for authoring business rules.



**Note**

The remaining tasks are described later in this course.

---

## Rule projects (1 of 2)

- Container for rule authoring artifacts that are required to produce a decision

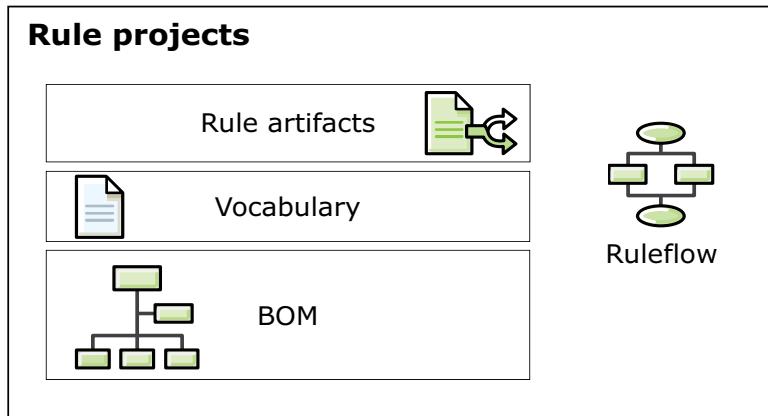


Figure 2-15. Rule projects (1 of 2)

When you start developing a business rule application, you first set up the rule authoring environment infrastructure for business users.

You work in Rule Designer to create the *rule project*, which is the starting point for the business rule application.

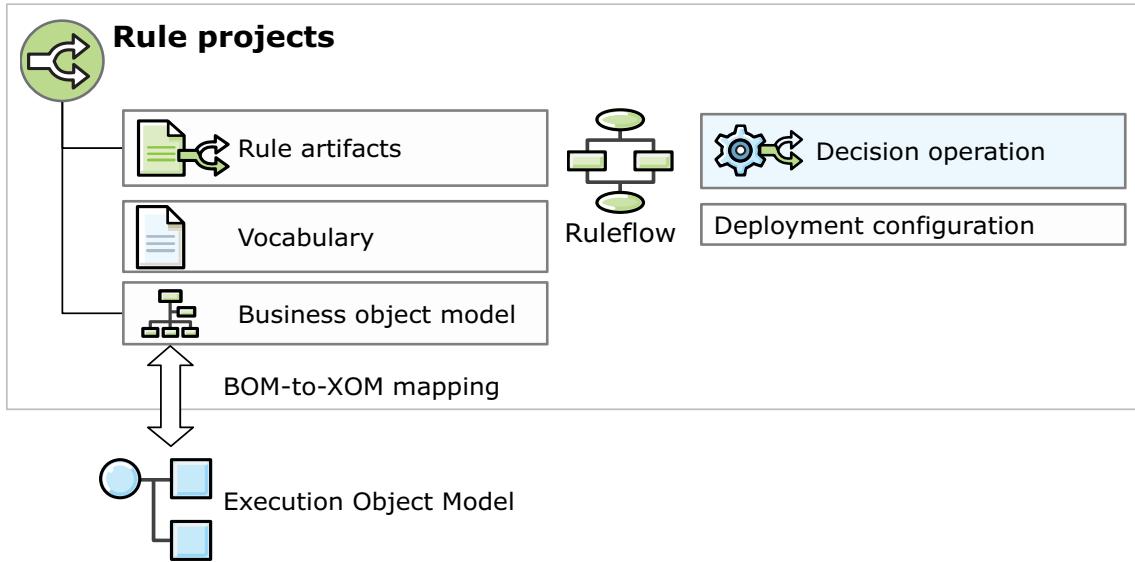
A decision service is a set of related *rule projects*. A rule project is a type of Eclipse project that is dedicated to the development of business rule applications. Decision services incorporate governance features so that all related rule projects are managed and deployed as a single unit.

In a rule project, you define a structure for your rule artifacts and set up the business object model (BOM) and vocabulary for editing the rules.

You also collaborate with the business analysts on the *ruleflow* that outlines the general sequence in which business logic must be evaluated within the ruleset to produce the correct decision. The ruleflow outline can be defined even before the rules are written.

## Rule projects (2 of 2)

- A ruleset is extracted from the finalized rule project



Developing decision services

© Copyright IBM Corporation 2019

Figure 2-16. Rule projects (2 of 2)

The set of rule artifacts that are used together to produce a decision is called a *ruleset*. A ruleset might include all the rules in your rule project or only some. You can also extract a ruleset from multiple rule projects.

Each ruleset within the decision service is defined as a **decision operation** with a unique signature of input and output parameters. You define *parameters* to pass objects from the calling application that the rules are going to be evaluated against. The decision result is returned to the calling application by ruleset parameters.

For your client application to use the ruleset, a contract must exist between the application and the ruleset. In the simplest case, this contract consists of two primary elements:

- **BOM-to-XOM mapping:** A mapping from the BOM to the Java objects or XML schemas of your application environment
  - These objects or schemas are called the execution object model (XOM).
  - The BOM-to-XOM mapping makes the ruleset (written in terms of the BOM) capable of handling the data that your application manipulates in the form of native Java objects or XML.
- **Parameters:** A set of input and output parameters that describe the data that you pass to the ruleset and the data that you expect to receive back as the decision results

- Through the BOM-to-XOM mapping, these data elements are provided to and from your application in the form of the XOM, but the rules reason on them in terms of the vocabulary and BOM.

After the ruleset contract is established, you can deploy your rules for execution. You define a deployment configuration to select which decision operation to deploy and to provide details of the server to which you are deploying.

## 2.3. Setting up decision services

## Setting up decision services

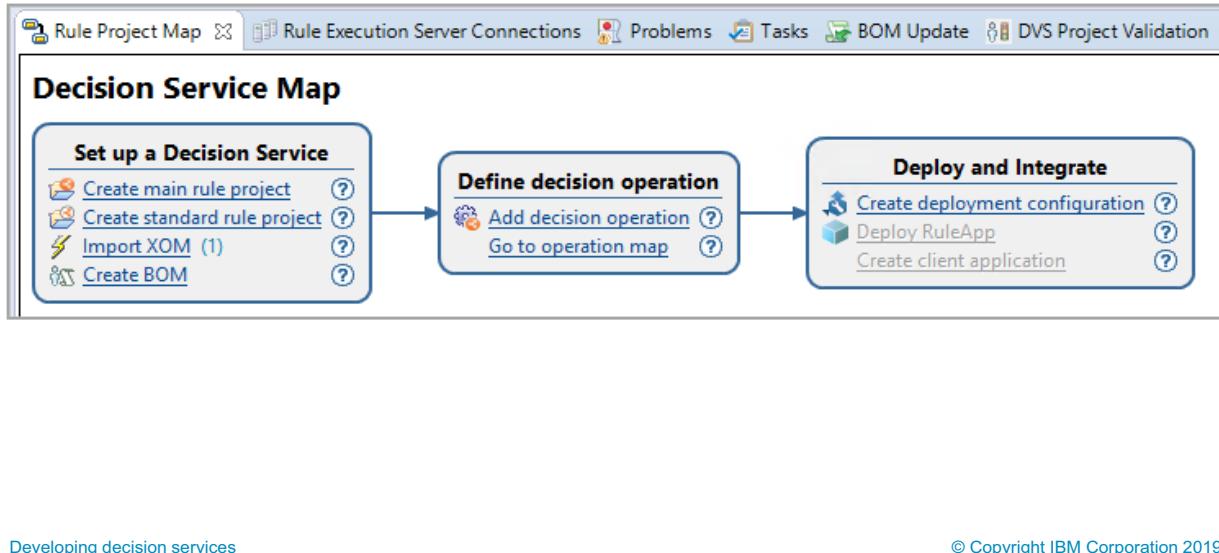
Developing decision services

© Copyright IBM Corporation 2019

*Figure 2-17. Setting up decision services*

## Decision service map

- In Rule Designer, the Decision Service Map outlines the main phases and tasks for setting up a decision service
- Decision Service Map example:



Developing decision services

© Copyright IBM Corporation 2019

Figure 2-18. Decision service map

Rule Designer includes a Decision Service Map that helps set up a decision service. The Decision Service Map is available in the Rule Project Map view in your Rule Designer workspace.

The map identifies the goals, and the tasks in each goal, as a guideline for development:

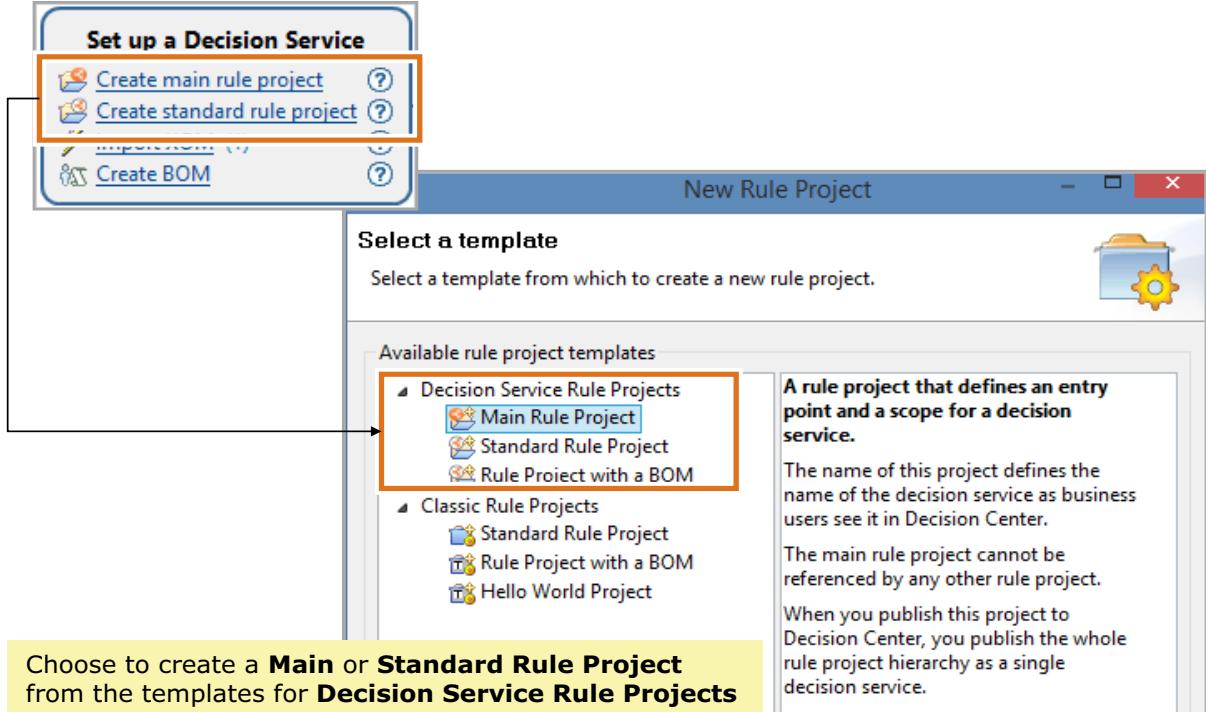
- Goals are represented as rounded *parts*.
- Tasks are represented as *links* in the goals of the map, and correspond to the steps that you saw previously.

The map guides you during development by identifying the sequence of tasks and their status. As you progress through development, links become enabled to help you see which task to do next. When you click a link in the map, Rule Designer opens the relevant dialog box or wizard.

You work with this map during the exercises.



## Create a decision service rule project in Rule Designer



Developing decision services

© Copyright IBM Corporation 2019

Figure 2-19. Create a decision service rule project in Rule Designer

To create a decision service rule project, you can click the links in the Decision Service Map to open the New Rule Project wizard in Rule Designer. You choose from the **Decision Service Rule Projects** templates that are provided.

If you have only one rule project in the decision service, you must define it as the main rule project.

## Decision service rule project templates

- Decision service rule project templates contain these folders:

| Folder            | Contains   |
|-------------------|--|
| <b>rules</b>      | Stores rule artifacts, including action rules, decision tables, decision trees, and ruleflows    |
| <b>bom</b>        | Stores the business object model and the vocabulary that is used in the rules                    |
| <b>deployment</b> | Stores the decision operations and deployment configurations                                     |
| <b>queries</b>    | Stores queries that can be run to find certain rules and apply actions on those rules            |
| <b>resources</b>  | Stores any type of file that is not part of the rule model                                       |
| <b>templates</b>  | Stores templates (partially filled rules) that can serve as a starting point when creating rules |

Figure 2-20. Decision service rule project templates

By default, a standard decision service rule project includes the basic rule project folders, plus the **deployment** folder, which is where you define the decision operations and deployment configurations.

## Design: XOM and BOM

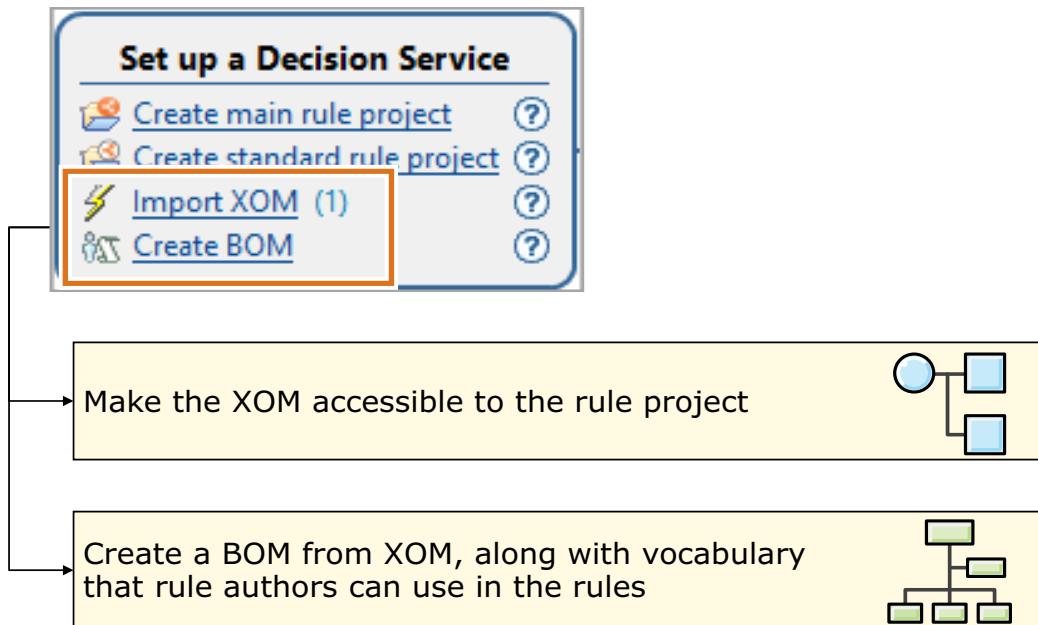


Figure 2-21. Design: XOM and BOM

Based on the rule project map that you saw earlier, the first step in preparing the rule project is **Design**, which includes designing the object models.

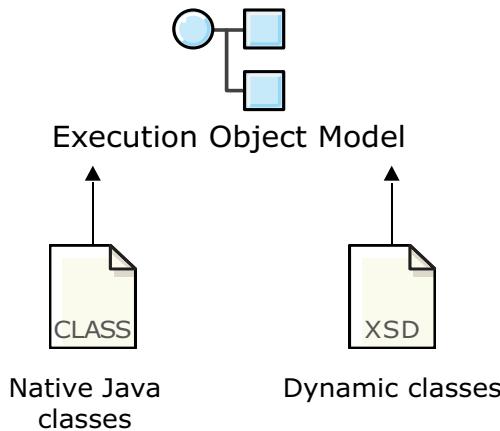
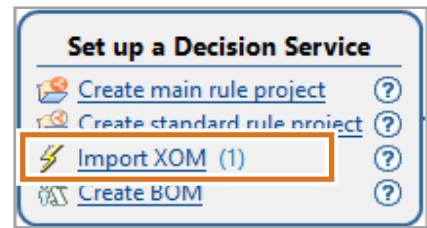
To accurately reflect the business perspective, developers write the code implementation of the object model to create the execution object model (XOM). You can then generate the BOM from the XOM.

The XOM is stored in a separate project; it is not part of the rule project. However, when you create the rule project, you import the XOM into the rule project, which associates that XOM to the rule project.

After the BOM is created, you can define the ruleset parameters. The ruleset parameters define which objects are passed to the rule engine and which objects are returned to the application. These objects must be defined in both the BOM and XOM so that you can create the ruleset parameters.

## Design: XOM

- Execution object model (XOM) makes rule execution possible
- XOM can be written in Java or XML



Developing decision services

© Copyright IBM Corporation 2019

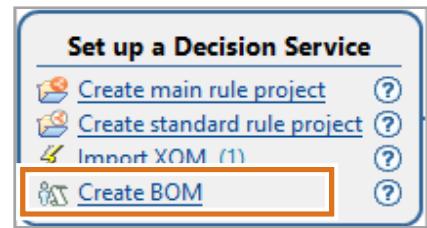
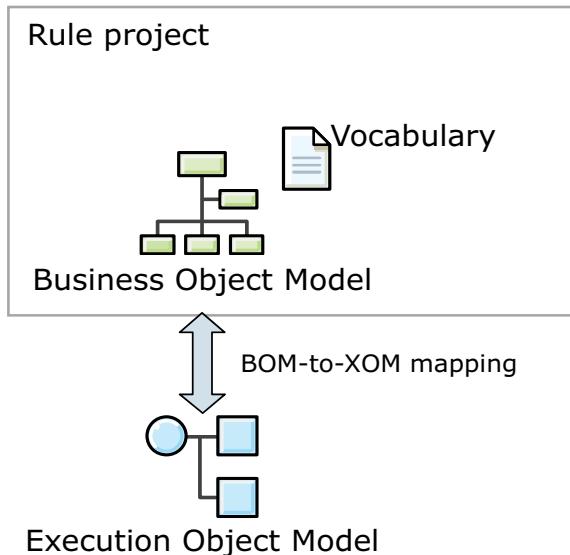
*Figure 2-22. Design: XOM*

The business logic can be implemented by using either Java objects or an XML schema (XSD). As mentioned earlier, the implementation code is called the execution object model (XOM). The XOM makes rule execution possible. While business rules use the vocabulary from the BOM, each business element in a rule must have a corresponding XOM implementation.

Regardless of how you implement the XOM (in Java or XML), you can build the BOM to match the original models and requirements. If the business users provide a clearly defined and complete vocabulary model, the implementation of that model requires fewer iterations of feedback to produce a stable BOM and XOM.

## Design: BOM

- Rule Designer can automatically generate a BOM from a XOM, along with a default vocabulary



Developing decision services

© Copyright IBM Corporation 2019

*Figure 2-23. Design: BOM*

The BOM plays a key role in the rule project because it is the source of vocabulary for the rules.

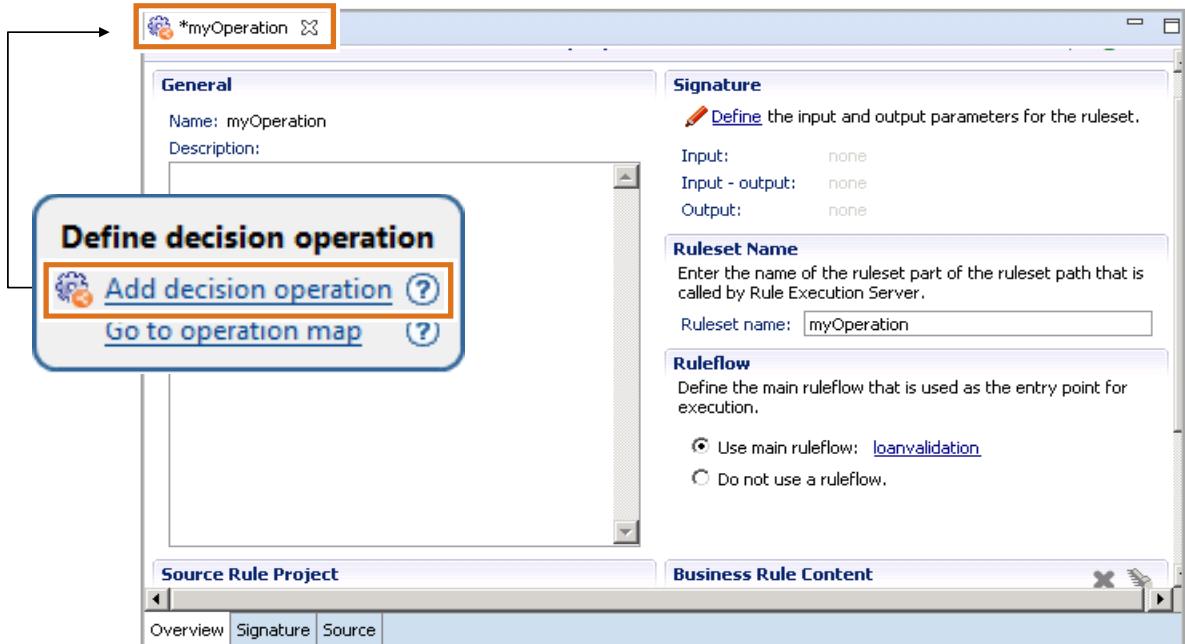
In some cases, you might choose to store the BOM in a separate rule project so that it can be shared across multiple rule projects.

After the XOM is imported into a rule project, you can use Rule Designer to automatically generate a BOM from the XOM. Although the XOM is stored in a separate project, the BOM-to-XOM mapping maintains the association between the BOM and the code.

When setting up the rule project, developers need business user input to accurately define the BOM, the vocabulary, and the data flow as these areas must reflect the business perspective.

## Defining the decision operation

- After setting up the projects, you must define the decision operation



Developing decision services

© Copyright IBM Corporation 2019

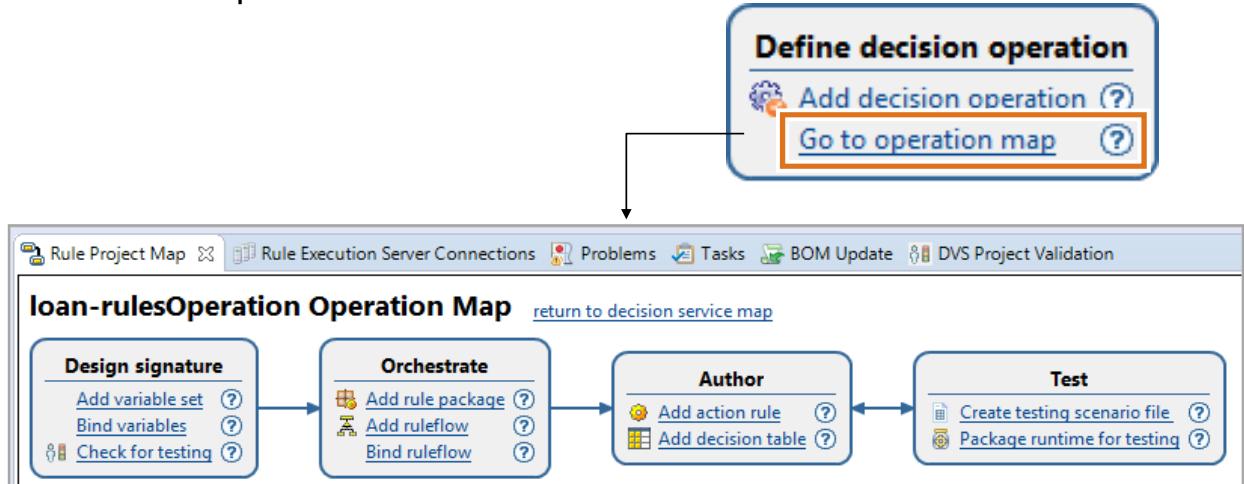
Figure 2-24. Defining the decision operation

After the BOM is created, you can define the decision operation.



## Using the Operation Map

- Use the Operation Map to guide you through the tasks of defining the decision operation



Developing decision services

© Copyright IBM Corporation 2019

*Figure 2-25. Using the Operation Map*

After you define a decision operation, you can define all the artifacts that are required to make a decision, including the rule variables and ruleset parameters for the signature.



## Designing the signature

- Create a variable set
  - Contains all your rule variables for the decision operation

The screenshot shows a software interface for designing a variable set. At the top right is a button labeled 'Design signature' with three options: 'Add variable set', 'Bind variables', and 'Check for testing'. The 'Add variable set' option is highlighted with a red border. Below this is a window titled 'loan-rulesParameters' containing a table of variable set entries:

| Name     | Type                   | Verbalization   |
|----------|------------------------|-----------------|
| borrower | training_loan.Borrower | the borrower    |
| loan     | training_loan.Loan     | the loan        |
| report   | training_loan.Report   | the loan report |

On the right side of the table are three buttons: 'Add', 'Remove', and 'Refactor'.

Figure 2-26. Designing the signature

To pass objects from the calling application to the rule engine, you need ruleset parameters. The ruleset parameters define which objects are passed to the rule engine and which objects are returned to the application. These objects must be defined in both the BOM and XOM.

The rules do not handle the objects directly, but use ruleset variables. Ruleset variables are used to exchange information internally within the ruleset or when you want to use the same variable across several rules. You bind the ruleset variables to ruleset parameters to access the objects from the calling application.

The first task is to create the variables in a *variable set*.



## Designing the signature

- Bind the variables to the ruleset parameters

Decision Operation Signature - loan-rulesOperation

**Design signature**

- [Add variable set](#)
- [Bind variables](#)
- [Check for testing](#)

| <b>Eligible variables</b><br>Select the ruleset variables that you want to use as parameters for the decision operation. Ruleset variables are defined in variable sets.<br><div style="border: 1px solid #ccc; padding: 2px; margin-top: 5px;"> <span style="color: #ccc;">Refresh</span> <span style="color: #ccc;">+ Add as ruleset parameter</span> </div> <div style="border: 1px solid #ccc; padding: 2px; margin-top: 5px;"> <span style="color: #ccc;">loan-rules</span> <ul style="list-style-type: none"> <li><span style="color: #ccc;">loan-rulesParameters</span> <ul style="list-style-type: none"> <li><span style="color: #ccc;">borrower</span></li> <li><span style="color: #ccc;">loan</span></li> <li><span style="color: #ccc;">report</span></li> </ul> </li> <li><span style="color: #ccc;">local var</span></li> </ul> </div> | <b>Input Parameters</b><br>Define the parameters required to call the execution.<br><table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th>Parameter name</th> <th>Verbalization</th> <th>Type</th> <th>Initial Value</th> </tr> </thead> <tbody> <tr> <td> borrower</td> <td>the borrower</td> <td>training_loan.Borrower</td> <td></td> </tr> </tbody> </table><br><b>Input - Output Parameters</b><br>Define the parameters that are required, modified, and then returned by the execution.<br><table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th>Parameter name</th> <th>Verbalization</th> <th>Type</th> <th>Initial Value</th> </tr> </thead> <tbody> <tr> <td> loan</td> <td>the loan</td> <td>training_loan.Loan</td> <td></td> </tr> </tbody> </table><br><b>Output Parameters</b><br>Define the parameters that are initialized and returned by the execution.<br><table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th>Parameter name</th> <th>Verbalization</th> <th>Type</th> <th>Initial Value</th> </tr> </thead> <tbody> <tr> <td> report</td> <td>the loan report</td> <td>training_loan.Report</td> <td></td> </tr> </tbody> </table> | Parameter name         | Verbalization | Type | Initial Value | borrower | the borrower | training_loan.Borrower |  | Parameter name | Verbalization | Type | Initial Value | loan | the loan | training_loan.Loan |  | Parameter name | Verbalization | Type | Initial Value | report | the loan report | training_loan.Report |  |
|---|--|------------------------|---------------|------|---------------|----------|--------------|------------------------|--|----------------|---------------|------|---------------|------|----------|--------------------|--|----------------|---------------|------|---------------|--------|-----------------|----------------------|--|
| Parameter name  | Verbalization  | Type                   | Initial Value |      |               |          |              |                        |  |                |               |      |               |      |          |                    |  |                |               |      |               |        |                 |                      |  |
| borrower  | the borrower   | training_loan.Borrower |               |      |               |          |              |                        |  |                |               |      |               |      |          |                    |  |                |               |      |               |        |                 |                      |  |
| Parameter name  | Verbalization  | Type                   | Initial Value |      |               |          |              |                        |  |                |               |      |               |      |          |                    |  |                |               |      |               |        |                 |                      |  |
| loan  | the loan   | training_loan.Loan     |               |      |               |          |              |                        |  |                |               |      |               |      |          |                    |  |                |               |      |               |        |                 |                      |  |
| Parameter name  | Verbalization  | Type                   | Initial Value |      |               |          |              |                        |  |                |               |      |               |      |          |                    |  |                |               |      |               |        |                 |                      |  |
| report  | the loan report  | training_loan.Report   |               |      |               |          |              |                        |  |                |               |      |               |      |          |                    |  |                |               |      |               |        |                 |                      |  |

Overview
Signature
Source

Developing decision services

© Copyright IBM Corporation 2019

Figure 2-27. Designing the signature

You bind the variables to ruleset parameters on the **Signature** tab of the decision operation.

To define ruleset parameters, you must collaborate with the business analysts to understand what information is required for a decision and what type of information to include as part of the decision results.

The ruleset parameters define the interface between the ruleset and the client application.

- The input parameters define the data available for processing by the rule engine.
- The output parameters reflect the business decision that the rule engine returns.

Parameters can also be defined as input/output.

## 2.4. Project properties

## Project properties

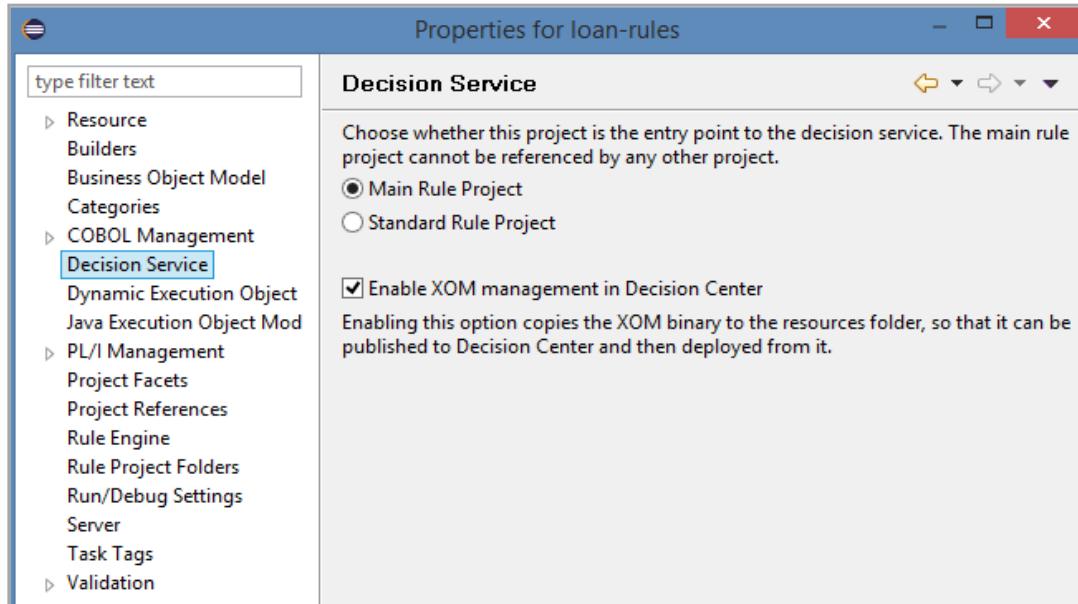
Developing decision services

© Copyright IBM Corporation 2019

*Figure 2-28. Project properties*

## Properties: Project hierarchy

- Right-click rule project to open the Properties dialog box
  - Example: Define a project as the main rule project for your decision service



Developing decision services

© Copyright IBM Corporation 2019

*Figure 2-29. Properties: Project hierarchy*

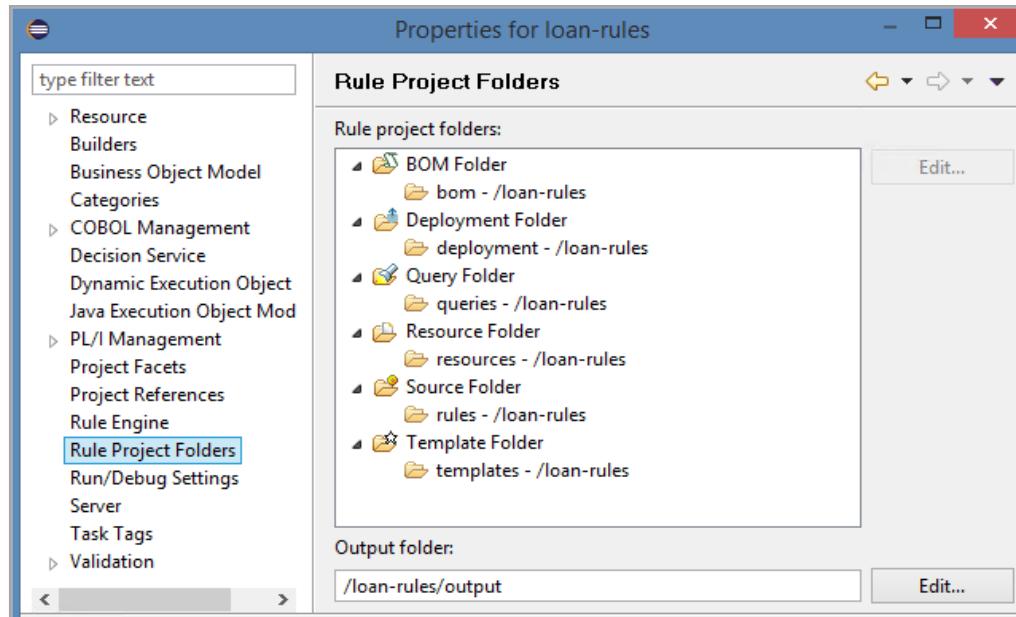
You can define specific properties on the rule project itself. Project properties are defined in the Properties dialog box, which you can open by right-clicking your project and selecting **Properties**.

For example, in this screen capture, you see the **Decision Service** property selected, and the option of whether to keep the project as **Main Rule Project** or demote it to **Standard Rule Project**.



## Properties: Folders

- Click **Rule Project Folders** to view or edit folder names and their location



Developing decision services

© Copyright IBM Corporation 2019

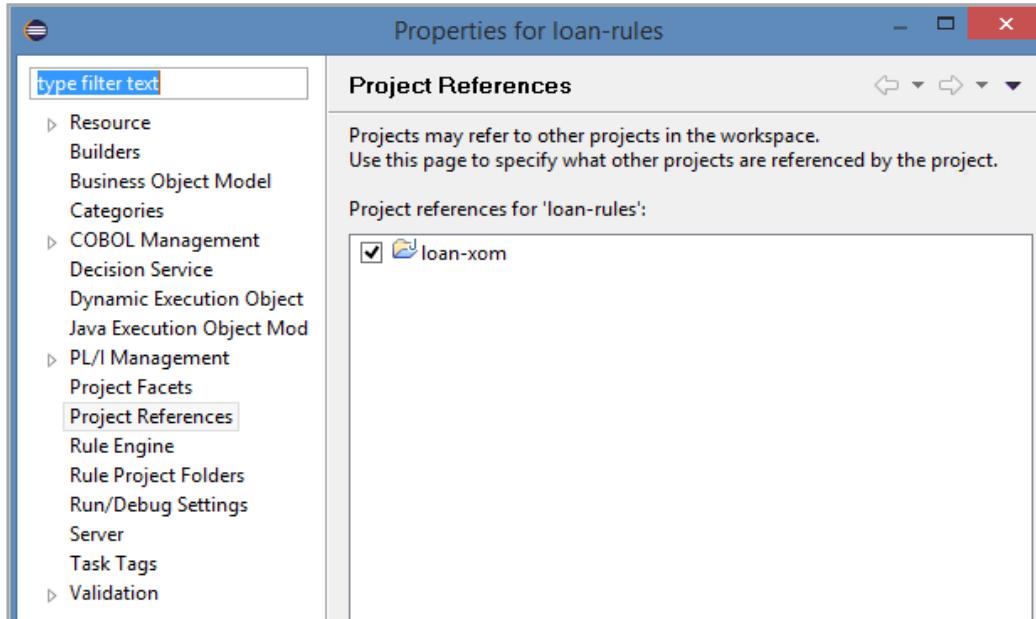
*Figure 2-30. Properties: Folders*

In this screen capture, you see the **Rule Project Folders** property open, and the list and location of folders that are available in the project.



## Properties: Project references

- Specify project references to define dependencies between projects



Developing decision services

© Copyright IBM Corporation 2019

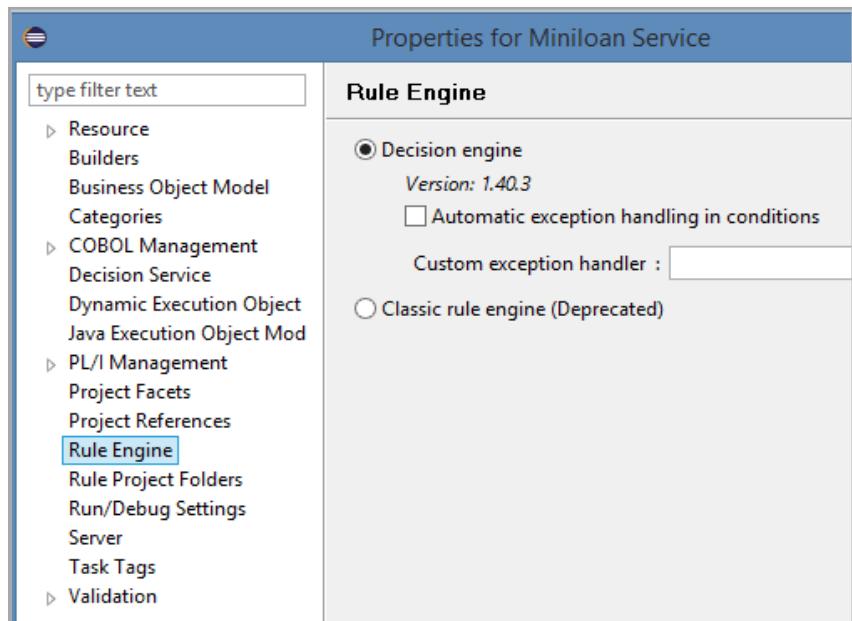
Figure 2-31. Properties: Project references

This screen capture shows the **Project References** property selected.

Project references define which other projects in your workspace are referenced by this project.

## Properties: Rule engine type

- Specify the type of rule engine to use to execute the rules in the project
  - Decision engine
  - Classic rule engine



Developing decision services

© Copyright IBM Corporation 2019

*Figure 2-32. Properties: Rule engine type*

The Decision engine is the default rule engine. You learn more about the rule engine later in this course.

## 2.5. Using modular project organization

## Using modular project organization

Developing decision services

© Copyright IBM Corporation 2019

*Figure 2-33. Using modular project organization*

## Modular structure (1 of 4)

- Projects can reference other projects
  - Facilitates dependencies between your rules and data
  - Facilitate the assignment of permissions in Decision Center

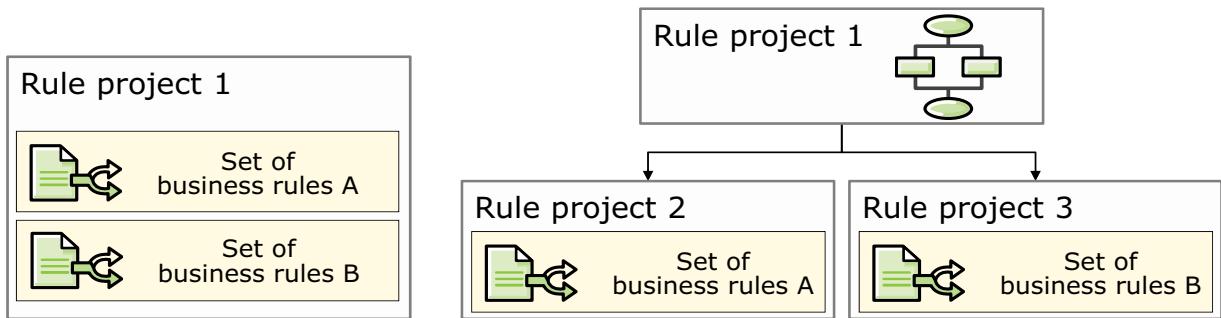


Figure 2-34. Modular structure (1 of 4)

As you saw in the project properties, you can define project references between projects, just as you can with Java projects. Rules in one project can depend on rules or other artifacts in a separate project.

This modularity facilitates dependencies between your rules and data. For example, you can have your BOM in one project and your rules spread across multiple projects.

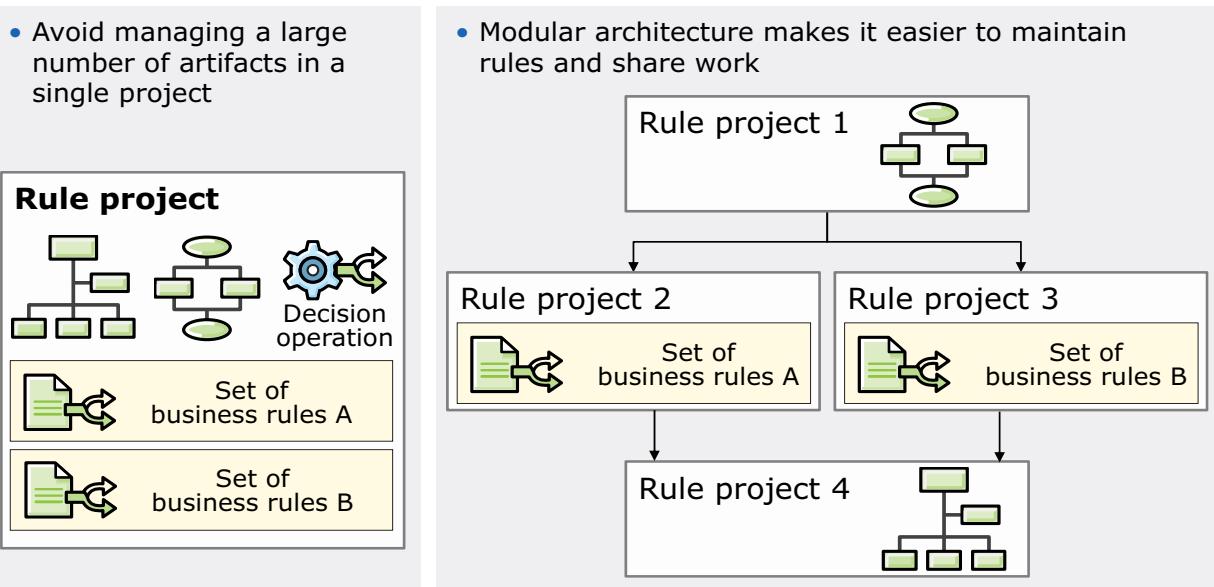
Rule projects can reference each other,

IBM Training

IBM

## Modular structure (2 of 4)

- To improve performance for large rule applications, use modular architecture



Developing decision services

© Copyright IBM Corporation 2019

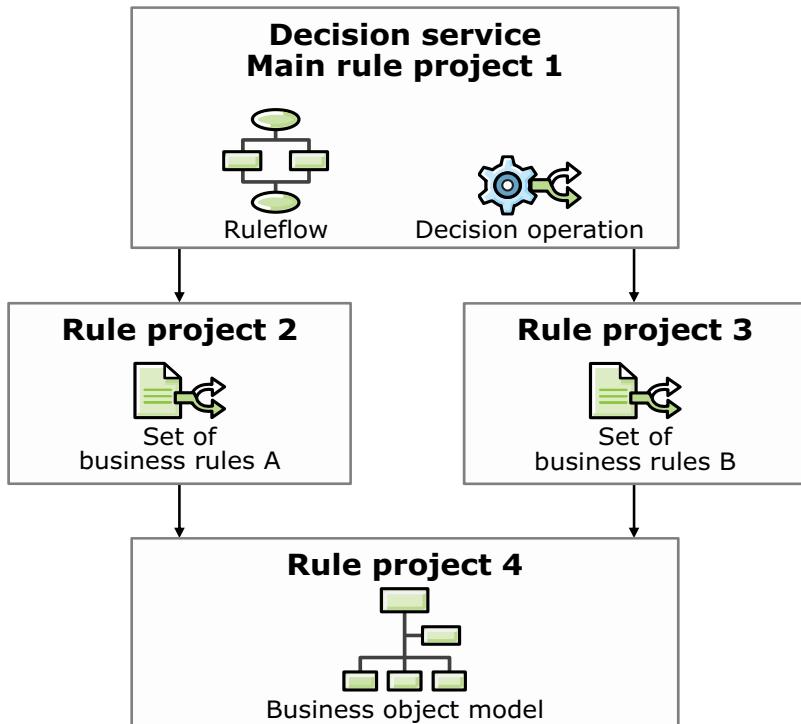
Figure 2-35. Modular structure (2 of 4)

The design of a decision service is to use multiple projects to handle the rules. This approach accommodates project growth because the number of rules in a project will increase significantly over time.

This type of modular structure also facilitates rule maintenance and sharing of work across multiple users.

## Modular structure (3 of 4)

- The decision service hierarchy is based on the principles of modular structure
  - Use Main Rule Project as top-level project in hierarchy
  - Use Standard Rule Project for child projects



Developing decision services

© Copyright IBM Corporation 2019

*Figure 2-36. Modular structure (3 of 4)*

The main rule project is the top-level project in the decision service hierarchy.

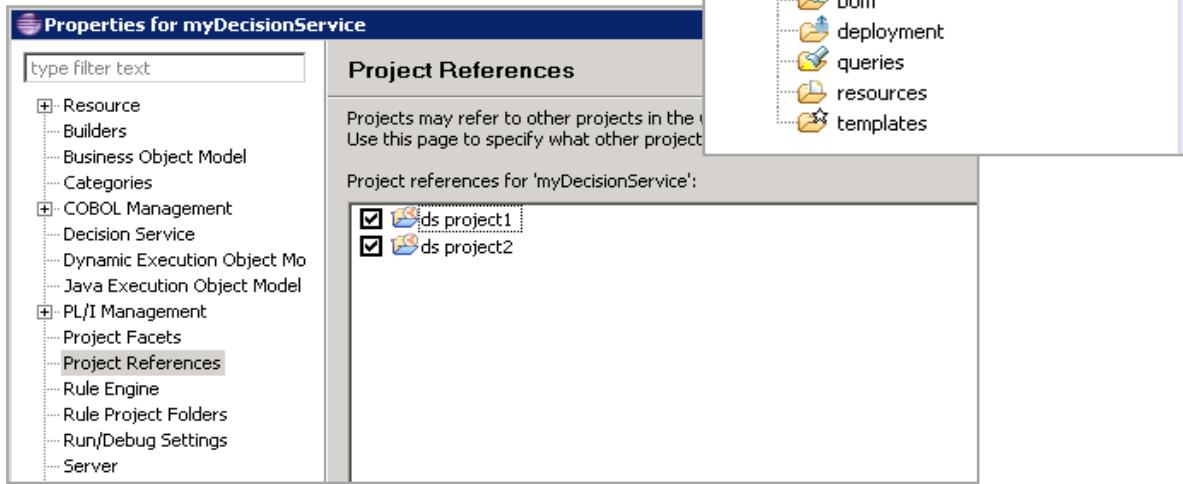
As you see depicted here, the rules are spread across Rule project 2 and Rule project 3. The BOM is in Rule project 4. The decision operation is defined in the main rule project and it can be extracted from one or all of the rule projects that are referenced by the main rule project.

By defining project dependencies between the decision service projects, the business users can manage the lifecycle of all projects within the decision service. This hierarchy simplifies management of the decision service lifecycle. For example, when you synchronize a decision service with Decision Center, all dependent projects are automatically included. When business users open a decision service in the Business console, depending on permissions, they can see any of the projects included in the decision service.



## Modular structure (4 of 4)

- A main decision service rule project is distinguished from other projects with an icon
- Define project references to other projects in the Properties dialog box



Developing decision services

© Copyright IBM Corporation 2019

Figure 2-37. Modular structure (4 of 4)

In the Rule Explorer view of Rule Designer, you can identify the main decision service rule project by the decision service icon.

If the decision service includes other rule projects, they are referenced by the main project, and you can find those references in the Project References property for the main rule project.

## 2.6. Sharing and synchronizing decision services

## Sharing and synchronizing decision services

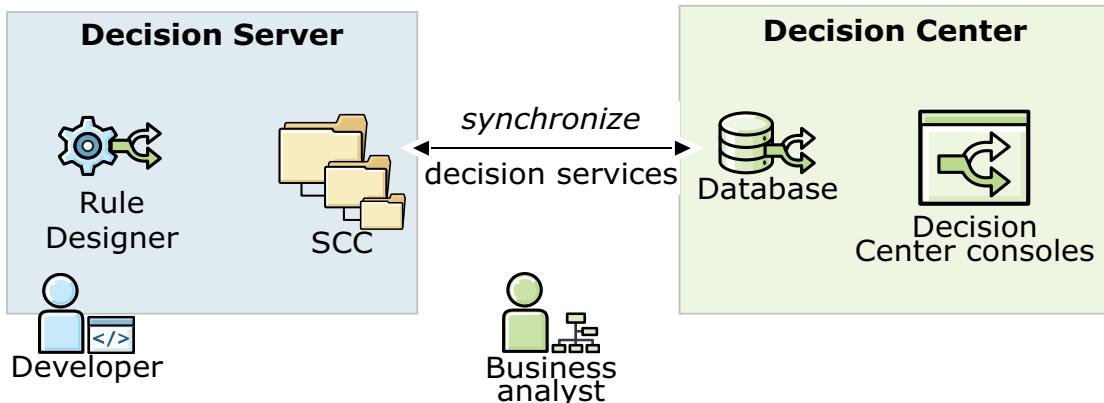
Developing decision services

© Copyright IBM Corporation 2019

*Figure 2-38. Sharing and synchronizing decision services*

## Synchronization between users

- Synchronization across business and development environments
  - Initiated and controlled from Rule Designer by technical BA or developers
  - For every project element, compares the versions that are stored in the Decision Center repository with the copy that is stored through Designer



Developing decision services

© Copyright IBM Corporation 2019

Figure 2-39. Synchronization between users

Synchronization across business and development environments is initiated and controlled from Rule Designer.

The synchronization mechanism compares the versions of each element of a rule project that is stored in the Decision Center repository with the copy in the Rule Designer workspace.

Synchronization is the key to collaborative work between business and IT users who work on the same projects but in separate environments.

For collaborative work, the Decision Center repository must be synchronized with the development tools in Rule Designer.

## Synchronization tools

- Publish an existing rule project from Rule Designer to Decision Center
- Create a project in Rule Designer from an existing Decision Center project
- Synchronize the Rule Designer and Decision Center copies of the project to account for changes on either side
- Resolve conflicts
  - When the same artifact is modified in both environments, compare differences and choose which version to keep

Figure 2-40. *Synchronization tools*

Synchronization tools include the ability to publish an existing rule project from Rule Designer to Decision Center, or create a rule project in Rule Designer from an existing Decision Center project, or synchronize the Rule Designer and Decision Center copies of the project to account for changes on either side.

Synchronization is required whenever one group (business or IT) makes an update that the other group must incorporate into their work. For example, when business users who are working in Decision Center identify changes or fixes that must be made to the vocabulary, their copy of the project must be synchronized back to Rule Designer. Developers can then modify the BOM and publish the project back to Decision Center.

## Synchronization architecture

- Synchronization uses three-way comparison of:
  - Project in local workspace of Rule Designer
  - Remote project in Decision Center repository
  - Reference that computes the state of the synchronization
- Reference state is created as a connection entry file in workspace when you connect to Decision Center:
  - Connection entries file: .syncEntries
- Three-way comparison creates a checksum on both remote and local rules, and then compares them to the reference state

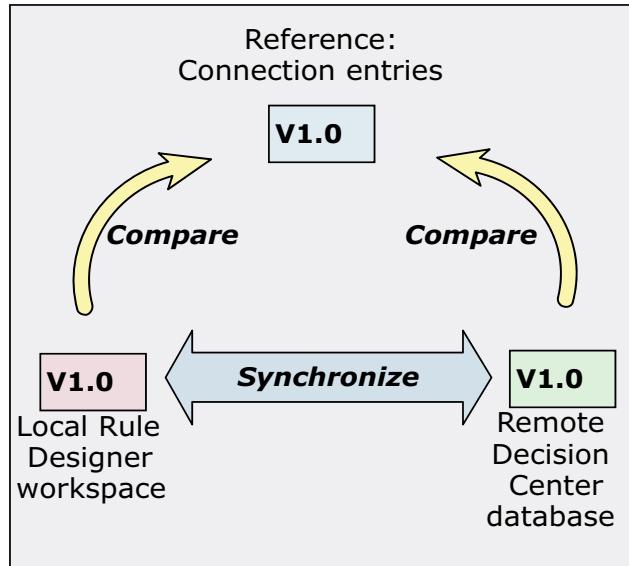


Figure 2-41. Synchronization architecture

The synchronization architecture uses a three-way comparison of the rule project in your local workspace in Rule Designer, the remote project in Decision Center, and the reference that computes the state of the synchronization.

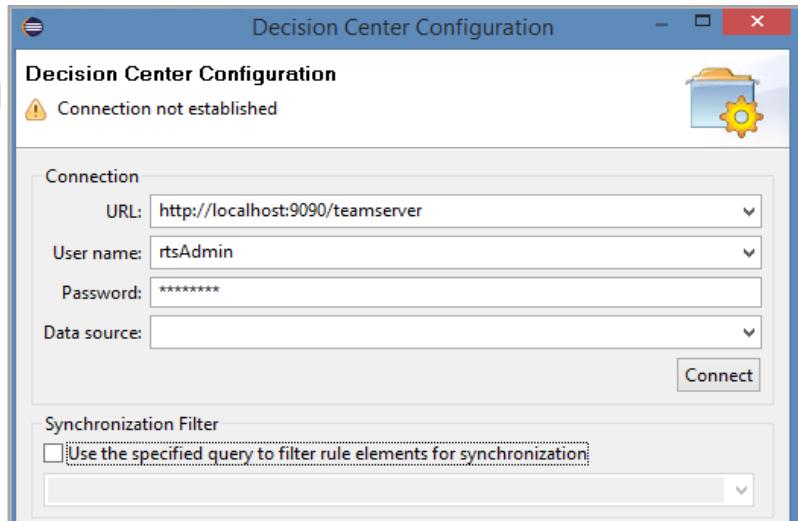
The reference state is created as a connection entry file, called `.syncEntries`, when you connect to Decision Center.

The three-way comparison creates a checksum on both the remote and the local rules and compares them to the reference state.

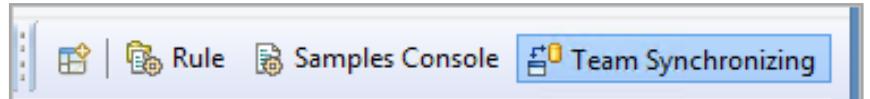


## Connection entries

- Synchronization is initiated and controlled from Rule Designer
- When you connect to Decision Center, the .syncEntries file is created



- During synchronization, Rule Designer opens the Team Synchronizing perspective



Developing decision services

© Copyright IBM Corporation 2019

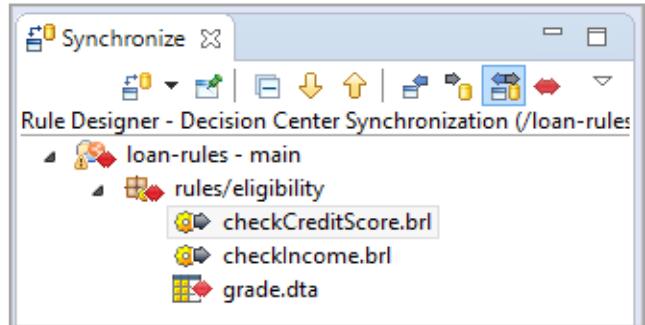
*Figure 2-42. Connection entries*

Here you see the connection dialog that opens when you want to synchronize between Decision Center and Rule Designer. The information that is entered here is stored in the .syncEntries file.

If there is a conflict during synchronization, Rule Designer opens to the Team Synchronizing perspective.

## Choosing how to synchronize

- Synchronize view lists changes and direction:
  - Outgoing
  - Incoming
  - Conflict



- Text Compare view
  - Detailed differences so you can determine how to update each artifact

```

<?xml version="1.0" encoding="UTF-8"?>
<model.brl:ActionRule xmi:version="2.0" xmlns="http://www.ilog.com/xsd/2009/01/brl">
  <name>checkCreditScore</name>
  <uuid>afb53b16-eb8e-4d39-a1d3-da3342e07328</uuid>
  <locale>en_US</locale>
  <!---!-->
  <!---!-->
  set 'minimum score' to 200 ;

```

Developing decision services

© Copyright IBM Corporation 2019

Figure 2-43. Choosing how to synchronize

In the Team Synchronizing perspective, the **Synchronize** view lists the conflicts and the direction of the change:

- Outgoing, which means a change was made in Rule Designer copy and needs to be published to Decision Center
- Incoming, which means a change was made in Decision Center and needs to overwrite the Rule Designer copy
- Conflict, which means a change was made in both the local and remote versions

You can use the **Text Compare** view to see detailed differences between the artifacts and determine how to respond.

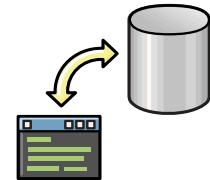
## Synchronization commands

- **Publish** sends artifacts from Rule Designer to Decision Center
- **Update** receives artifacts from Decision Center into Rule Designer
- **Override and Publish** resolves conflict by replacing the Decision Center artifact with the Rule Designer version
- **Override and update** resolves conflict by replacing the Rule Designer artifact with the Decision Center version

Figure 2-44. Synchronization commands

## Choosing the master source

- Establish which source to use as the master, either:
  - Decision Center repository
  - Source code control (SCC)
- Business-user-centric approach:
  - Decision Center repository is considered master
  - Repository provides rule management features, such as version control, baselines, and access control
  - Copies that are stored through Rule Designer are considered temporary
  - Decision Center repository does not contain the artifacts that are required for rule execution, such as XOMs, .jar files, and libraries
- Developer-centric approach:
  - Projects that are stored and managed through SCC
  - A technical user synchronizes the source with Decision Center repository
  - Execution-related artifacts are stored with projects



[Developing decision services](#)

© Copyright IBM Corporation 2019

Figure 2-45. Choosing the master source

To avoid conflicts, you need to choose the master source: Decision Center or source code control (SCC). This decision must be agreed on by both business and development teams.

If Decision Center repository is to be the master source, then business users are in control of the copy of record. If the source code control system is to be the master source, then you need to make sure that copy is regularly synchronized with Decision Center to ensure that business users are working with the latest copy.

You can assign one dedicated technical user to synchronize the two repositories.

## Unit summary

- Identify the development tasks in building a decision management application
- Describe how to set up a decision service in Rule Designer
- Share and synchronize decision services between the business and development environments

Figure 2-46. Unit summary

## Review questions

1. True or False: Agile Business Rule Development involves collaboration between development and business teams during the final phases of a project.
2. True or False: Decision services use a modular approach to organizing rule projects.
3. True or False: To set up a decision service, you use the Decision Service Map to guide you through the tasks.
4. True or False: Synchronization between Rule Designer and Decision Center is controlled from Decision Center.



Figure 2-47. Review questions

Write your answers here:

- 1.
- 2.
- 3.
- 4.

## Review answers

1. True or False: Agile Business Rule Development involves collaboration between development and business teams during the final phases of a project.  
**The answer is False.** Agile Business Rule Development involves collaboration between development and business teams throughout project development, and especially during the early phases.
2. True or False: Decision services use a modular approach to organizing rule projects.  
**The answer is True.**
3. True or False: To set up a decision service, you use the Decision Service Map to guide you through the tasks.  
**The answer is True.**
4. True or False: Synchronization between Rule Designer and Decision Center is controlled from Decision Center.  
**The answer is False.** Synchronization between Rule Designer and Decision Center is controlled from Rule Designer.



Developing decision services

© Copyright IBM Corporation 2019

Figure 2-48. Review answers

## Exercise: Setting up decision services

Developing decision services

© Copyright IBM Corporation 2019

*Figure 2-49. Exercise: Setting up decision services*

## Exercise introduction

- Create main and standard decision service projects
- Set up the decision service to reference the execution object model (XOM)
- Generate a business object model (BOM) and a default vocabulary
- Create a decision operation
- Define ruleset variables and ruleset parameters
- Create rule packages
- Synchronize decision services with Decision Center



Figure 2-50. Exercise introduction

# Unit 3. Modeling decisions

## Estimated time

00:30

## Overview

This unit introduces decision modeling in Decision Center and IBM Decision Composer.

## How you will check your progress

- Review
- Exercise

## Unit objectives

- Explain when to use a decision model service
- Describe how to model decisions

Figure 3-1. Unit objectives

This unit introduces decision modeling in Decision Center and IBM Decision Composer.

After completing this unit, you should be able to explain when to use a decision model service and describe how to model decisions.

## Topics

- Introducing decision modeling in Decision Center
- Creating decision models
- Modeling with IBM Decision Composer

*Figure 3-2. Topics*

## 3.1. Introducing decision modeling in Decision Center

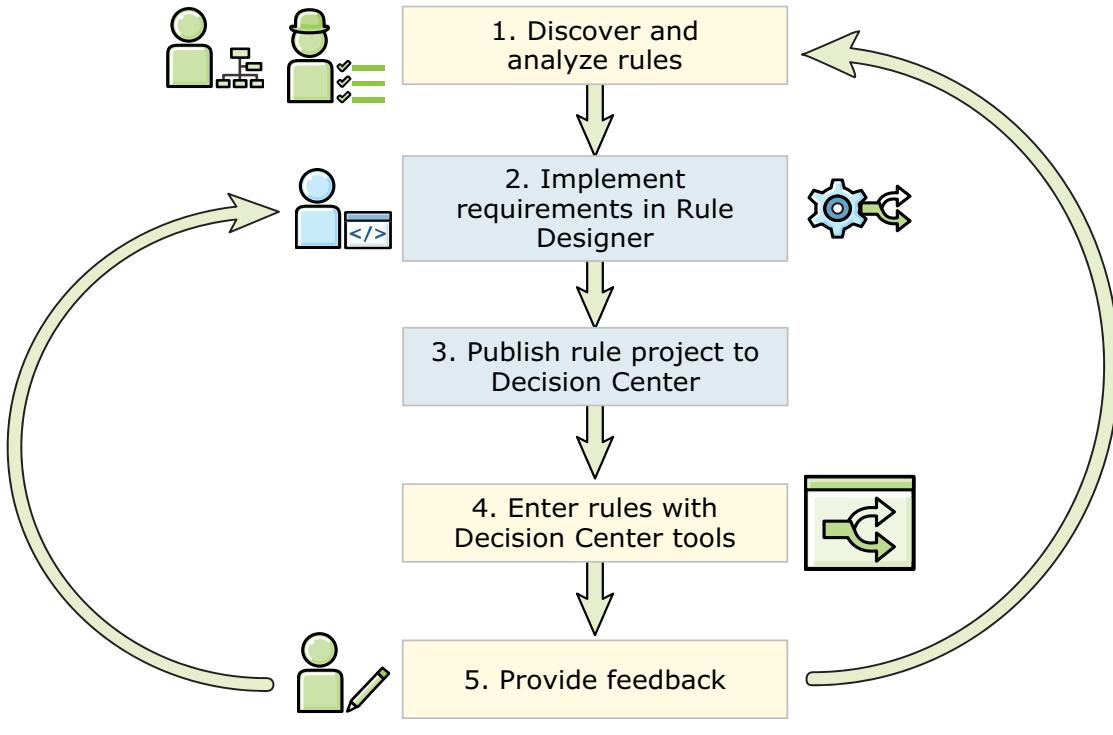
## Introducing decision modeling in Decision Center

Modeling decisions

© Copyright IBM Corporation 2019

*Figure 3-3. Introducing decision modeling in Decision Center*

## Recall: Using an agile development approach



Modeling decisions

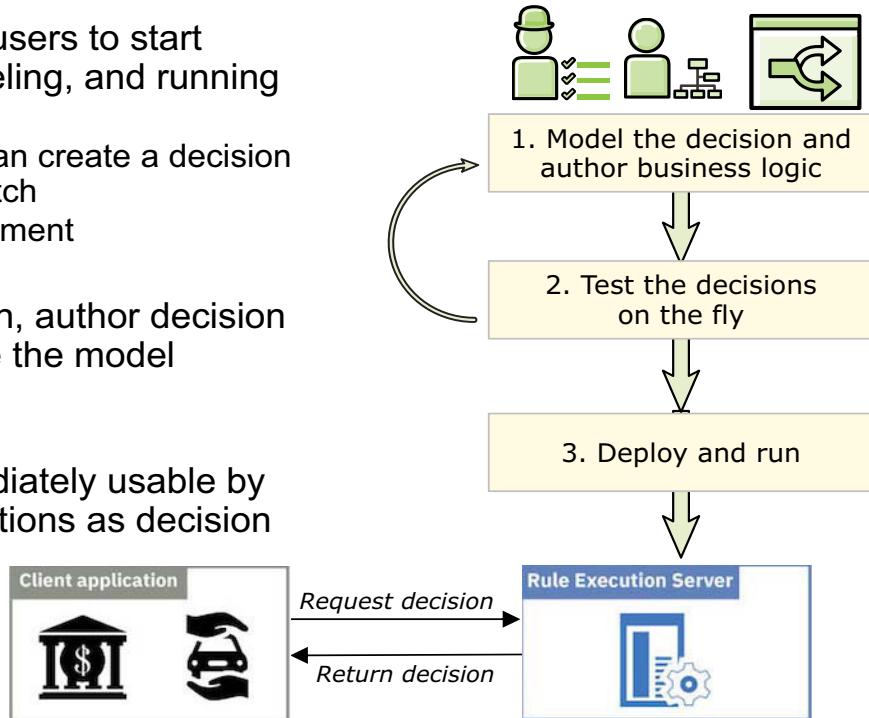
© Copyright IBM Corporation 2019

*Figure 3-4. Recall: Using an agile development approach*

As you learned in the previous unit, development usually starts with the developers and business users working together to discover the rules and vocabulary, and implement them in the tool. Business users define models as requirements, by using UML or DMN modeling language. Developers use those models to set up the rule authoring environment and make it available to the business users. As the project progresses, the developers make sure to keep the business and development environments synchronized.

## Modeling decisions in Business console

- Enable business users to start discovering, modeling, and running decisions
  - Business users can create a decision service from scratch
  - Bypass IT involvement
- Model the decision, author decision logic, and validate the model simultaneously
- Models are immediately usable by enterprise applications as decision services



Modeling decisions

© Copyright IBM Corporation 2019

Figure 3-5. Modeling decisions in Business console

Another way to ensure that the decision logic is captured correctly from the business users, is for the business users to build it themselves. Working “top-down” business users build a decision model that captures the required data and business logic. And in Decision Center Business console, that model can actually be used as the decision service.

Both IBM Decision Composer and Decision Center Business console provide modeling tools so that business users can bypass developer involvement to not only build and test their model, but also deploy it as a decision service. The decision service can run against real data in a production environment.

## Choosing the top-down decision model approach

- Business users can create decision services autonomously
  - End-to-end development by business users
  - Create, test, deploy, and execute decision model service against real data
  - Test, simulate, govern, maintain decision model in Business console
- Use to understand the underlying model of a decision service
  - Template for developers in preparation for larger Operational Decision Manager projects
- Use for simple decisions
  - Not suitable for complex decisions

*Figure 3-6. Choosing the top-down decision model approach*

During this course, you focus on the bottom-up approach, which requires developer involvement. However, the main reason for choosing a decision model service is so that the business experts can be autonomous in the end-to-end creation and deployment of a decision service. Decision models can be edited directly in a relatively non-technical language. And the models can be fully tested and adjusted before even thinking about deployment.

Another reason to choose the decision model service approach when you want to understand the underlying model of a decision service, in preparation for larger Operational Decision Manager projects. Developers can use it as a **template** to build a regular decision service in Rule Designer.

The decision model service is a simplified version of a decision service, and should be used for simple decisions only. If you use the decision model as a template, then you can work in Rule Designer to produce a more flexible and robust version of what the business users captured.

## Decision model services versus decision services

- No BOM or ruleflow
  - The model uses a diagram to defines links between decisions and data nodes
  - No Java or XML XOM is required for the model to be executable
- No synchronization with Rule Designer
  - Decision model services cannot be used in Rule Designer
- The decision model works as one versionable element
  - Any changes to the diagram or to individual rules are saved as a separate version of the decision model
- Cannot be modified simultaneously by multiple users

Figure 3-7. Decision model services versus decision services

This slide lists some of the ways in which a decision model service differs from a decision service. For more information, see the IBM Knowledge Center.

[www.ibm.com/support/knowledgecenter/SSQP76\\_8.10.x/com.ibm.odm.dcenter.model/topics/con\\_dc\\_mod\\_recommend.html](http://www.ibm.com/support/knowledgecenter/SSQP76_8.10.x/com.ibm.odm.dcenter.model/topics/con_dc_mod_recommend.html)

## 3.2. Creating decision models

## Creating decision models

Modeling decisions

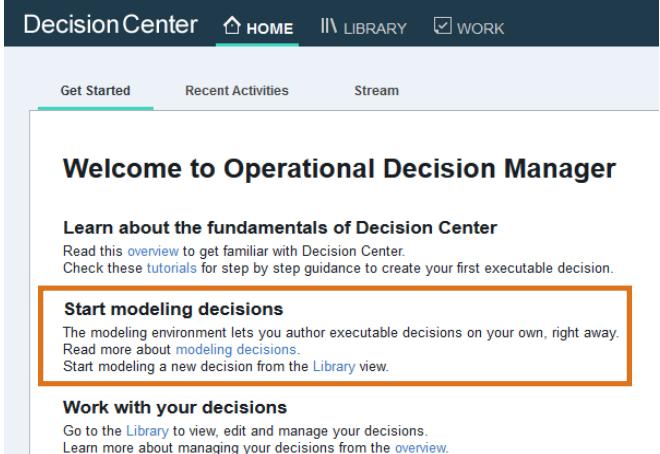
© Copyright IBM Corporation 2019

*Figure 3-8. Creating decision models*

IBM Training 

## Modeling in Business console

- See the **Start modeling decisions** section on the **Home** tab in Business console



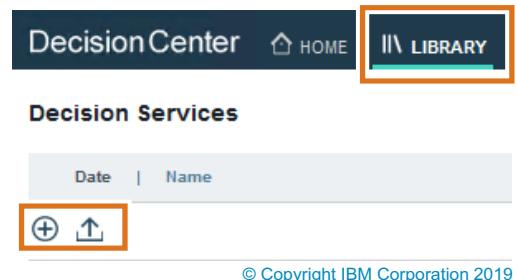
Welcome to Operational Decision Manager

**Learn about the fundamentals of Decision Center**  
Read this [overview](#) to get familiar with Decision Center.  
Check these [tutorials](#) for step by step guidance to create your first executable decision.

**Start modeling decisions**  
The modeling environment lets you author executable decisions on your own, right away.  
Read more about [modeling decisions](#).  
Start modeling a new decision from the [Library view](#).

**Work with your decisions**  
Go to the [Library](#) to view, edit and manage your decisions.  
Learn more about managing your decisions from the [overview](#).

- From the **Library** tab, access or import existing models, or create from scratch



Decision Center  

### Decision Services

Date | Name



© Copyright IBM Corporation 2019

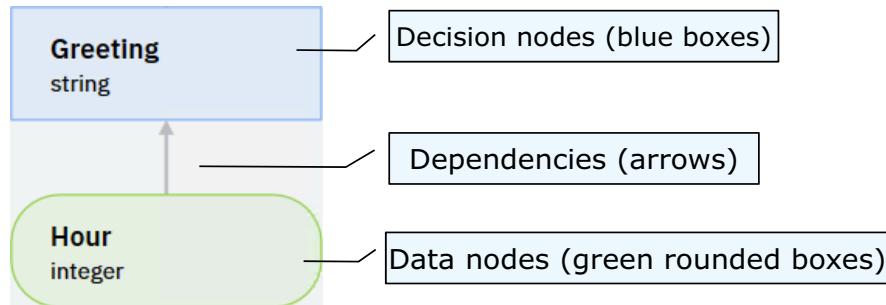
Figure 3-9. Modeling in Business console

For this course, you work with decision models in Business console.

The home page of the Business console includes a brief section on modeling decisions and links to the Library tab. From the Library, you can access your decision models, create new ones, or import models that you worked on in Decision Composer.

## Creating the decision model diagram (1 of 2)

- To model a decision, you:
  - Model the node structure in a diagram
  - Author the decision logic
- Use these notation standards to model the node structure:



- Modeling notation is inspired from DMN ([www.omg.org/dmn](http://www.omg.org/dmn))

*Figure 3-10. Creating the decision model diagram (1 of 2)*

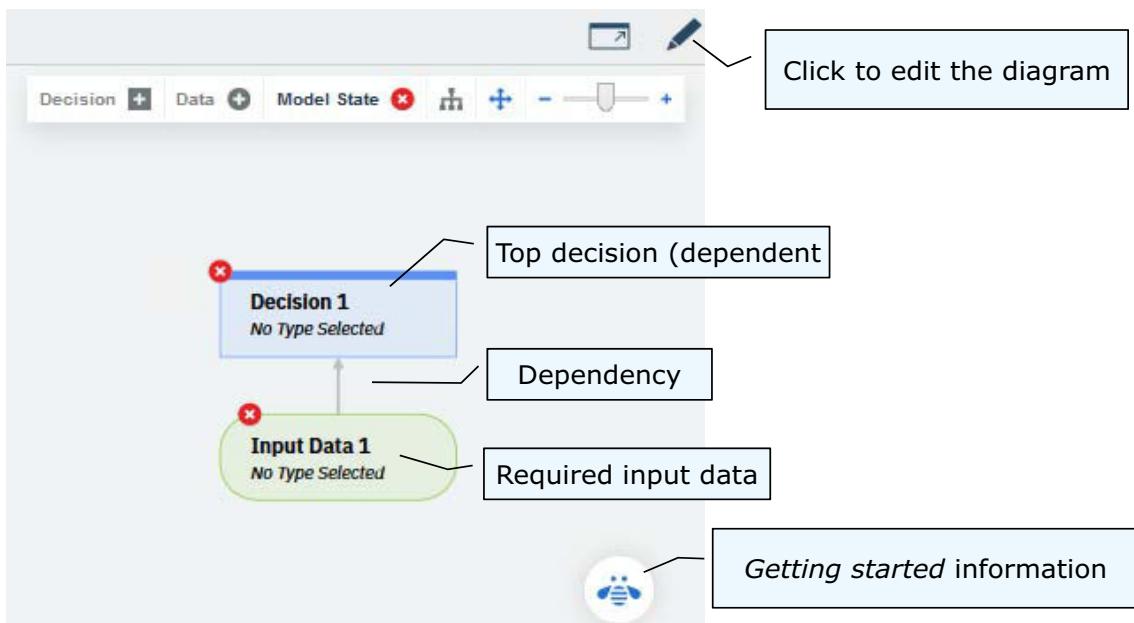
To model a decision, you model the node structure and author the decision logic.

Your decision model outlines what data is required for each decision. Decision nodes require input from data nodes or other decision nodes. Data nodes do not require input.

The modeling notation is inspired from DMN ([www.omg.org/dmn](http://www.omg.org/dmn)).

## Creating the decision model diagram (2 of 2)

- New diagram opens with an empty decision node linked to an empty data node



*Figure 3-11. Creating the decision model diagram (2 of 2)*

When you create a new decision model, the diagram view opens with a decision node and a data node.

Generally, as you build your diagram, you should not include more than 50 nodes.

## Modeling the node structure (1 of 2)

- Decision nodes and data nodes definitions require:
  - Name
  - Data type

| Decision node definition  | Data node definition  |
|---|---|
| <div style="border: 1px solid #ccc; padding: 5px; margin-bottom: 10px;"> <b>Name</b> </div> <div style="border: 1px solid #ccc; padding: 5px; margin-bottom: 10px;"> <b>Output data type</b> </div> | <div style="border: 1px solid #ccc; padding: 5px; margin-bottom: 10px;"> <b>Name</b> </div> <div style="border: 1px solid #ccc; padding: 5px; margin-bottom: 10px;"> <b>Output data type</b> </div> |
| <b>Decision 1</b>   | <b>Input Data 1</b>   |
| <b>Details</b>  | <b>Details</b>  |
| Description (optional)<br><i>Describe the node (optional)</i>   | Description (optional)<br><i>Describe the node (optional)</i>   |
| Output variable name<br><i>Decision 1</i>   | Output variable name<br><i>Input Data 1</i>   |
| <input checked="" type="checkbox"/> Same as decision node   | <input checked="" type="checkbox"/> Same as data node   |
| Output type<br><i>Make a selection</i>  | Output type<br><i>Make a selection</i>  |
|  <i>A data type is required</i>  |  <i>A data type is required</i>  |
| <b>Decision logic</b>   | <b>Default value</b>  |
|  <i>No tables and rules added yet</i>  | <i>Please select a type first.</i>  |

- You can choose default data types or create custom types

Modeling decisions

© Copyright IBM Corporation 2019

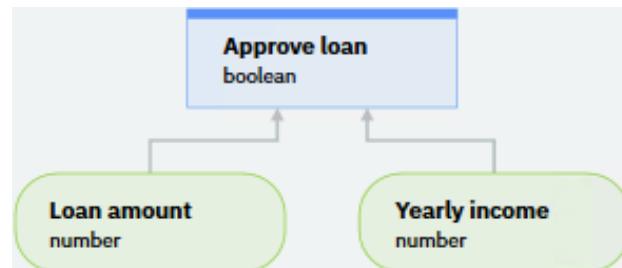
Figure 3-12. Modeling the node structure (1 of 2)

This slide shows the fields that you define when you create decision nodes and data nodes. Data nodes and decision nodes both require a name and a data type. The data type can be a simple type, like string or number. Or you can create custom data types based on the objects in your domain.

## Modeling the node structure (2 of 2)

- Decision nodes require input from:
  - Data nodes
  - Other decision nodes
- Example: Loan approval
  - Main decision: **Approve loan** node
  - Decision output: True or False (Boolean)

- What information is required to make the decision?
  - Amount of the loan request
  - Income of the borrower



- The arrows indicate **Approve loan** decision node depends on the **Loan amount** and **Yearly income** input data nodes to produce a decision

Figure 3-13. Modeling the node structure (2 of 2)

As you begin your model, you start from the final or main decision at the top and work down to capture the input that is required to produce this decision.

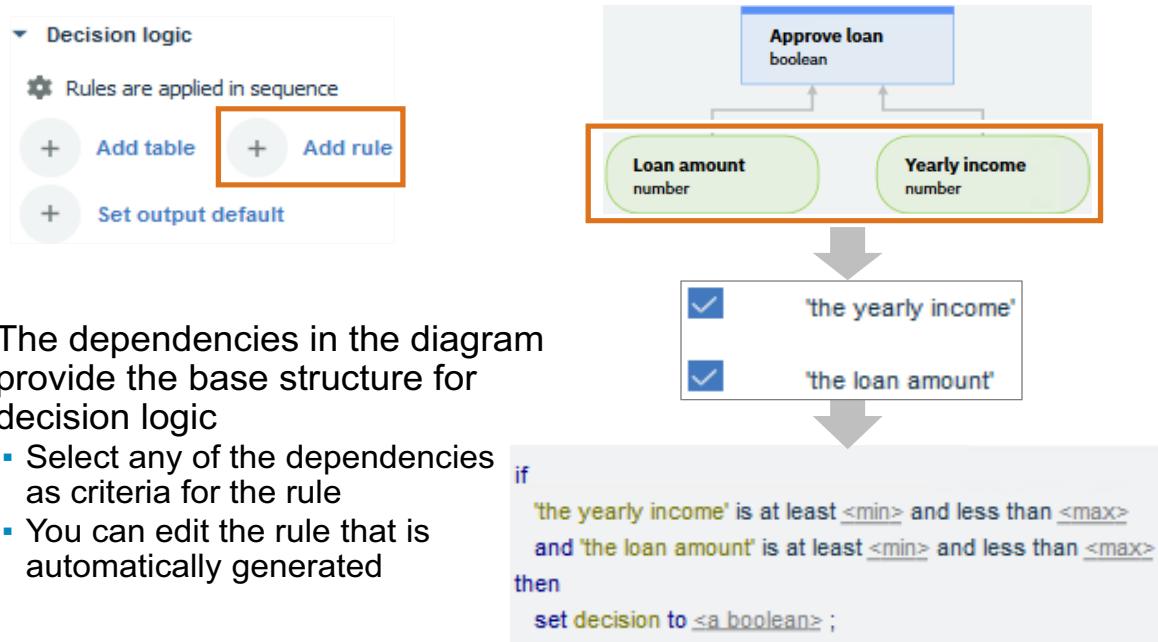
For example, consider a decision model for a loan approval. You start the model with the main **Approve loan** decision at the top. At a minimum, this main decision depends on the **loan amount that is requested** and the borrower's **yearly income**. Notice that the arrow direction indicates dependency.

You can start with a simple model, test it, and progressively add nodes to capture underlying subdecisions. For example, the **Approve loan** decision would depend on subdecisions, such as: is the loan request valid; is the potential borrower eligible; what terms apply to the loan for repayment. Each subdecision would require additional data input. You can validate the model as you add nodes.



## Authoring decision logic (1 of 2)

- Add rules or decision tables to decision nodes to implement decision logic



Modeling decisions

© Copyright IBM Corporation 2019

Figure 3-14. Authoring decision logic (1 of 2)

Decision logic is added to decision nodes. You can add rules and decision tables to decision nodes.

In this example, the **Approve loan** decision node uses the **yearly income** and the **loan amount** nodes to produce the decision. The modeling tool generates a default rule or decision table based on the input data.

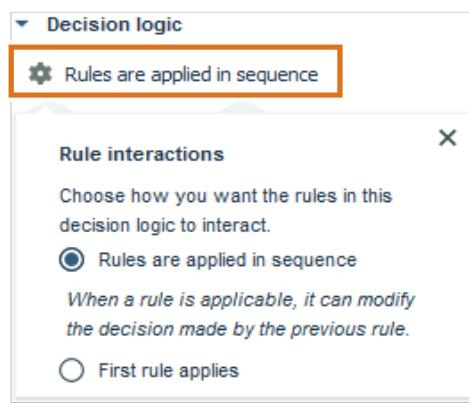
You can edit rules with the Decision Center modeling language, as described in the IBM Knowledge Center at this URL:

[www.ibm.com/support/knowledgecenter/SSQP76\\_8.10.x/com.ibm.odm.dcenter.ref.dc/topics/con\\_dc\\_model\\_ref.html](http://www.ibm.com/support/knowledgecenter/SSQP76_8.10.x/com.ibm.odm.dcenter.ref.dc/topics/con_dc_model_ref.html)



## Authoring decision logic (2 of 2)

- By default, rules execute in the order they are listed
  - The most recently added rules execute first
  - Change the order by dragging the rules
- Use the **Rule interaction** setting to change rule execution



Modeling decisions

© Copyright IBM Corporation 2019

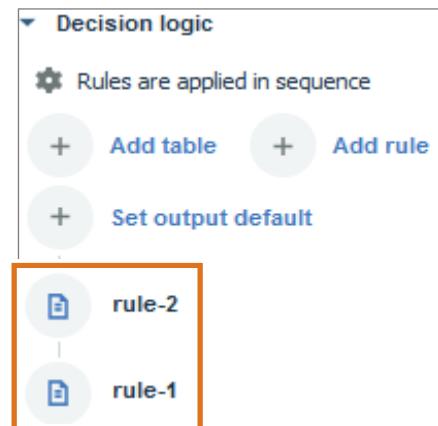


Figure 3-15. Authoring decision logic (2 of 2)

As you add rules in a decision node, those rules are listed sequentially. The same order is used for rule execution. If you need to change the order of those rules, you can manually drag the rules into the correct sequence. You can also use the **Rule interaction** setting.

When you test the behavior of your rules, you can see *how* the sequence in which they execute affects your results.



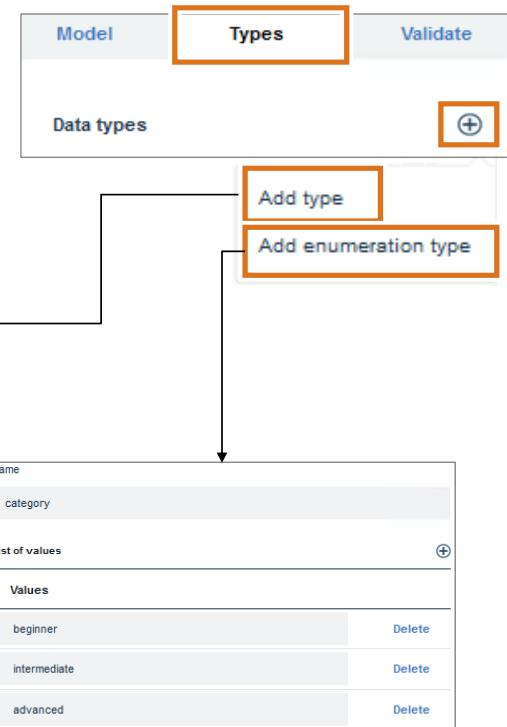
## Defining custom types

- Define custom types on the **Types** tab

- Create types to define objects

Type details

|            |        |                                 |
|------------|--------|---------------------------------|
| Name       | person | X                               |
| Attributes |        |                                 |
| Name       | Type   | List                            |
| name       | string | <input type="checkbox"/> Delete |
| age        | number | <input type="checkbox"/> Delete |
| address    | string | <input type="checkbox"/> Delete |



- Create enumeration types to define lists of values, such as for domains

Modeling decisions

© Copyright IBM Corporation 2019

Figure 3-16. Defining custom types

As mentioned earlier, you can create custom types based on the objects in your domain. You can also create enumeration types to define lists of values, such as for domains.

These custom types can be used as output types for decision nodes and data nodes.

### 3.3. Modeling with IBM Decision Composer

## Modeling with IBM Decision Composer

Modeling decisions

© Copyright IBM Corporation 2019

*Figure 3-17. Modeling with IBM Decision Composer*

## IBM Decision Composer on IBM Cloud

- Available at <https://decision-composer.ibm.com>
  - Free online tool
  - Requires an IBM ID
  - Run up to 1000 decision services per month
- Same modeling tools as Business console
- Fully compatible with Business console
  - Export models built in Decision Composer
  - Import them to Business console for testing, simulation, governance, and deployment

Figure 3-18. IBM Decision Composer on IBM Cloud

In addition to using the modeling tools in Decision Center Business console, you can use IBM Decision Composer, which is a free online tool. Decision Composer provides the same modeling tools that you have in Decision Center.

Decision Composer is fully compatible with Decision Center. Models can be exported from Decision Composer and imported to Decision Center Business console.



## IBM Decision Composer online

- Start from scratch or from extensive Samples library

The interface features three vertical cards:

- Samples:** Shows a hot air balloon icon. Description: "You can import industry relevant samples to explore how decisions can be built and used." Call-to-action: [Explore samples](#)
- Create decisions:** Shows a rocket launching icon. Description: "You can go ahead and create a new decision right now. Our tutorial system is on hand to help you every step of the way." Call-to-action: [Create a new decision](#)
- Decision library:** Shows an open book icon. Description: "This is the main resource center where you can find and work with decisions you've created and shared decisions." Call-to-action: [Browse the library](#)

- Use the **IBM Decision Composer** home icon to navigate back to the main page



Modeling decisions

© Copyright IBM Corporation 2019

Figure 3-19. *IBM Decision Composer online*

After signing in, you see these entry points:

- Samples
- Create decisions
- Decision library

Decision Composer includes an extensive selection of samples that you can explore to help you learn how to improve your decision models.

The screenshot shows the 'Decision Composer online' interface. At the top left is the 'IBM Training' logo, and at the top right is the 'IBM' logo. Below the header, the title 'Decision Composer online' is displayed. A bullet point lists: 'Import complete decision models from the extensive selection of samples to your Decision Library'. Below this, a navigation bar includes a back arrow and the text 'Decision Library / Import samples'. To the right of the navigation bar is a purple 'Import' button, which is highlighted with an orange border. The main content area contains a message: 'Select one or more samples to import to the Decision Library'. Two sample cards are shown: 'Greetings' (highlighted with an orange border) and 'Runner'. The 'Greetings' card describes how to compose greetings based on current hour and gender. The 'Runner' card describes recommending races for runners based on weekly mileage, average speed, and other factors.

Modeling decisions

© Copyright IBM Corporation 2019

Figure 3-20. Decision Composer online

To get started in Decision Composer, you might want to select one of the samples and import it to your library. From the library, you can explore how the model is made and try running it with the test data that is provided.



## For more information

- See the IBM Decision Composer Help resources

The screenshot shows the IBM Decision Composer interface. On the left, there's a sidebar with a globe icon and a list of links: 'Learn about modeling', 'Learn more', 'Invite me to Slack', 'Go to slack', 'IBM developerWorks', 'Get in touch', and 'About Decision Composer'. The 'About Decision Composer' link is highlighted with an orange box. At the top right, there's a 'First time here?' button with the text 'Follow the getting started guidance!'. Below it, a 'Getting started overview' section is shown, featuring a bee icon and a list of tips categorized by model element. A callout box points from the 'Learn about modeling' link in the sidebar to the 'About Decision Composer' link.

**Links to documentation**

**Getting started tips**

**First time here?**  
Follow the getting started guidance!

**Getting started overview**

**How to do what?**  
Start a quick tour around:

- 1 Toolbar
- 2 Model state
- 3 Orientation
- 4 Model
- 5 Types
- 6 Validation

**How do I model?**  
Model a diagram with simple tips

|                                   |                                |
|-----------------------------------|--------------------------------|
| Decision diagram<br><i>2 tips</i> | Decision node<br><i>5 tips</i> |
| Data node<br><i>4 tips</i>        | Create a link<br><i>2 tips</i> |
| Validation<br><i>2 tips</i>       |                                |

Modeling decisions

© Copyright IBM Corporation 2019

Figure 3-21. For more information

For help with using Decision Composer, the interface provides tooltips and links to documentation.

Make sure to download the ODM Decision Modeling guide, available at this URL:

[developer.ibm.com/odm/wp-content/uploads/sites/38/2018/10/ibm\\_odm\\_decision\\_modeling.pdf](http://developer.ibm.com/odm/wp-content/uploads/sites/38/2018/10/ibm_odm_decision_modeling.pdf)

## Unit summary

- Explain when to use a decision model service
- Describe how to model decisions

*Figure 3-22. Unit summary*

## Review questions

1. True or False: Business users can create decision services from scratch in Decision Center Business console.
2. True or False: Decision model services cannot be executed in a production environment against real data.



Modeling decisions

© Copyright IBM Corporation 2019

Figure 3-23. Review questions

Write your answers here:

- 1.
- 2.

## Review answers (1 of 2)

1. True or False: Business users can create decision services from scratch in Decision Center Business console.  
The answer is True.
  
2. True or False: Decision model services cannot be executed in a production environment against real data.  
The answer is False. Decision model services are simplified decision services but they are fully consumable by enterprise applications.



Figure 3-24. Review answers (1 of 2)

## Exercise: Modeling decisions

Modeling decisions

© Copyright IBM Corporation 2019

*Figure 3-25. Exercise: Modeling decisions*

## Exercise introduction

- Create a model diagram
- Define the decision and data node structure
- Create custom data types
- Author the business logic in decision modeling language
- Test the model
- Export and import models from IBM Decision Composer
- Deploy and run a decision model service



Figure 3-26. Exercise introduction

## Exercise overview

- During the exercise, you create a decision model that matches runners to races based on the following criteria:
  - The ability of the runner must match the level of difficulty of the race
  - The runner and the race must be in the same location
- What is the “final” decision:
  - A **runner** is matched to one or more **races**
- What are the “things” the decision is about:
  - Runner
  - Race
- What influences the decision:
  - **Location** of the runner + **location** of the race
  - **Ability level** of the runner + **difficulty level** of the race



*Figure 3-27. Exercise overview*

During the exercise, you create a decision model based on the scenario of matching runners to appropriate races. The match is based on the ability of the runner and the difficulty level of the race. You see how to model the runner, the race, and how to author the decision logic to create a match.

---

# Unit 4. Programming with business rules

## Estimated time

00:45

## Overview

This unit describes the rule engine and how rule execution works. It also describes the rule execution modes.

## How you will check your progress

- Review

## Unit objectives

- Describe the rule engine
- Describe rule execution
- Explain rule execution modes and execution principles

*Figure 4-1. Unit objectives*

## Topics

- Introducing ruleset integration and execution
- What is a rule engine?
- Understanding rule execution modes
- RetePlus example: Trucks and drivers

*Figure 4-2. Topics*

## 4.1. Introducing ruleset integration and execution

## Introducing ruleset integration and execution

Programming with business rules

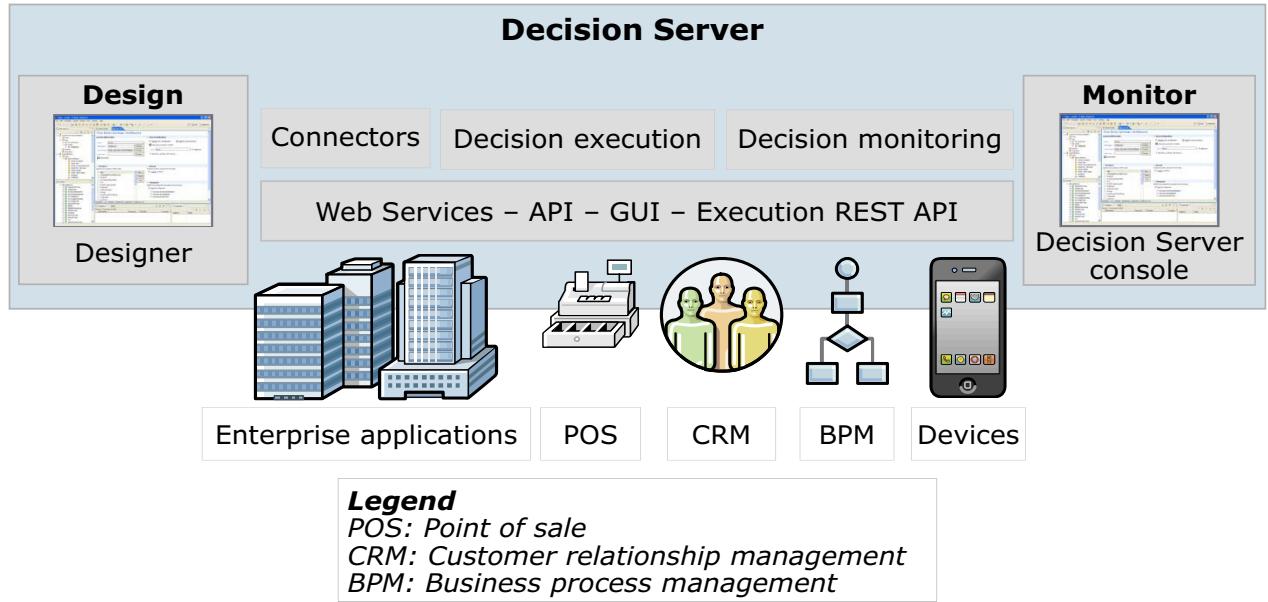
© Copyright IBM Corporation 2019

*Figure 4-3. Introducing ruleset integration and execution*



## Integrating Operational Decision Manager

- Integration makes decision services accessible to your business solution
- Integration steps vary according to the execution environment



Programming with business rules

© Copyright IBM Corporation 2019

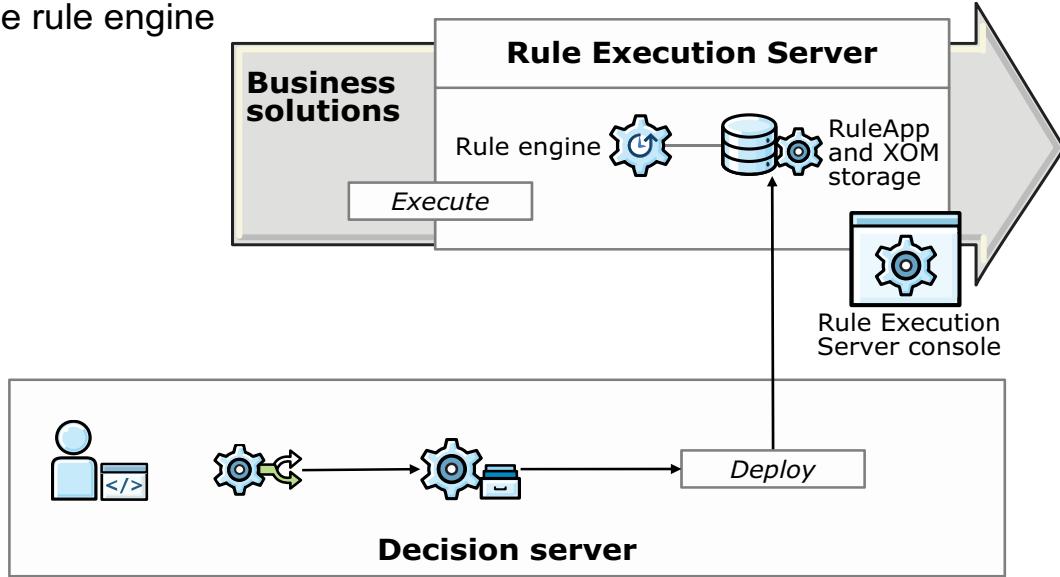
Figure 4-4. Integrating Operational Decision Manager

The objective in integrating Operational Decision Manager into your architecture is to integrate the business logic (which is embodied by the ruleset or decision service) in your *execution environment* for its execution by the *rule engine*.

Integration steps vary according to the execution environment.

## Integration steps

- 1. Packaging:** Extract rules and package as an executable container
- 2. Deployment:** Make ruleset available to rule engine
- 3. Execution:** The client application sends requests for ruleset execution to the rule engine



Programming with business rules

© Copyright IBM Corporation 2019

Figure 4-5. Integration steps

Integration follows these basic steps:

### 1. Packaging

The ruleset is packaged into a RuleApp, which is a deployable management unit. To prepare for deploying decision services, you create a deployment configuration from which you can easily create and deploy a RuleApp archive.

### 2. Deployment

In the deployment configuration, you choose which decision operations to deploy and which server to deploy to. When you deploy, the ruleset becomes accessible to the client application's execution environment.

### 3. Execution

You integrate the execution of your rule by writing the code to run the rule engine on this RuleApp in your client application. Your client application sends a request for a decision, and the rule engine executes the ruleset or decision operation.

## Execution environments

- Ruleset execution can be embedded or managed
- Embedded
  - The engine is integrated with a Java application
  - Use to run rules locally for test purposes
- Managed
  - Rule execution is managed with Rule Execution Server

*Figure 4-6. Execution environments*

The rule engine can either be embedded directly in your client application or managed within the Rule Execution Server, which provides a managed environment for rule execution.

- A common usage of an embedded rule engine is for conducting local tests. You use this option when you debug your rules with Rule Designer.
- You use the managed execution within the Rule Execution Server when your business rule application is deployed in your enterprise environment.

## 4.2. What is a decision engine?

## What is a decision engine?

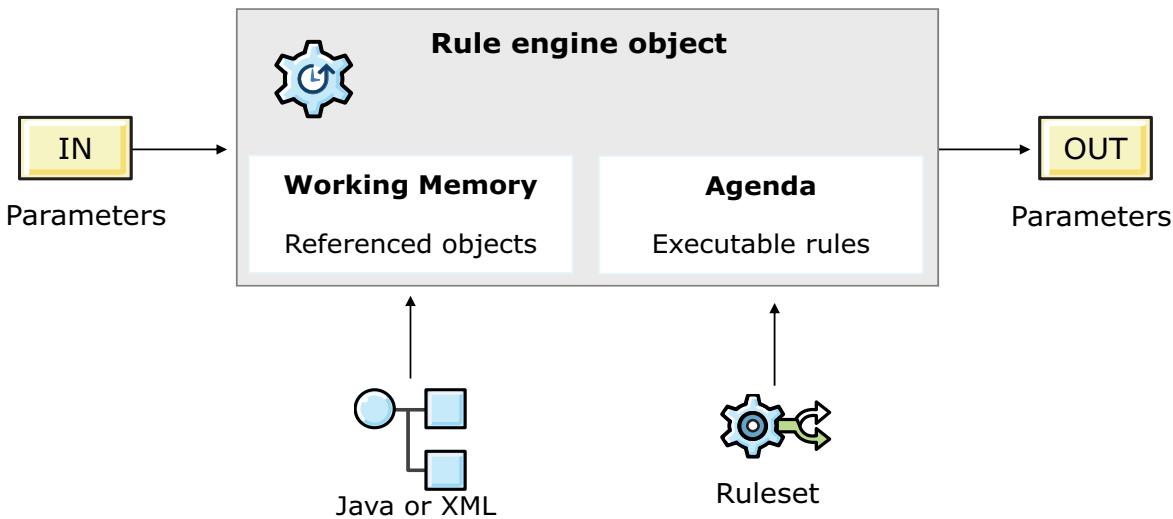
Programming with business rules

© Copyright IBM Corporation 2019

*Figure 4-7. What is a decision engine?*

## Engine

- The engine is the module that executes the rules



Programming with business rules

© Copyright IBM Corporation 2019

*Figure 4-8. Engine*

The rule engine is a Java object for executing rules.

The rule engine interacts with the application objects and the rules in the following way:

- References to the application objects are added to the rule engine. Through the XOM, the rule engine accesses the application objects. The rule engine uses these references to monitor the application objects.
- The rule engine processes the rules that you provide. It evaluates the rules against the application objects and executes the rules when appropriate.

The selection of the rules and the order in which they are selected also depend on the execution mode.

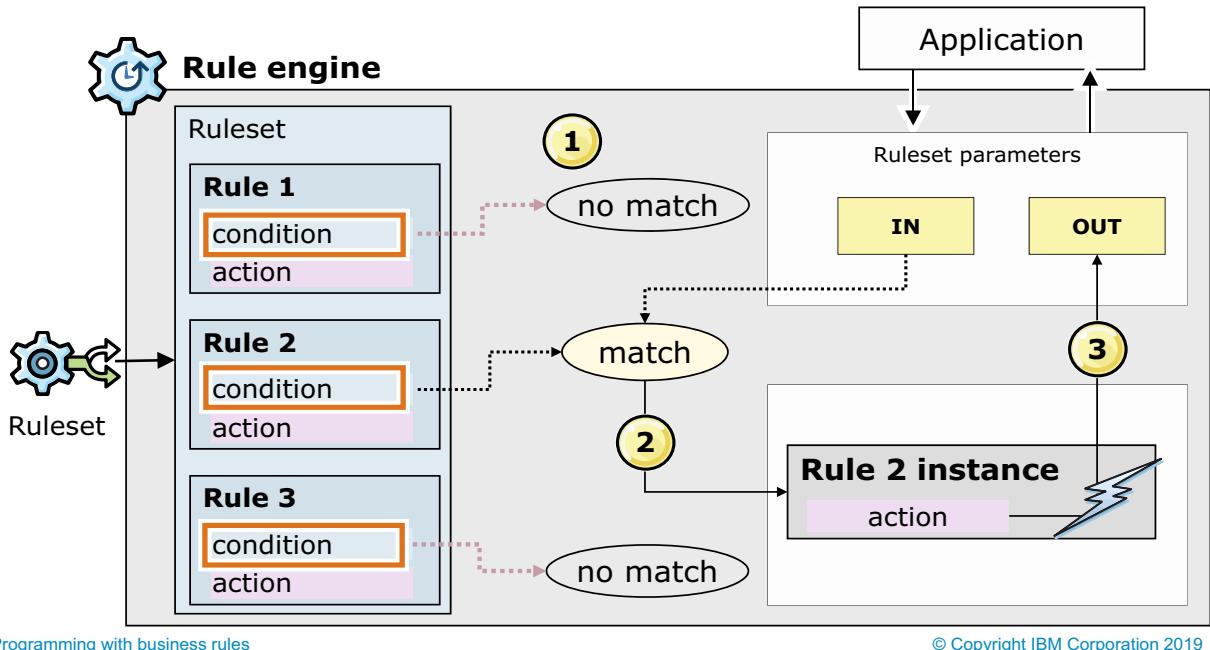
The default execution mode is Fastpath, but you can also select the RetePlus or the sequential execution modes.

For more information, see the product documentation for the topic:

[www.ibm.com/support/knowledgecenter/SSQP76\\_8.10.x/com.ibm.odm.dserver.rules.designer.run/executing\\_decision\\_topics/con\\_decision\\_engine.html](http://www.ibm.com/support/knowledgecenter/SSQP76_8.10.x/com.ibm.odm.dserver.rules.designer.run/executing_decision_topics/con_decision_engine.html)

## Rule execution by the rule engine

- The engine compares the objects to all of the rule conditions of each rule in the ruleset to determine which rule actions to execute



Programming with business rules

© Copyright IBM Corporation 2019

Figure 4-9. Rule execution by the rule engine

This diagram outlines how the rule engine performs rule execution.

As shown on the diagram, the two main steps of the rule execution by the rule engine are as follows:

- In the first step, the rule engine compares each of the objects that are passed from the application to the conditions of each rule in the ruleset. The engine evaluates all the rules against every object. This process is called *pattern matching*.
- The rule engine creates a *rule instance* for each match between a business rule and an object or group of objects.
- The rule engine executes the rule instance by completing the rule action. The execution mode determines how the rule instance is executed.

## Decision engine API

- A decision engine is an instance of the Engine interface
  - An application can contain several rule engine objects
- Use the decision engine API to define the loading of ruleset archives into an engine loader, which represents the result of compilation of a ruleset
- Decision Server class library includes these packages for different components:
  - com.ibm.rules.engine.runtime
  - com.ibm.rules.engine.ruledef.runtime
  - com.ibm.rules.engine.ruleflow.runtime
  - com.ibm.rules.engine.load

Figure 4-10. Decision engine API

## 4.3. Understanding rule execution modes

## Understanding rule execution modes

Programming with business rules

© Copyright IBM Corporation 2019

*Figure 4-11. Understanding rule execution modes*

## Execution modes

- Control the way rule instances are fired
- Possible execution modes in Operational Decision Manager:
  - Fastpath
  - Sequential
  - RetePlus

Figure 4-12. Execution modes

In Operational Decision Manager, the possible execution modes are Fastpath, sequential, and RetePlus.

## Execution modes: Fastpath (1 of 2)

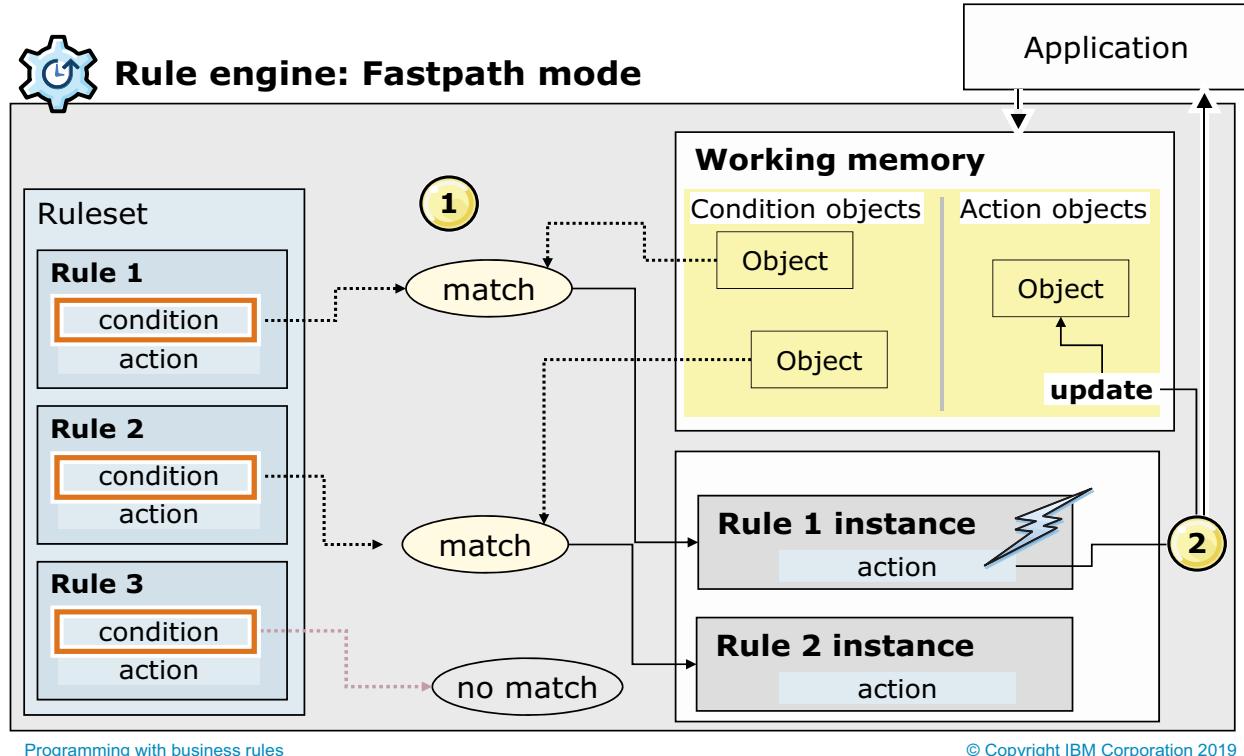


Figure 4-13. Execution modes: Fastpath (1 of 2)

The Fastpath algorithm is the default mode.

Fastpath is a sequential mode of execution, but like RetePlus, it can also detect semantic relationships between rule tests during the pattern matching process.

In Fastpath, the rule engine can use a working memory or parameters.

Like in RetePlus, Fastpath also involves the pattern matching process. Instead of using an agenda, it creates a tree that is based on semantic relationships between rule condition tests (**step 1**).

For each match, a rule instance is created and immediately fired. When a rule instance is fired, and the action is executed, it might modify objects in the working memory. However, as with sequential, these modifications are not accounted for when the pattern matching process is redone (**step 2**).

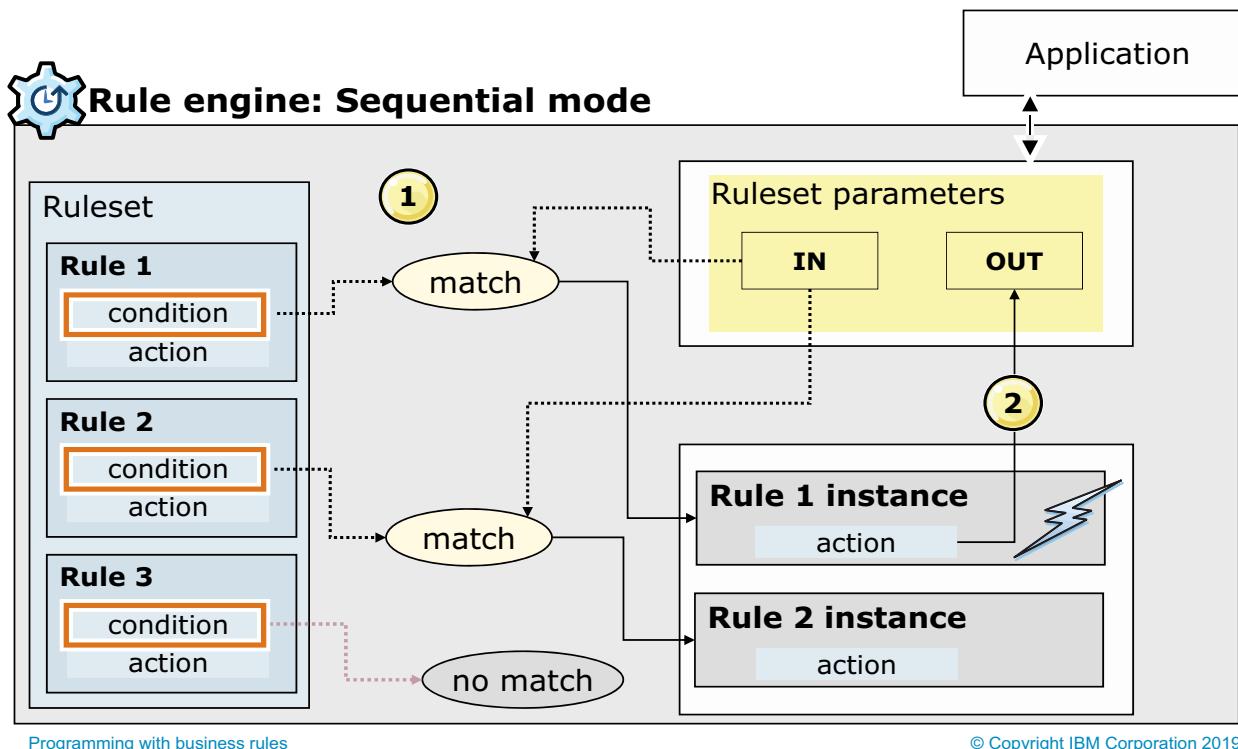
## Execution modes: Fastpath (2 of 2)

- A mode for sequential execution, with detection of semantic relationships between rule tests
- Characteristics:
  - No rule chaining
  - No agenda
  - Input: Objects in the working memory, or ruleset parameters
- For applications that do validation and compliance, or stateless correlation between objects
- Appropriate for rulesets with:
  - Rules with shared test patterns
  - Rules with heterogeneous bindings
  - Rules that use ruleset parameters or working memory objects

Figure 4-14. Execution modes: Fastpath (2 of 2)

Because Fastpath combines features of RetePlus for pattern matching, along with features of sequential for rule firing, this algorithm can produce good performance for correlation applications, validation, and compliance applications.

## Execution modes: Sequential (1 of 2)



Programming with business rules

© Copyright IBM Corporation 2019

Figure 4-15. Execution modes: Sequential (1 of 2)

The next algorithm is sequential, and as its name suggests, in sequential mode, the rule engine executes rule instances in their order of appearance, or sequentially.

With the sequential execution mode, the application passes the objects through parameters.

For each match that the rule engine detects between an object and a rule condition, a rule instance is created and immediately fired; it is not stored anywhere. When a rule instance is fired, the corresponding rule action is executed. If the action modified a value in some way, rules that fire after this update account for that modification. But rules that already fired are not reevaluated for new matches.

Because of its systematic nature, the sequential execution mode fits well with validation and compliance applications.

## Execution modes: Sequential (2 of 2)

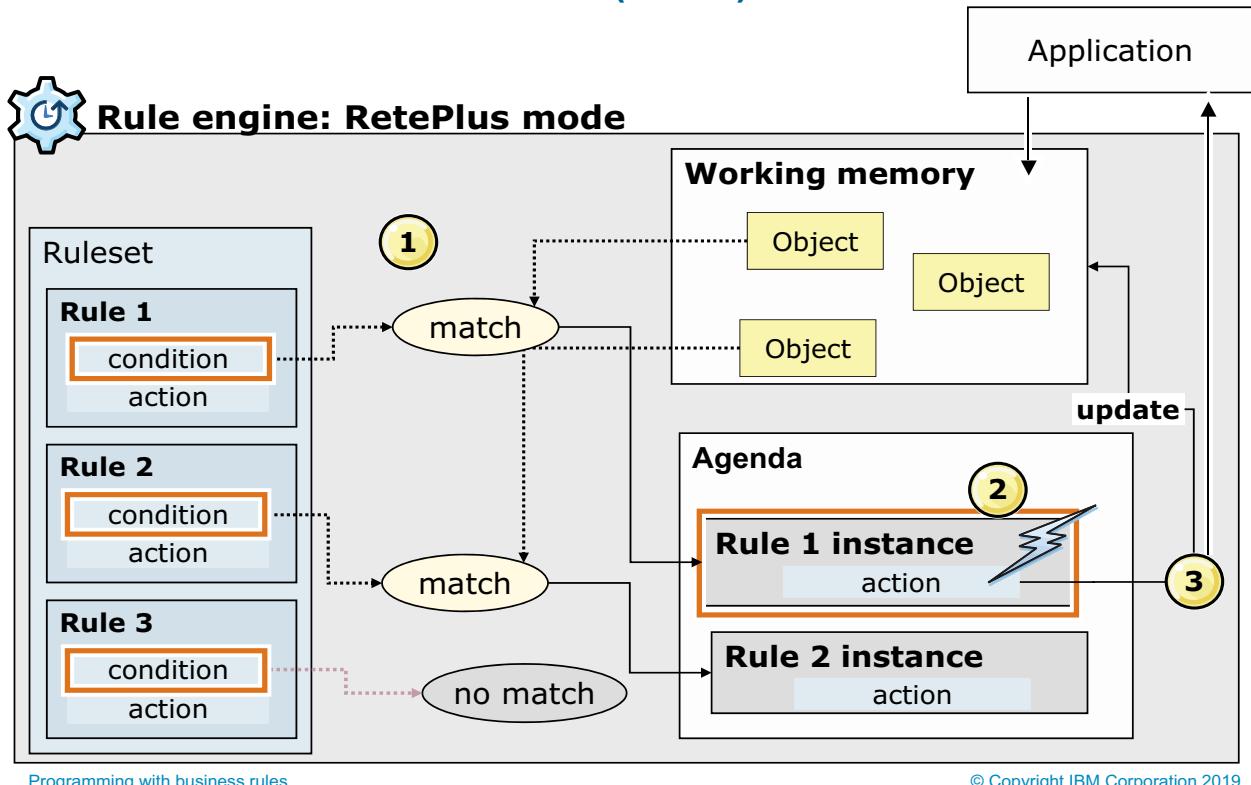
- A mode for sequential execution
- Characteristics:
  - No rule chaining
  - No agenda
  - Input: Ruleset parameters are suggested
- For applications that do validation or compliance
- Appropriate for rulesets with:
  - Numerous rules with randomly ordered tests
  - Rules that use ruleset parameters
  - Rules that use the same class in their conditions, but do different tests on this class

Figure 4-16. Execution modes: Sequential (2 of 2)

When the rules are executed in sequential order, you have no rule chaining, and also have no need for an agenda.

Sequential works with parameters as input. If your application has many rules with randomly ordered tests, and passes date through parameters, you should use the sequential mode.

## Execution modes: RetePlus (1 of 3)



Programming with business rules

© Copyright IBM Corporation 2019

Figure 4-17. Execution modes: RetePlus (1 of 3)

This diagram reviews the steps that the rule engine follows when it runs in RetePlus mode.

The client application passes objects to the rule engine. Generally, with the RetePlus algorithm, instead of using parameters, objects are inserted into *working memory*, which is a logical space to store objects during execution.

As a first step, the rule engine begins the pattern-matching process by testing every condition of every rule in the ruleset with each of the objects in working memory. Every time that a match is found, a rule instance is created and put into the agenda, which is also a logical space where rule instances are stored until they are fired or executed. If a rule condition matches several objects, the agenda can have several rule instances for the same rule. So every match between a rule and an object creates a rule instance that gets stored in the agenda.

In step 2, the rule engine then decides which rule instance in the agenda to fire. This step is where your ordering principles come into play. After a rule instance is fired, it is removed from the agenda, and the corresponding rule action is executed.

In step 3, you see that the executed action might affect the objects that are passed from the application. As a side effect, the action might modify existing objects, create objects, or delete objects. Depending on what update was made, the rule engine might be triggered to repeat the pattern-matching process and reevaluate the rules against the objects to see whether rule instances are created. This process is called *rule chaining*, where execution of one rule can change

an object so that the object then matches another rule. For rule chaining to work, you must ensure that when you define your objects in the BOM, you select the **Update object state** property to notify the rule engine when an object state is updated.

## Execution modes: RetePlus (2 of 3)

- Mode that works in most cases
- Characteristics:
  - Rule chaining
  - Agenda
  - Uses objects in the working memory or ruleset parameters as input
- Excels in incremental, data-driven execution (the execution reacts to changes in the data)

Figure 4-18. Execution modes: RetePlus (2 of 3)

RetePlus works well with most types of rules. Keep in mind its main characteristics: it uses working memory, the agenda, and it can include rule chaining.

## Execution modes: RetePlus (3 of 3)

- Best performance for applications that do computation or correlation between objects
- Appropriate for rulesets with:
  - Rules with dynamic priorities
  - Rule chaining: The execution of a rule might cause the rule engine to fire other rules
  - Rule actions that manipulate working memory objects (update, retract, insert)
  - Event management

Figure 4-19. Execution modes: RetePlus (3 of 3)

This slide provides some additional guidelines to help you identify which types of applications are best suited to using RetePlus mode.

## RetePlus: Rule order and selection

- Refraction
  - Helps prevent *trivial* loops
- Recency
  - Resolve conflicts when several rule instances are candidates for firing at the same time

Figure 4-20. RetePlus: Rule order and selection

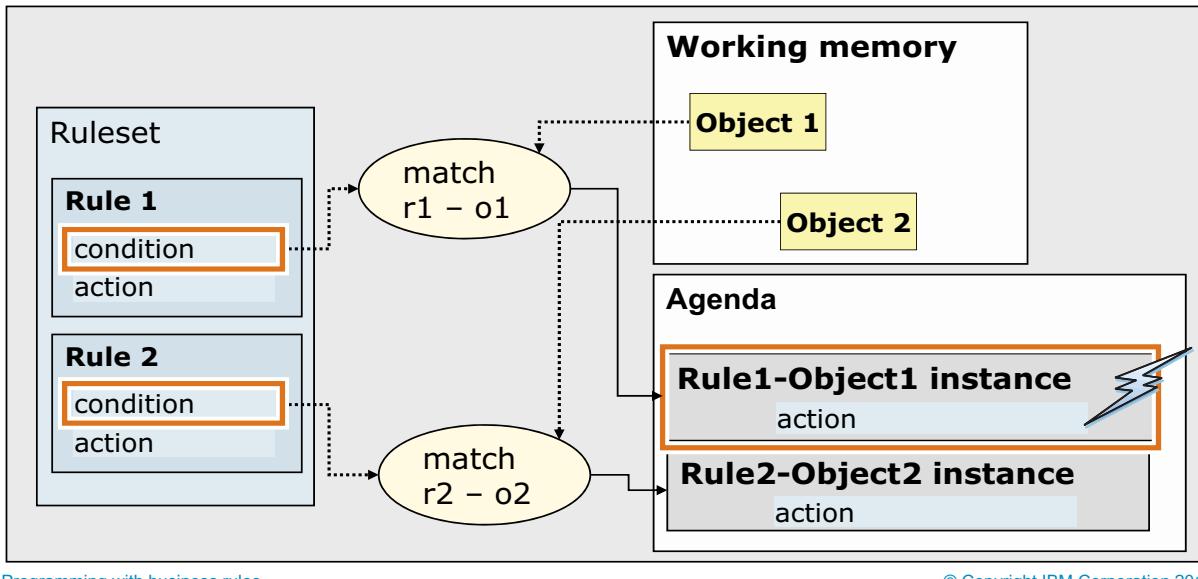
As previously explained, when a rule condition matches objects in working memory, a rule instance is placed into the agenda.

The agenda then orders rule instances and selects the one to fire first, based on the *Refraction*, *Priority*, and *Recency* principles, in this order. The principle of refraction can be used to prevent loops. Priority and recency are used to resolve conflicts when several rule instances are candidates for firing at the same time.

## Refraction (1 of 2)

Example:

- Rule1 matches Object1
- A rule instance [Rule1-Object1] is placed in the agenda



Programming with business rules

© Copyright IBM Corporation 2019

Figure 4-21. Refraction (1 of 2)

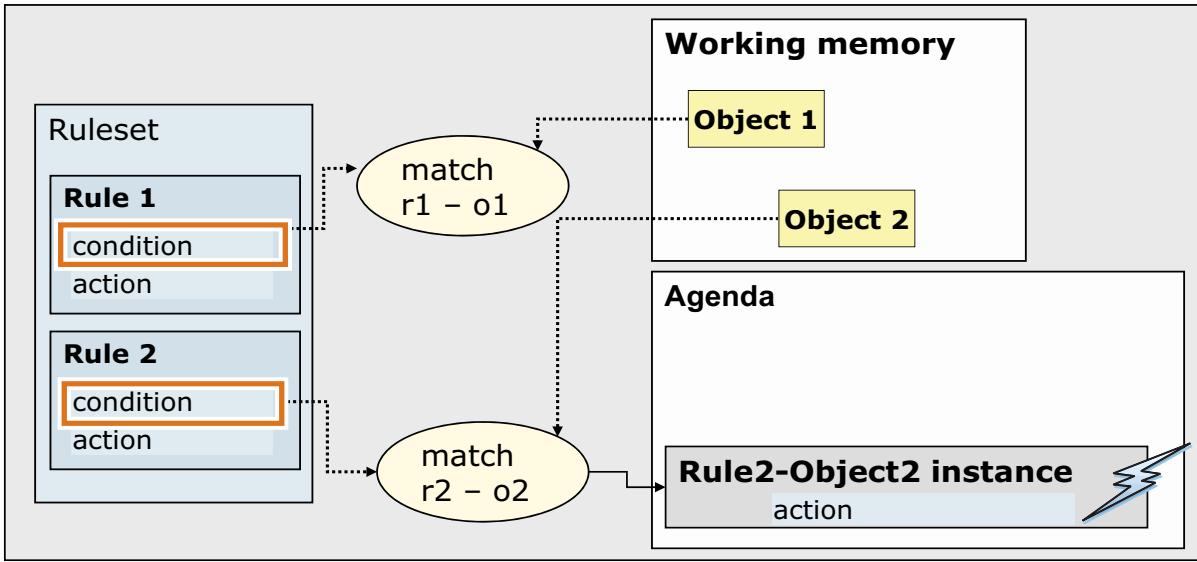
Refraction helps eliminate trivial loops. A rule instance cannot be put back in the agenda if:

- Matched objects are not modified
- Matched objects are modified in such a way that the rule condition continues to hold true

When matched objects are modified in such a way that the condition is no longer met, but later modified so that the condition is met again, the rule is again put in the agenda. For example, if a rule called `Rule1` matches an object that is called `Object1`, a rule instance `[Rule1-Object1]` is placed in the agenda.

## Refraction (2 of 2)

- After [Rule1-Object1] fires, if it does not modify Object1, this rule instance is removed from the agenda
- Rule1 is not reinserted in the agenda, even if Rule1 and Object1 still match



Programming with business rules

© Copyright IBM Corporation 2019

Figure 4-22. Refraction (2 of 2)

With refraction, after [Rule1-Object1] fires, and if it does not modify Object1, this rule instance is removed from the agenda. The rule engine does not reinsert Rule1 in the agenda later, even if Rule1 and Object1 still match.

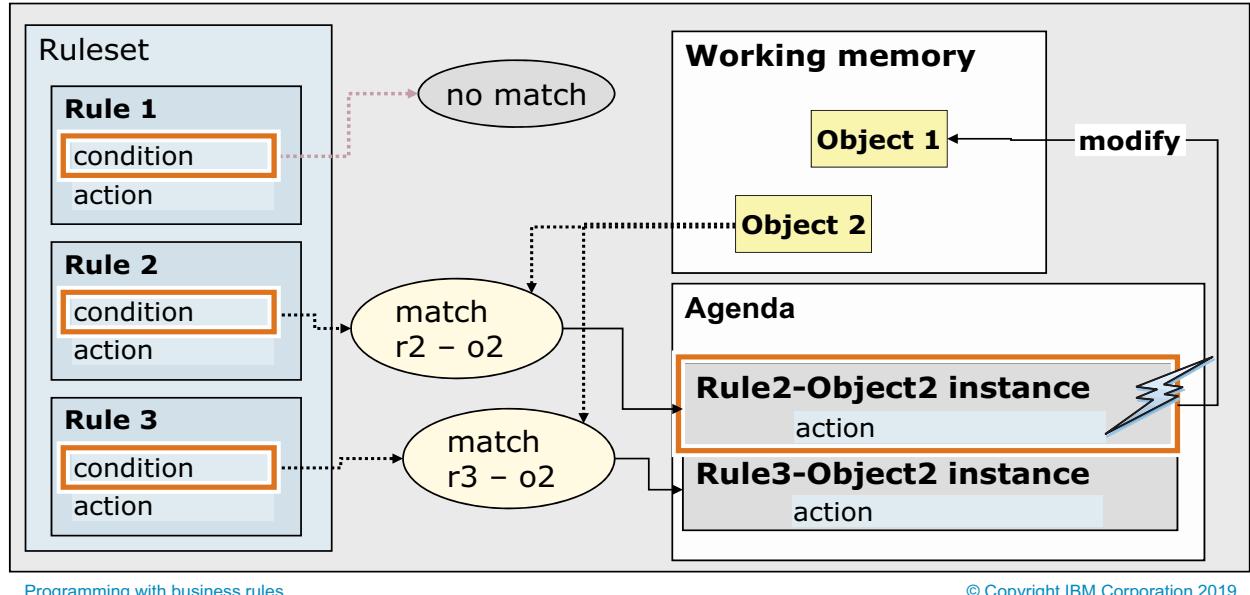


### Note

You can use the rule engine API to override refraction.

## Recency (1 of 2)

- If two rule instances have the same priority, the rule instance for the rule and object that were matched last fires first



Programming with business rules

© Copyright IBM Corporation 2019

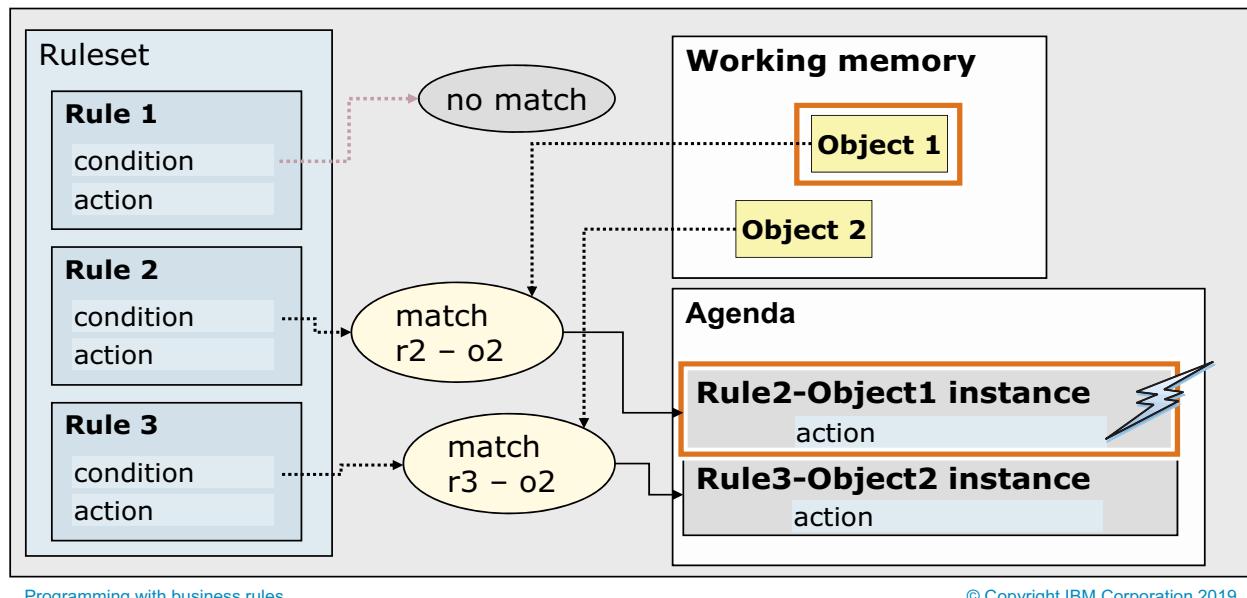
Figure 4-23. Recency (1 of 2)

If two rule instances have the same priority, the rule instance for the rule and object that were matched last is fired first.

For example, as shown in the diagram, two of the rules (**Rule2** and **Rule3**) match **Object2** (but not **Object1**). Two rule instances, [**Rule2-Object2**] and [**Rule3-Object2**], are placed in the agenda. Assuming **Rule2** matched **Object2** after **Rule3** matched **Object2**, [**Rule2-Object2**] fires first (last in, first out).

## Recency (2 of 2)

- Updating an object in the working memory might result in a new match and a new rule instance added in the agenda



Programming with business rules

© Copyright IBM Corporation 2019

Figure 4-24. Recency (2 of 2)

When [Rule2-Object2] fires, it modifies the members of Object1. The modified Object1 now matches Rule2, and the new rule instance [Rule2-Object1] is placed into the agenda. This new rule instance comes from the most recent match.

[Rule2-Object1] is fired before [Rule3-Object2], even though [Rule3-Object2] was already in the agenda.

## Choosing an execution mode

- Execution mode selection

| Choose     | When ...  |
|------------|---|
| RetePlus   | <ul style="list-style-type: none"> <li>All included semantically</li> <li>Stateful application</li> <li>Rule chaining</li> <li>Useful if you have many objects and limited changes</li> </ul>                     |
| Sequential | <ul style="list-style-type: none"> <li>Covers most cases</li> <li>Many rules, few objects; has limitations</li> <li>Use with homogeneous rules</li> <li>Highly efficient in multi-threaded environment</li> </ul> |
| Fastpath   | <ul style="list-style-type: none"> <li>Rules implementing a decision structure, many objects</li> <li>Highly efficient in multi-threaded environment</li> </ul>   |

Figure 4-25. Choosing an execution mode

To make the best choice, answer the following questions:

- What type of application do your rules implement?
- What types of objects do your rules use?
- What is the effect of rule actions?
- What sort of tests do you find in rule conditions?
- What priorities have you set on your rules?

## Choosing execution mode according to application type

| Application type   | Execution mode   |
|--|--|
| <b>Compliance and validation</b> <ul style="list-style-type: none"> <li>Loosely interrelated rules that check a set of conditions to yield a go/no go or similar constrained result</li> <li>Application types: Underwriting, fraud detection, data validation, form validation</li> <li>Business rules generally have a yes or no result, and provide some explanation on the decision</li> </ul> | <b>Fastpath or sequential</b>                                  |
| <b>Computation</b> <ul style="list-style-type: none"> <li>Strongly interrelated rules that compute metrics for a complex object model</li> <li>Application types: Scoring and rating, contracts, allocation</li> <li>Business rules carry out different calculations on an object that is responsible for providing a final value (or rating)</li> </ul>   | <b>RetePlus</b> (if inference is necessary) or <b>Fastpath</b> |
| <b>Correlation</b> <ul style="list-style-type: none"> <li>Strongly interrelated rules that correlate information from a set of objects to compute some complex metrics</li> <li>Application types: Billing</li> <li>Business rules applications insert information</li> </ul>  | <b>RetePlus</b> (if inference is necessary) or <b>Fastpath</b> |
| <b>Stateful session</b> <ul style="list-style-type: none"> <li>Strongly interrelated rules that correlate events in a stateful engine session</li> <li>Application types: Alarm filtering and correlation, GUI customization, web page navigation</li> </ul>   | <b>RetePlus</b> without a ruleflow                             |

Figure 4-26. Choosing execution mode according to application type

## 4.4. RetePlus example: Trucks and drivers

## RetePlus example: Trucks and drivers

Programming with business rules

© Copyright IBM Corporation 2019

Figure 4-27. RetePlus example: Trucks and drivers

## RetePlus in action

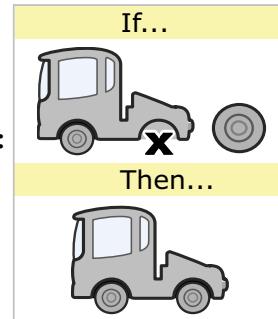
- The ruleset contains these rules:

- ChangeTire:**

If all of the following conditions are true:

- a truck has a flat tire
- a tire is available

Then change the tire;



- AssignDriver:**

If all of the following conditions are true:

- a truck is available
- a driver is available

Then assign the truck to the driver;



Figure 4-28. RetePlus in action

The following graphics illustrate the interaction with the RetePlus execution mode between the rule engine, the working memory, and the agenda.

This example considers a Trucks and Drivers rule project that defines two action rules, `AssignDriver` and `ChangeTire`.

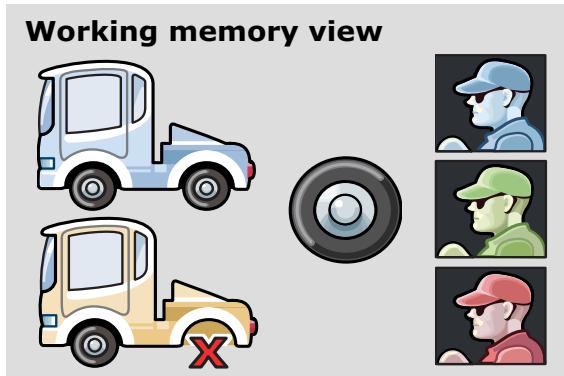
These action rules evaluate objects in working memory that are based on two BOM classes: `Truck` and `Driver`. They do not work on parameters or ruleset variables.

In the ruleflow, the two action rules are configured to execute with the RetePlus algorithm.

Every time an object is modified, its new state might result in a new match to a rule. The rule engine must be notified when objects are modified so that it can repeat the pattern matching process. To ensure this notification, you must select the appropriate **Update object state** options in the `Truck` and `Driver` BOM classes.

At first, the rule engine loads the ruleset (that is, the two `AssignDriver` and `ChangeTire` action rules) as a ruleset.

## Input objects in working memory

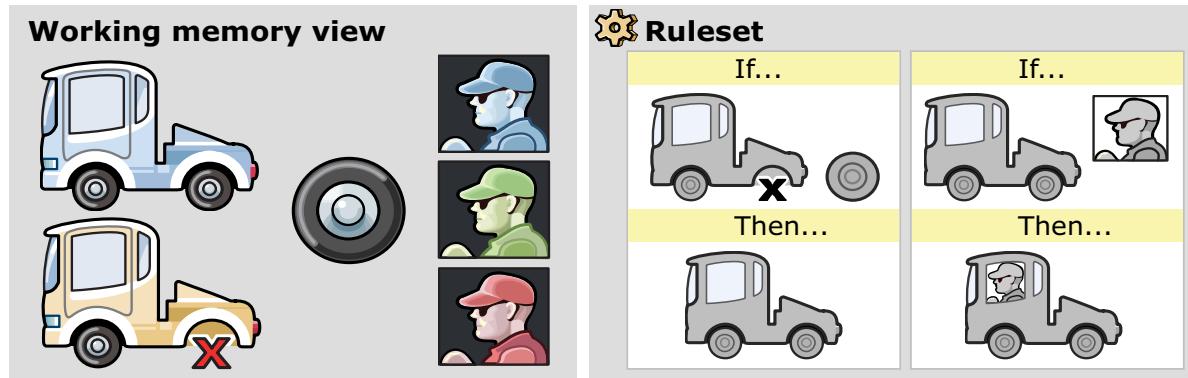


- The client application sends objects as input
- Input objects are passed to the rule engine through the working memory

Figure 4-29. Input objects in working memory

The client application passed these two trucks, a tire, and three drivers as input objects to the rule engine, and these objects are inserted in the working memory.

## Evaluating rules with objects



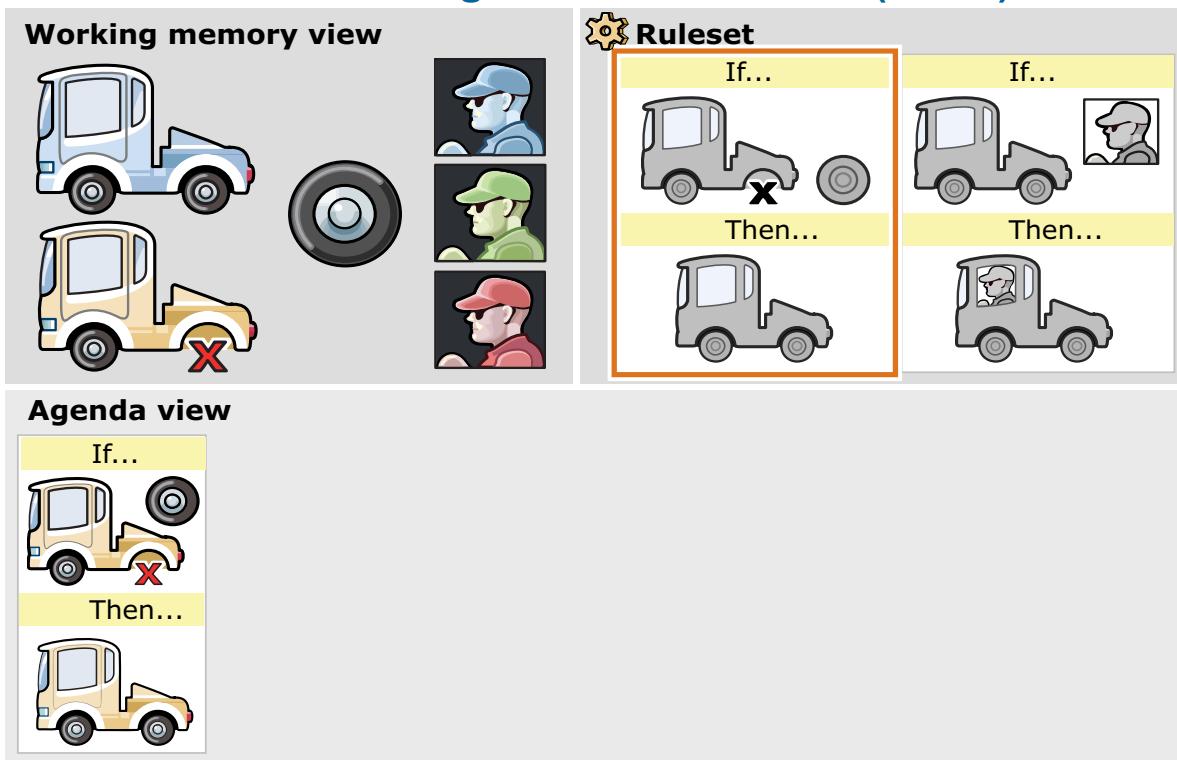
- The rule engine tests each rule condition against each object in working memory

Figure 4-30. Evaluating rules with objects

The rule engine starts the evaluation by testing each rule condition from the two rules in the ruleset against each object in the working memory.



## Rule instances in the agenda for execution (1 of 2)



Programming with business rules

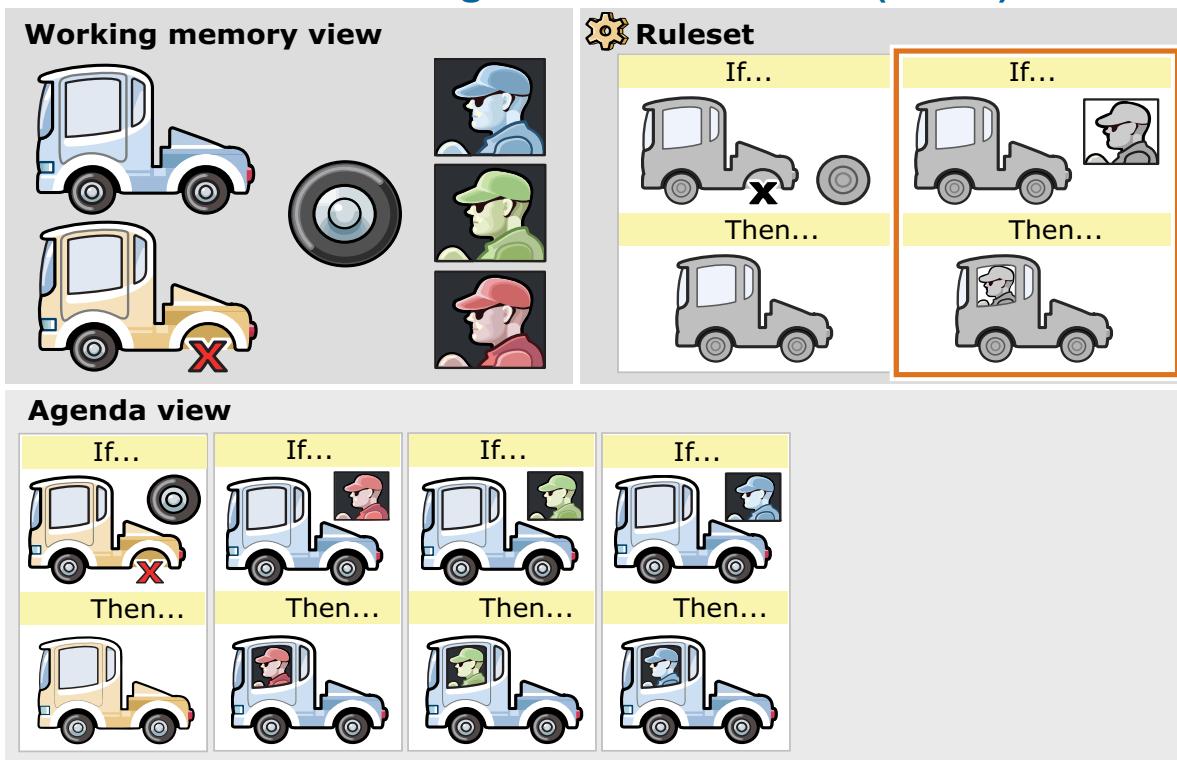
© Copyright IBM Corporation 2019

Figure 4-31. Rule instances in the agenda for execution (1 of 2)

When the rule engine evaluates the `ChangeTire` rule conditions, it finds a matching truck and an available tire in the working memory. Because the rule condition is now true, an instance of the `ChangeTire` rule is put into the agenda.



## Rule instances in the agenda for execution (2 of 2)



Programming with business rules

© Copyright IBM Corporation 2019

Figure 4-32. Rule instances in the agenda for execution (2 of 2)

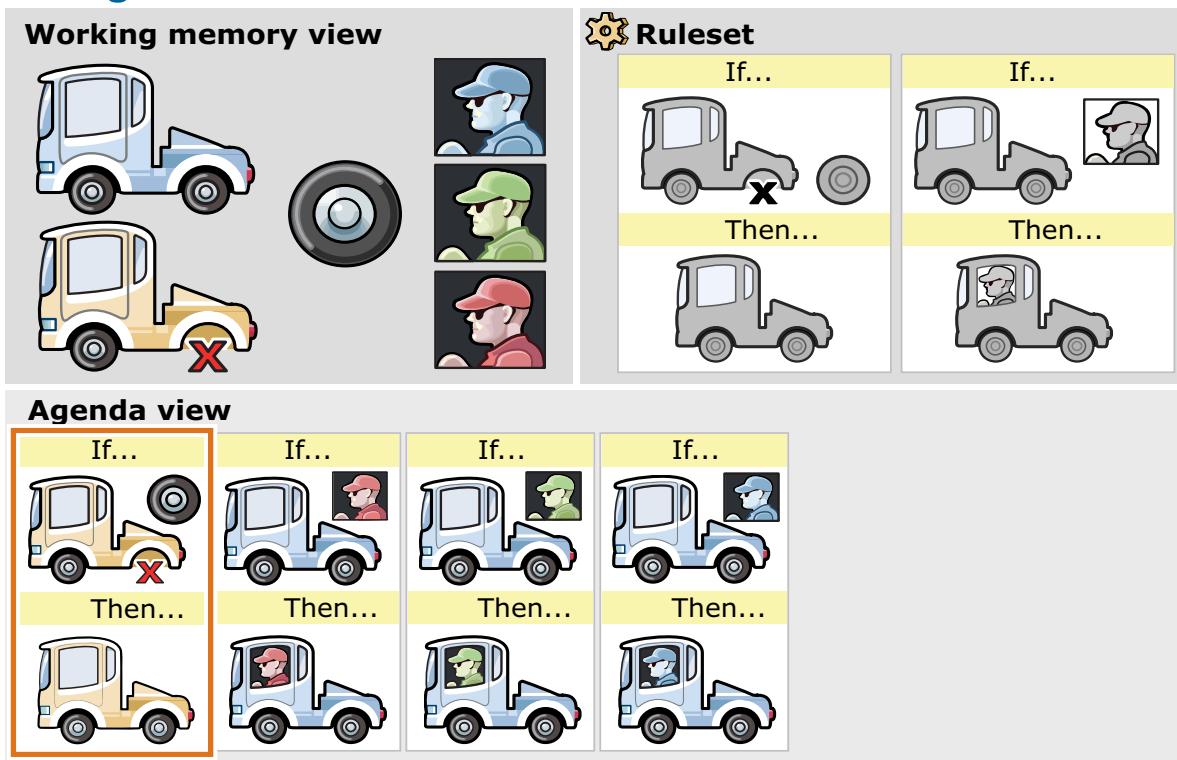
The `AssignDriver` rule evaluates to `true` three separate times:

- Blue truck + driver with blue hat
- Blue truck + driver with green hat
- Blue truck + driver with red hat

Each of these rule instances is put into the agenda.



## ChangeTire executes



Programming with business rules

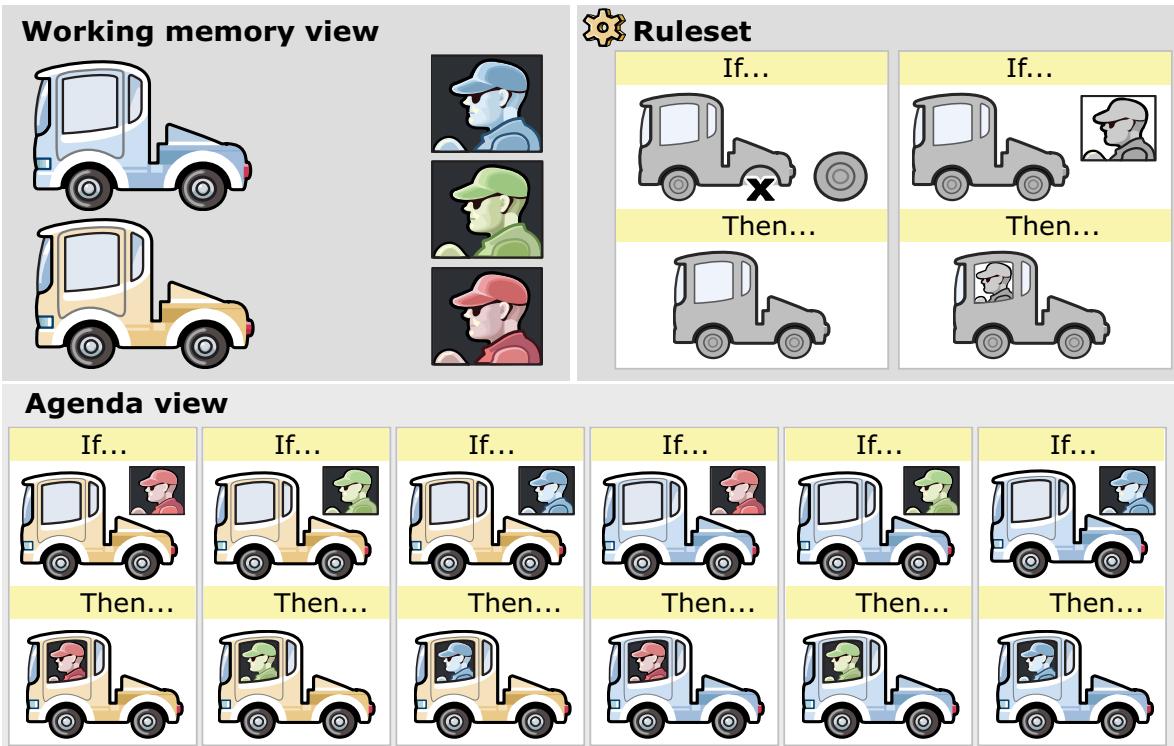
© Copyright IBM Corporation 2019

Figure 4-33. ChangeTire executes

The rule engine selects one of the rule instances to execute.

Consider that the ChangeTire rule has the higher priority, so the ChangeTire rule instance executes first.

## Side effects: Updated objects create matches



Programming with business rules

© Copyright IBM Corporation 2019

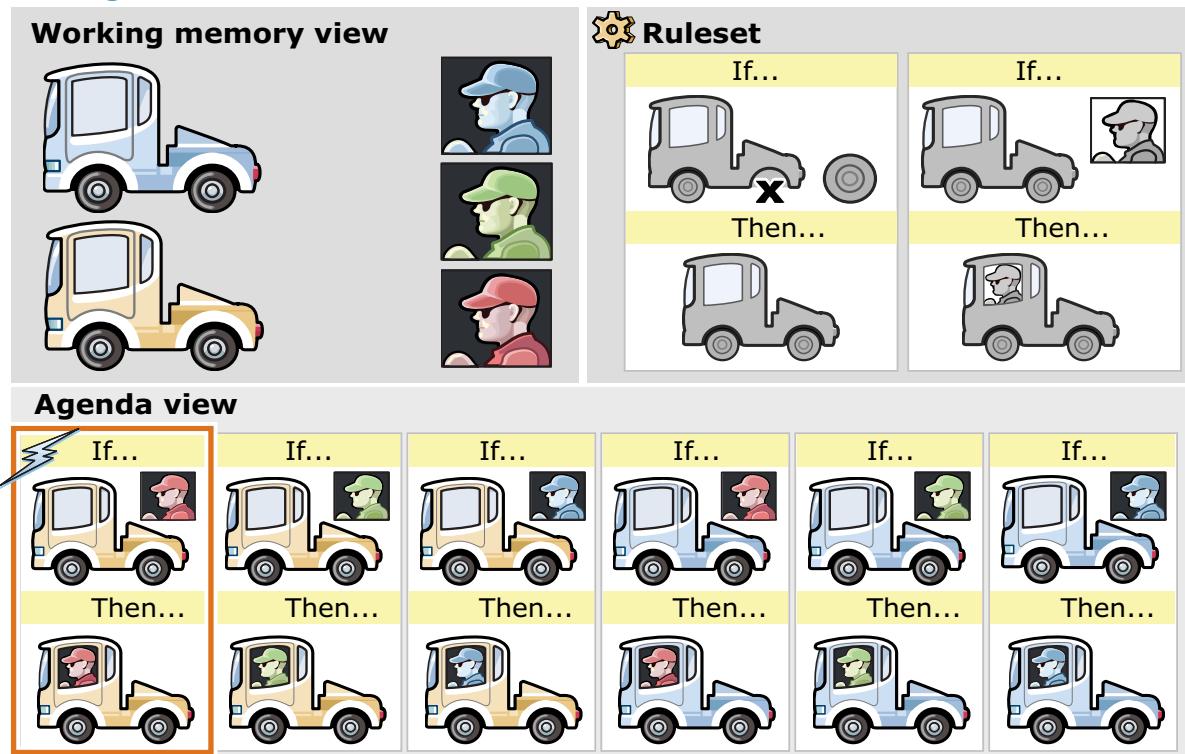
Figure 4-34. Side effects: Updated objects create matches

Immediately after executing the `ChangeTire` rule instance, the objects in the working memory are updated.

As a side effect of the `ChangeTire` rule, a new truck object is now available to be assigned. Because the working memory objects are updated, the rule engine retests all rule conditions against the objects and now finds new matches for the `AssignDriver` rule. The rule engine matches each driver to the newly available brown truck to create three instances of the `AssignDriver` rule in the agenda.



## AssignDriver executes



Programming with business rules

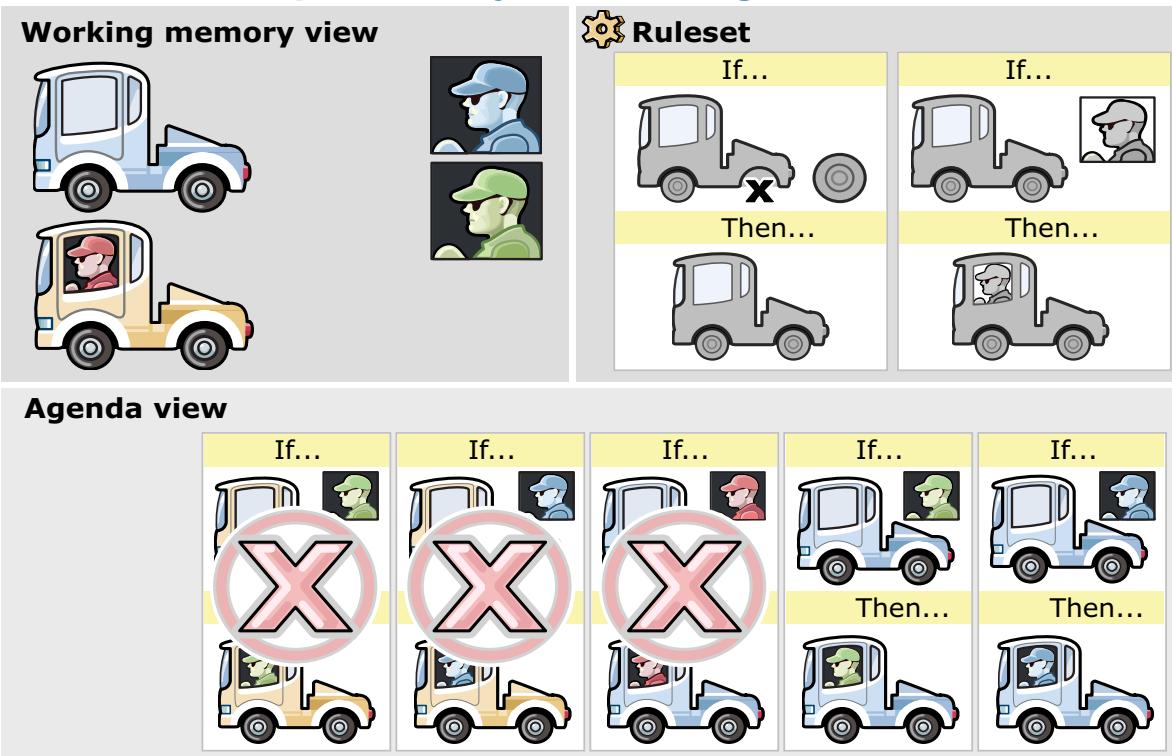
© Copyright IBM Corporation 2019

Figure 4-35. AssignDriver executes

Based on recency (last match fires first), the rule engine executes the rule instance that uses the brown truck and the driver with the red hat.



## Side effects: Updated objects no longer match



Programming with business rules

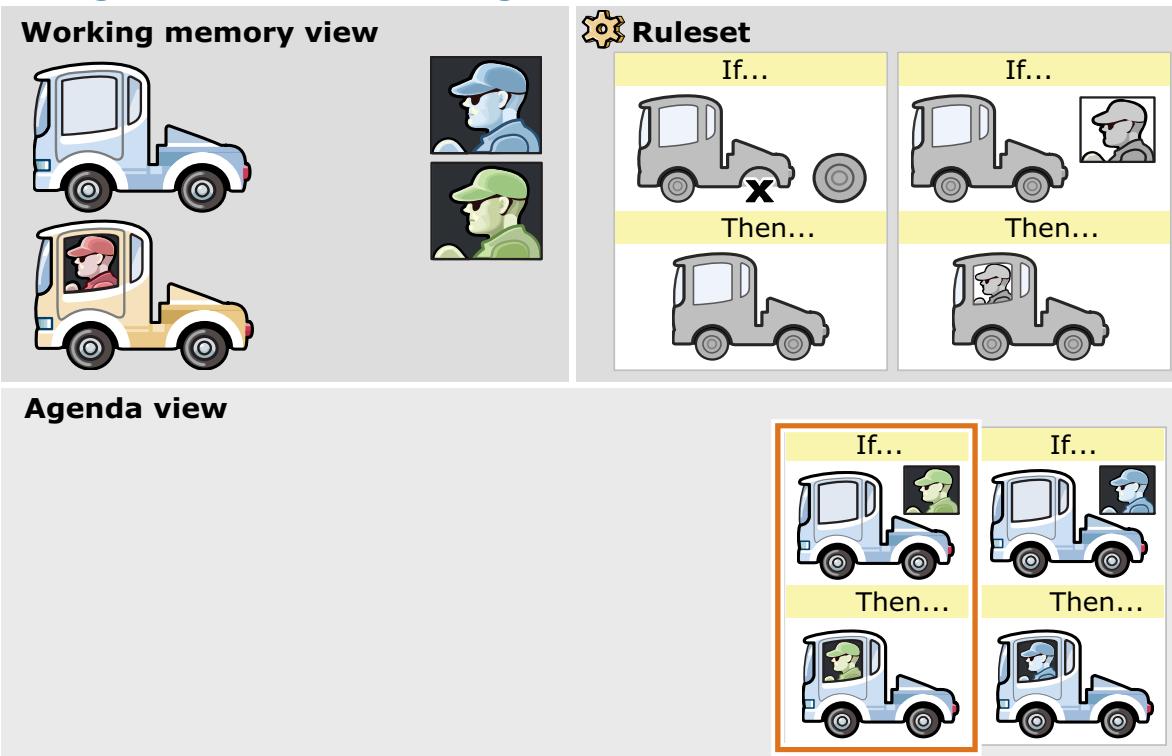
© Copyright IBM Corporation 2019

Figure 4-36. Side effects: Updated objects no longer match

The working memory objects are immediately updated and the rules retested. This time, the AssignDriver rule cannot be matched with the brown truck or the red-hat driver because their state is updated and they are no longer available. As a result, the rule instances that use those objects are deleted from the agenda.



## AssignDriver executes again



Programming with business rules

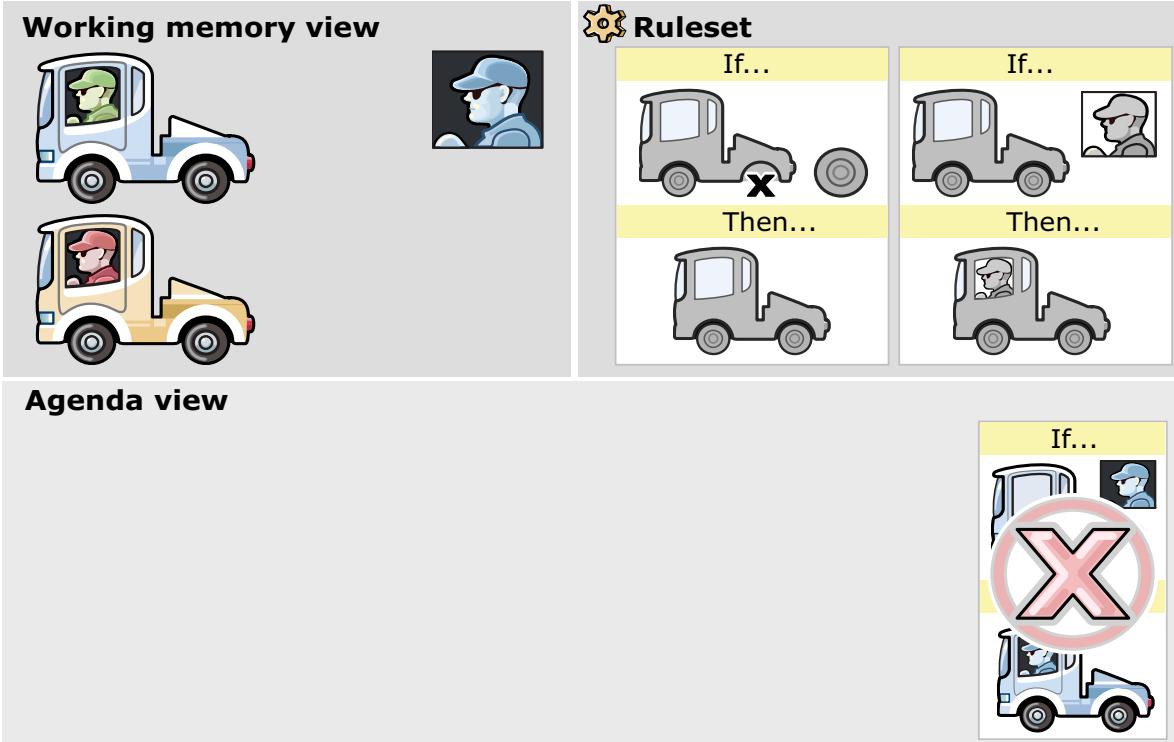
© Copyright IBM Corporation 2019

Figure 4-37. AssignDriver executes again

The rule engine can now execute the next most recent rule instance, which is the AssignDriver rule with the blue truck and the driver with the green hat.



## Side effects of AssignDriver



Programming with business rules

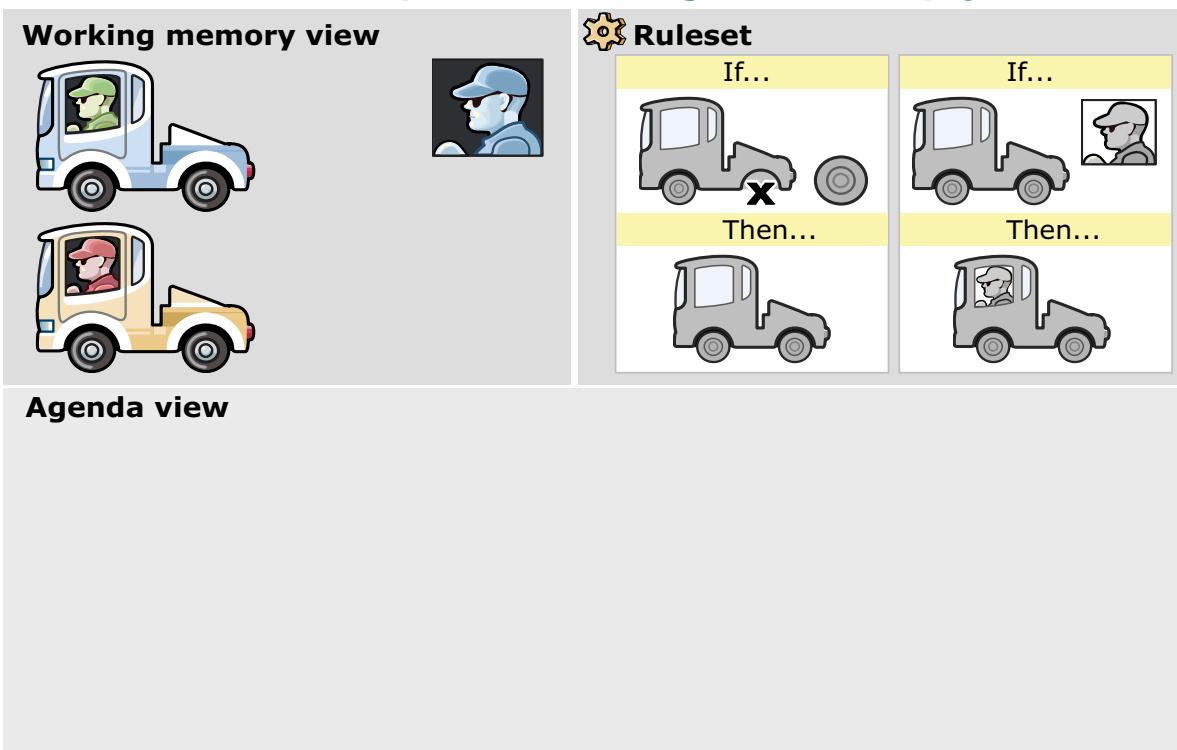
© Copyright IBM Corporation 2019

Figure 4-38. Side effects of AssignDriver

Again, the working memory is updated and the rules retested. The blue truck is no longer available so the remaining rule instance is deleted from the agenda.



## Rule execution completes when agenda is empty



Programming with business rules

© Copyright IBM Corporation 2019

*Figure 4-39. Rule execution completes when agenda is empty*

No other matching objects can be found for the ruleset, so the agenda is empty and rule execution is complete.

## Unit summary

- Describe the rule engine
- Describe rule execution
- Explain rule execution modes and execution principles

*Figure 4-40. Unit summary*

## Review questions

1. The rule engine can use these modes of execution. Choose all that apply.
  - A. RetePlus
  - B. Fastpath
  - C. Sequential
2. True or False: The default rule engine mode is RetePlus.
3. True or False: The rule engine uses pattern matching to test rule conditions against objects that are passed from the application.



Figure 4-41. Review questions

Write your answers here:

- 1.
- 2.
- 3.

## Review answers (1 of 2)

1. The rule engine can use these modes of execution. Choose all that apply.
  - A. RetePlus
  - B. Fastpath
  - C. Sequential

The answer is A, B, and C.
2. True or False: The default rule engine mode is RetePlus.  
The answer is False. The default rule engine mode is Fastpath.
3. True or False: The rule engine uses pattern matching to test rule conditions against objects that are passed from the application.  
The answer is True.



Figure 4-42. Review answers (1 of 2)

---

# Unit 5. Developing object models

## Estimated time

01:00

## Overview

In this unit, you learn how to design the object models upon which rules are written and executed, and how to create the vocabulary that is required to author business rules.

## How you will check your progress

- Review
- Exercises

## Unit objectives

- Describe the association between the BOM and the vocabulary that is used in rules
- Define the XOM
- Work with BOM-to-XOM mapping
- Use refactoring tools to maintain consistency between the BOM and XOM

Figure 5-1. Unit objectives

## Topics

- Designing the models
- Defining business and execution object models
- Editing the business object model
- Mapping the BOM to the XOM
- Working with the vocabulary
- Refactoring

## 5.1. Designing the models

## Designing the models

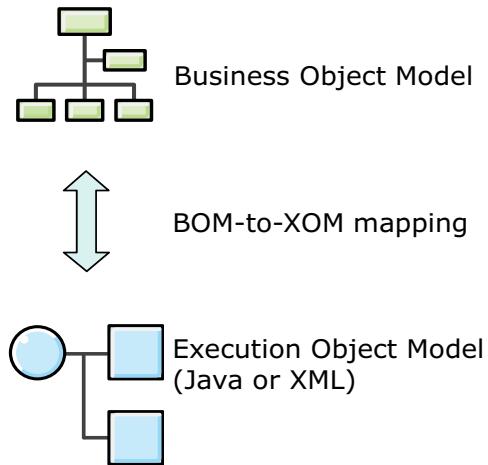
Developing object models

© Copyright IBM Corporation 2019

*Figure 5-3. Designing the models*

## Introduction to the models

- The business object model (BOM) and the execution object model (XOM) are two related object models
- The BOM is the model that defines the entities that are used in business rule artifacts
- The XOM is the model against which rules are executed
- The BOM is mapped to the XOM at run time



*Figure 5-4. Introduction to the models*

The business object model (BOM) and the execution object model (XOM) are related models.

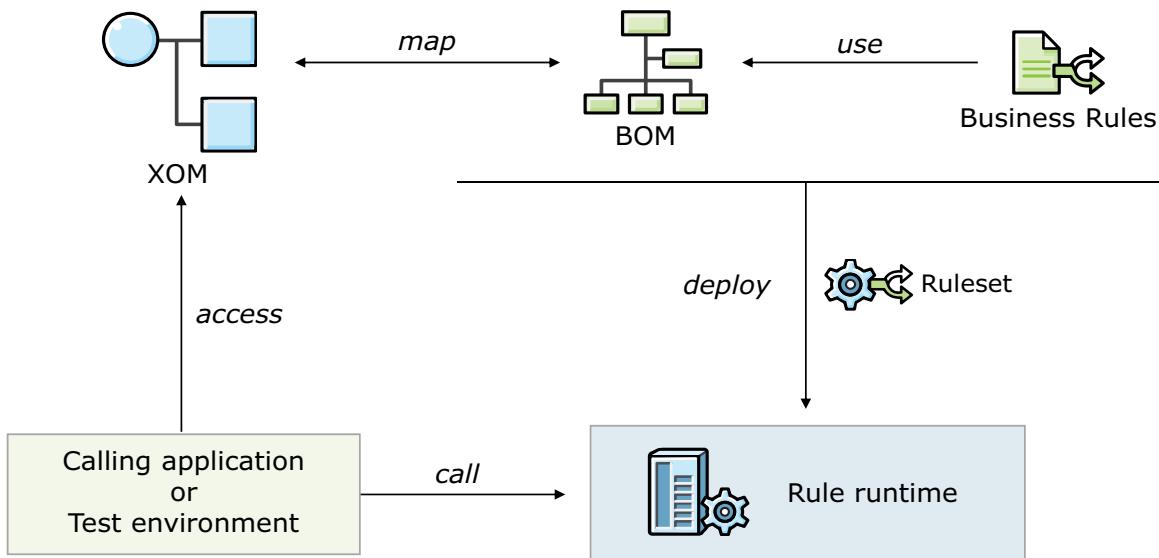
The BOM defines the entities that are used in rule artifacts, and is the model against which rules are written.

The XOM is the model against which rules are executed, and is used to interpret the BOM for rule execution.

For rule execution to work properly, all BOM elements that are used in the rules must map to XOM elements.

## BOM and XOM at run time

- The XOM references the application objects and data
- Through the XOM, the rule engine can access application objects and methods



Developing object models

© Copyright IBM Corporation 2019

Figure 5-5. BOM and XOM at run time

Through the XOM, the rule engine can access application objects and methods, which can be Java objects, XML data, or data from other sources. At run time, rules that were written against the BOM are run against the XOM.

## Designing the BOM and the XOM

- The BOM and the XOM are designed according to the requirements of the business team
- Top-down
  - Start by using modeling tools in Decision Center Business console to build the model and author the business logic
- Bottom-up
  - Start by first implementing the XOM in Rule Designer
  - After the XOM is implemented, you can automatically generate the BOM, which becomes the source of vocabulary in the rule editors
  - Business users author the business logic
- Development of the BOM and XOM can take several iterations

*Figure 5-6. Designing the BOM and the XOM*

The two approaches to designing the BOM and XOM are: bottom-up and top-down. ODM fully supports both approaches.

For developers, the starting point is based on models that the business analysts provide as vocabulary requirements. Generally, you use those requirements to implement the code for the XOM first, then generate the BOM automatically.

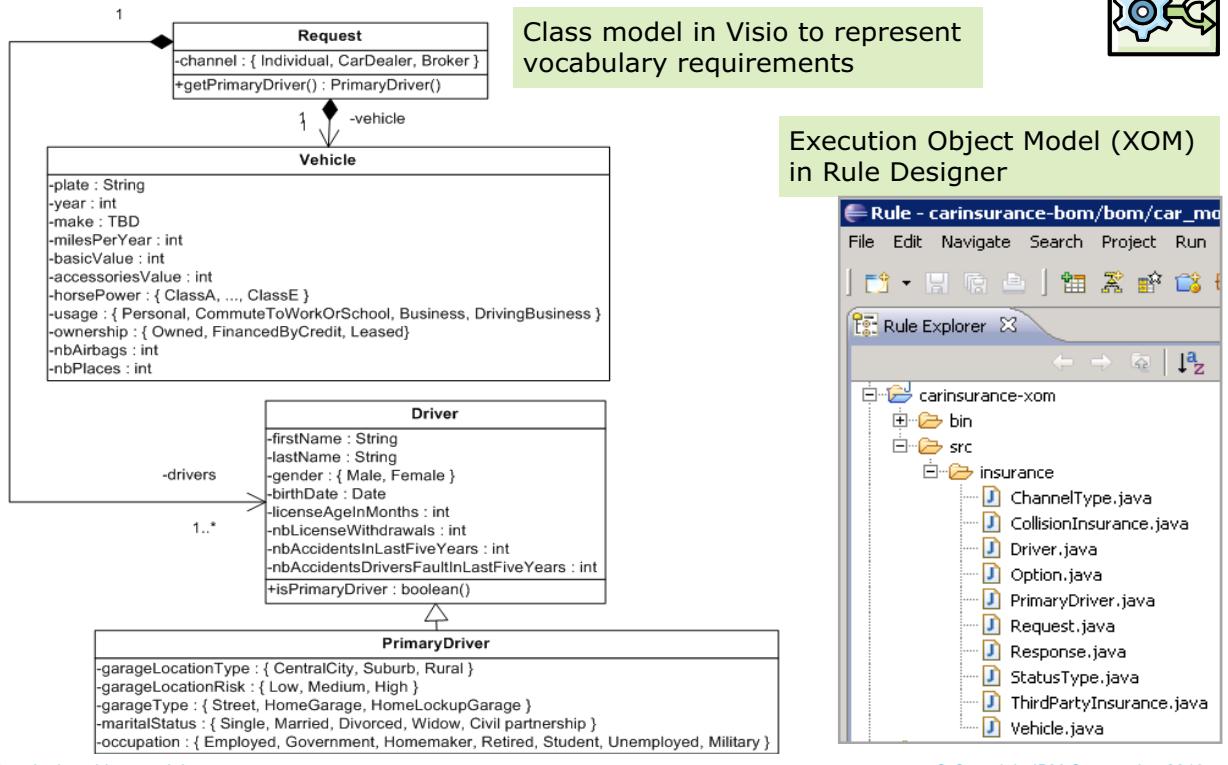
As you learned in the [Unit 3, "Modeling decisions"](#), business users can use a top-down approach to model decisions in Decision Center Business console. These decision models can be used directly as decision services or passed to the developers as requirements if more complex development is needed.

It's also possible to build the BOM first as a prototype rule creation and implement the XOM later. This requires some manual mapping of the BOM to the XOM.

With either approach, the BOM and the XOM evolve in parallel. Changes to the vocabulary require updates to the BOM and possibly the XOM, so as rule authors start working with the vocabulary as they write rules, you can expect several iterations before the BOM and XOM are stable.



## Example: Car insurance class diagram



Developing object models

© Copyright IBM Corporation 2019

Figure 5-7. Example: Car insurance class diagram

This slide shows an example of vocabulary that is modeled in a class diagram.

The developer can easily read these requirements and implement them in either Java or XML.

On the right, you see a screen capture of Rule Designer, and you can see that the classes from the Visio diagram are implemented in Java.

For example, on the left in the diagram, you can see the Driver class. On the right, you see the `Driver.java` file. If you open that file, you would see all the attributes and methods that are listed in the class diagram.

The business analysts might not always provide such a thorough model for vocabulary, but the more complete the model is, the smoother the implementation. The ODM BA class helps business analysts learn the basics of UML class diagrams so they can effectively communicate the vocabulary requirements to the developer.

**IBM Training**

**Example: Implementation of insurance model**

The screenshot shows the IBM Rule Designer interface. On the left, the 'Rule Explorer' view displays the XOM structure under 'carinsurance-xom'. A file named 'Driver.java' is selected and highlighted with a red box. On the right, the 'Outline' view shows the Business Object Model (BOM) generated for the 'Driver' class. The 'driver' node is highlighted with a blue box. Below it is a list of natural language phrases and their corresponding rule definitions.

**XOM in Designer**

**Business view of code is called Business Object Model (BOM) generated in Rule Designer**

```

driver
  @ {a driver} is primary driver
  @ {the birth date} of {a driver}
  @ {the first name} of {a driver}
  @ {the gender} of {a driver}
  @ {the last name} of {a driver}
  @ {the license age in months} of {a driver}
  @ {the number accidents} of {a driver}
  @ {the number license withdrawals} of {a driver}
  @ {the number of at-fault accidents in the last five years} of {a driver}
  → @ set the birth date of {a driver} to {a date}
  → @ set the first name of {a driver} to {a string}
  → @ set the gender of {a driver} to {a gender type}
  → @ set the last name of {a driver} to {a string}
  → @ set the license age in months of {a driver} to {a number}
  → @ set the number accidents of {a driver} to {a number}
  → @ set the number license withdrawals of {a driver} to {a number}
  → @ set the number of at-fault accidents of {a driver} in the last 5 years to {a number}
  @ {the age (in years)} of {a driver} at {a date}
  
```

Developing object models

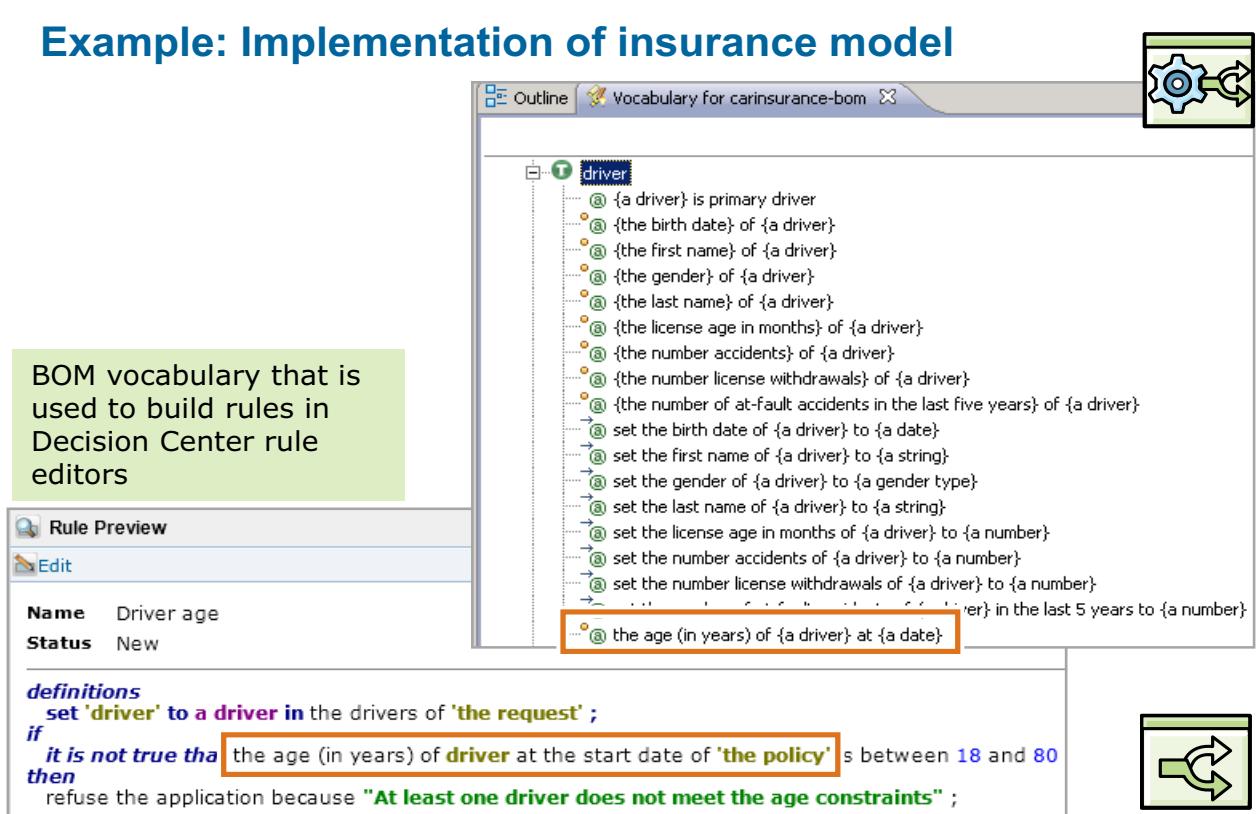
© Copyright IBM Corporation 2019

Figure 5-8. Example: Implementation of insurance model

Here, from the same `Driver.java` class, you can see the list of natural language phrases that business users can read and work with. From the XOM, Rule Designer can automatically generate a business view of the code that is called the Business Object Model (BOM), which includes a natural language vocabulary layer. The BOM vocabulary is what the business users see in the rule editors.

IBM Training 

## Example: Implementation of insurance model



**BOM vocabulary that is used to build rules in Decision Center rule editors**

**Rule Preview**

**Edit**

**Name** Driver age  
**Status** New

```

definitions
  set 'driver' to a driver in the drivers of 'the request' ;
if
  it is not true that the age (in years) of driver at the start date of 'the policy' is between 18 and 80
then
  refuse the application because "At least one driver does not meet the age constraints" ;

```

Developing object models

© Copyright IBM Corporation 2019

Figure 5-9. Example: Implementation of insurance model

When business users write rules, the rule editors prompt the authors with vocabulary from the BOM. Without the BOM vocabulary, rules cannot be written.

## 5.2. Defining business and execution object models

## Defining business and execution object models

Developing object models

© Copyright IBM Corporation 2019

*Figure 5-10. Defining business and execution object models*

## Business object model (BOM)

- The BOM is similar to a Java object model
  - Consists of BOM elements: packages, classes, class members (methods, attributes)
- The BOM is made of one or several BOM entries
- BOM entries are ordered with a BOM path
- You can associate a natural-language vocabulary with the BOM elements to make editing of business rules easier

Developing object models

© Copyright IBM Corporation 2019

Figure 5-11. Business object model (BOM)

Before you can write rules, you must set up a BOM that defines the objects that are described in your rule artifacts.

The BOM is similar to a Java object model, consisting of classes and class *members*. The members can be attributes and methods just like a regular Java class.

A single BOM can include one or several BOM *entries*, and each entry defines a set of business elements.

A BOM path comprises one or more BOM path entries and defines the order of the BOM entries.

You can associate a natural-language vocabulary with the BOM elements to make editing business rules easier.

## BOM entries

- Define a set of business elements in the BOM
  - Use multiple BOM entries to define a modular BOM
- Can be created manually, or automatically from the XOM
  - A BOM entry is attached to a single XOM source to simplify its refactoring on XOM changes
- Are composed of several files, with extensions `bom`, `voc`, and `b2x`



*Figure 5-12. BOM entries*

Each BOM entry is a group of several files, including:

- The `.voc` files, which are locale-specific and describe the vocabulary that is associated with the BOM
- A `.b2x` file, which describes the mapping between the BOM and the XOM
- A `.bom` file, which describes the structure of the BOM

When you create the XOM first, you can generate a BOM automatically from that XOM, which creates a direct correspondence between the two models. You can also later extend the BOM with modifying the original XOM. You can also create BOM entries and later map them to a XOM.

## BOM path

- A BOM is made of one or several BOM entries, which can be ordered with the BOM path
- If you have two business elements with the same name in two BOM entries, the one in the first BOM entry in the path overrides the other

Figure 5-13. BOM path

Your rule project uses BOM entries that are defined in its BOM. It can also reference BOM entries in other rule projects, through rule project references.

You can define a BOM path for your rule project to organize the BOM entries. A BOM path is similar to the class path that is used to organize classes in a Java application.

BOM classes are looked up according to the ordered list of BOM entries that are defined in the BOM path. If you have more than one BOM entries that use the same name for a BOM class, the first class in the BOM path is selected.

You can modify the order of the BOM entries in the BOM path to control this precedence.

When a project references other projects, the BOM path uses both the BOM path that is locally defined for this project and the BOM paths that are defined for the referenced projects.

The order in which the projects are referenced determines the order of the BOM paths.

## Execution object model (XOM)

- The XOM is the model against which rules are executed
- It references the application objects and data, and is the base implementation of the BOM
- The XOM must be set on the rule project to create BOM elements directly from XOM elements, and execute rules

Figure 5-14. Execution object model (XOM)

The execution object model (XOM) constitutes an abstraction of the physical models.

The XOM references the application objects and data, and is the base implementation of the BOM.

## Types of XOM

- You can build the XOM from different data sources, including:
  - Java classes (Java XOM)
  - XML schemas (dynamic XOM)
- You can create the XOM by defining a Java XOM, a dynamic XOM, or both:
  - To define a Java XOM, select Java projects or JAR files
  - To define a dynamic XOM, select XML schema files

Figure 5-15. Types of XOM

ODM supports two types of XOM:

- *Java XOM*, which is built from compiled Java classes
- *Dynamic XOM*, which is built from XML Schema Definitions (XSD)

The term *dynamic* means that no Java object is constructed or generated. Instead, dynamic XML objects are created to represent the instances of the dynamic classes.

## Rule project and XOM

- The XOM is required at run time
- Rule project must reference the XOM to be able to check the validity of your ruleset in runtime conditions
- You can manually set the rule project properties to reference the correct XOM

Figure 5-16. Rule project and XOM

Because the XOM is required at run time, your rule project must reference the XOM to be able to check the validity of your ruleset in runtime conditions.

During the exercise, you learn how to define the references to XOMs in your rule project.

## 5.3. Editing the business object model

In this topic, you learn how to edit the business object model.

## Editing the business object model

Developing object models

© Copyright IBM Corporation 2019

*Figure 5-17. Editing the business object model*

## Introduction

- After you create your BOM, you can edit each of its elements (classes, methods, attributes) in the BOM editor
- For example, you must edit the BOM elements to:
  - Define the vocabulary that is associated with the BOM elements and required to author the business rules
  - Guide or enhance how business rules can be authored, by adding domains and categories
  - Define data for tests and simulations

Figure 5-18. Introduction

After initial creation of the BOM, you can edit each of its elements in the BOM editor in Rule Designer.

During the exercises, you work with the features of the BOM editor. The next slides provide a quick tour of some of the screens that you work with.

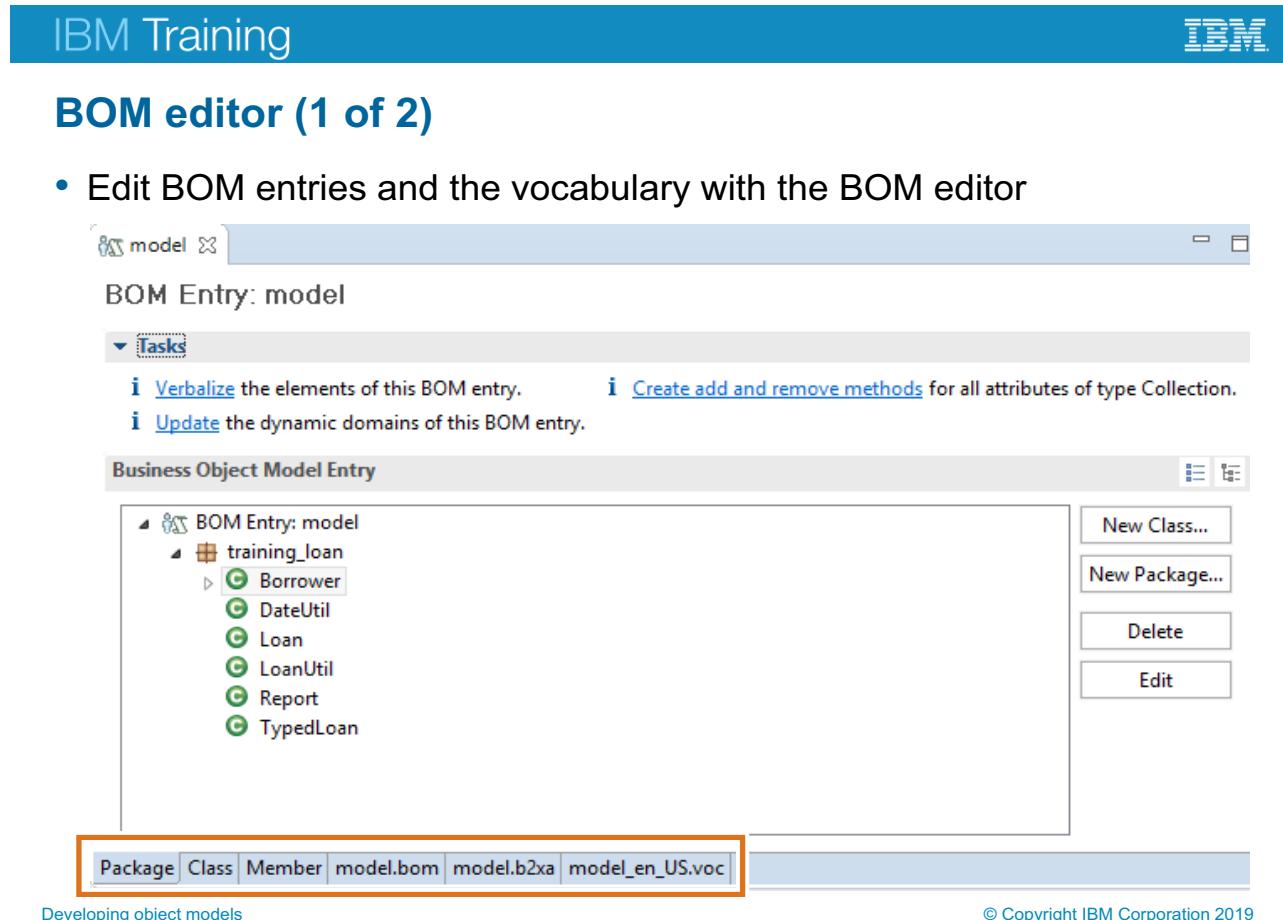


Figure 5-19. BOM editor (1 of 2)

To edit a BOM entry with the BOM editor, right-click the BOM entry in the Rule Explorer and click **Open With > BOM Editor**. Alternatively, you can double-click the BOM entry.

The BOM editor presents the information about the BOM element in multiple pages:

- On the **Package** page, you manage the structure of the BOM by manipulating packages and classes.
- On the **Class** page, you modify the information that is related to a class.
- On the **Member** page, you modify the information that is related to a member of a class.

During the exercises, you learn how to modify the BOM by editing the structure in the Package page, or class information in the Class page, and modifying attributes and methods on the Member page.

The screenshot shows the BOM editor interface for the class `Borrower`. The main title is "BOM editor (2 of 2)". The top navigation bar has tabs for "Class", "Member", "Relationship", and "Diagram". The "Class" tab is selected.

**General Information:**

- Name: Borrower
- Namespace: training\_loan
- Superclasses: java.lang.Object, java.io. (with a "Change..." button)
- Interfaces: (empty)
- Deprecated

**Class Verbalization:**

- Remove the verbalization.
- Generate automatic variable
- Term: borrower
- I the borrower, a borrower, the borrowers....

**Members:** Specify the members of this class.

- birthDate
- creditScore
- firstName
- lastName
- latestBankruptcyChapter
- latestBankruptcyDate
- latestBankruptcyReason
- spouse
- SSN

**Domain:** Create and edit a domain for this class.

**Categories:** Define the categories associated with this class.

**Custom Properties:**

Navigation bar at the bottom: Package, Class, Member, model.bom, model.b2xa, model\_en\_US.voc

Developing object models

© Copyright IBM Corporation 2019

Figure 5-20. BOM editor (2 of 2)

On the Class page and the Member page, you can edit the properties of the BOM element, such as the name and type of the element. The BOM editor is also where you define the vocabulary that is used in rules by verbalizing or assigning natural language terms to the BOM classes and members.

On the Class page, you can see all methods and attributes for that class. To edit attributes or methods, you can start from the Member section of the Class page, or click the **Member** tab.

Depending on what page of the editor you use, the available properties that you can edit vary. For example, in the General Information section, you can edit the definition of the BOM element itself.



## General information for methods

**Member Borrower (class: training\_loan.Borrower)**

**General Information**

|                                     |   |
|-------------------------------------|---|
| Name:                               | Borrower  |
| Type:                               | <input type="text"/> <a href="#">Browse...</a>              |
| Class:                              | training_loan.Borrower <a href="#">Browse...</a>            |
| <input type="checkbox"/> Deprecated | <input type="checkbox"/> Testing and simulation constructor |

**Arguments**

Edit the arguments of this member.

| Name | Type             | Domain |                           |
|------|------------------|--------|---------------------------|
| arg1 | java.lang.String |        | <a href="#">Add...</a>    |
| arg2 | java.lang.String |        | <a href="#">Remove...</a> |
| arg3 | java.util.Date   |        | <a href="#">Up</a>        |
| arg4 | java.lang.String |        | <a href="#">Down</a>      |
|      |                  |        | <a href="#">Edit...</a>   |

Developing object models

© Copyright IBM Corporation 2019

Figure 5-21. General information for methods

The General Information section for methods that are used as constructors also includes the **Testing and simulation constructor** check box. Constructors take arguments that are defined in the Arguments section. You assign meaningful names to these arguments so that business users can validate their values during testing and simulation.

In the General Information section, you can edit the definition of the BOM element itself.

The available properties depend on the type of the selected BOM element (class, method, or attribute). This section of the BOM shows the **Testing and simulation constructor** check box, which indicates that this element can be used as a constructor when generating the scenario templates for testing and simulation.

The Arguments section is only visible for a BOM method, and is used to manage the arguments of this method. You can add, remove, or edit an argument. You can also change the order of the arguments

IBM Training 

## General information for attributes

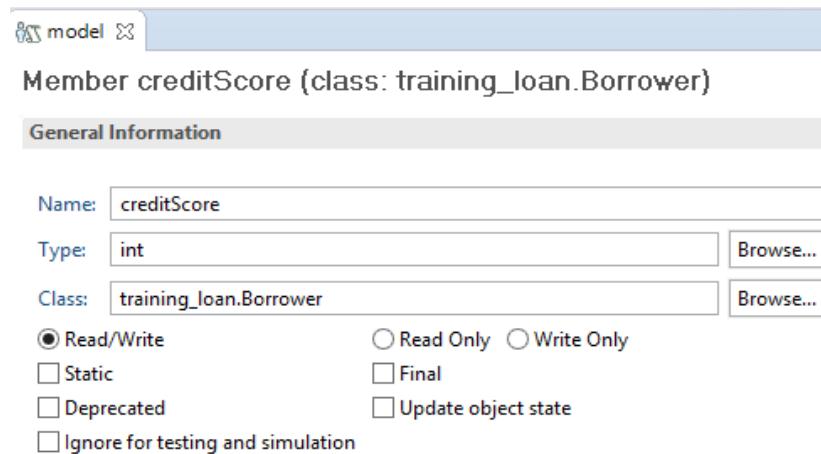


Figure 5-22. General information for attributes

When you define a BOM attribute, you see these fields in the editor.

The General Information section for BOM attributes includes these fields:

- **Name, Type, and Class:** These properties define the BOM element.
- **Read/Write, Read Only, and Write Only** (for an attribute): These properties indicate whether the business rule can get or set the value of the attribute, or both.
- **Static and Final** (for a method or an attribute): These properties have the same meaning as the corresponding Java keywords.
- **Deprecated** (for all): The **Deprecated** property indicates whether this BOM element is deprecated. As a result of this deprecation:
  - A warning is added in the Problems view for all rules that use this BOM element, indicating that this BOM element is deprecated.
  - The verbalization of this BOM element is underlined in yellow in the Intellirule editor.
  - This BOM element is not visible in the completion menus.
- **Update object state** (for a method or an attribute): The **Update object state** property of the BOM element indicates whether the rule engine is notified each time the state changes for an

object corresponding to this BOM element. The state can change either through a call to the method or by a direct update of the attribute.

This notification is required only by the RetePlus execution mode.

- **Ignore for testing and simulation:** By default, a scenario file template contains a column for each constructor argument and for each attribute (providing it is non-static and writable). This property indicates whether this BOM argument or attribute is excluded from the scenario file template.



### Note

The **Ignore for testing and simulation** and **Testing and simulation constructor** properties relate to enabling tests and simulations. You learn more about these features in [Unit 11, "Enabling tests and simulations"](#)

## 5.4. Mapping the BOM to the XOM

In this topic, you learn how the BOM and the XOM interrelate at run time. You also learn about BOM-to-XOM mapping.

## Mapping the BOM to the XOM

Developing object models

© Copyright IBM Corporation 2019

*Figure 5-23. Mapping the BOM to the XOM*

## What is BOM-to-XOM mapping?

- BOM-to-XOM mapping (B2X) is the correspondence between the BOM and the XOM
  - Defined in a BOM-to-XOM mapping XML schema
- Mapping translates BOM-based rule artifacts into XOM-based rule artifacts at run time
- Implicit mapping
  - The BOM element maps to the XOM element with the same fully qualified name
  - Implicit mapping is used by default
- Explicit mapping
  - Rule language mapping
  - Extender mapping: Using a Java extender class, with a static element with the same name as the BOM element

Figure 5-24. What is BOM-to-XOM mapping?

At run time, the *BOM-to-XOM (B2X) mapping* between the BOM and the XOM translates the rule artifacts in the BOM into rule artifacts that are based on the XOM.

At run time, the rule engine takes the rules, the BOM, the BOM-to-XOM mapping, and the XOM as input, and builds the internal structure that is required to process the objects of the application. With this structure, the rule engine can access application objects (for example Java objects or XML data) and methods. When searching for a XOM class, the rule engine searches in the dynamic XOM first, then in the Java XOM. So, rule artifacts that are written in terms of elements in the BOM are interpreted in terms of elements in the XOM.

The BOM-to-XOM mapping mechanism provides different types of mapping. The **default mapping** is where the BOM element maps to the XOM element with the same fully qualified name. Two explicit mappings are supported.

- **Rule language mapping**
- **Extender mapping**

## Rule language mapping

- Rule language mapping associates a business element with XOM-based rule language code
  - You can call functions, ruleset parameters and variables, and rule instances
- Includes:
  - Advanced Rule Language (ARL) mapping
  - ILOG Rule Language (IRL) mapping
- ARL
  - Used with the decision engine
  - Stored in a .b2xa file
- IRL
  - Used with the classic rule engine
  - Stored in a .b2x file
  - When migrating a project from the classic rule engine to the decision engine, the .b2x file is migrated to a .b2xa file

*Figure 5-25. Rule language mapping*

The Advanced Rule Language is designed for the decision engine to be used as an alternative way to create a BOM-to-XOM mapping. This rule language has a similar syntax to Java 7, with some rule-focused additions.

If your decision service project uses the decision engine, the BOM to XOM Mapping section allows you to use ARL. If the decision service used the classic rule engine (which is deprecated), the BOM to XOM Mapping section uses IRL. When you switch from the classic engine to the decision engine the first time, the B2X file is migrated automatically to a B2XA file.

Γ Note ——————

With the classic rule engine, business rules are converted to IRL before they can be processed by the rule engine. With the decision engine, the rules are fully compiled to intermediate code or Java bytecode. Although the decision engine does not depend on IRL for rules, you can use IRL for BOM-to-XOM mapping with both the decision engine and classic rule engine.

For more information about the decision engine and ARL, see the product documentation.

## Use of explicit mappings

- Add a BOM virtual member without touching the XOM
- Test instances of an execution class
- Quickly prototype the BOM on an existing XOM
- Rebind the BOM on a modified XOM to avoid modifying the rules
- At run time, the rule engine searches for the right mapping in this order:
  - Explicit mapping in rule language
  - Explicit extender mapping
  - Implicit mapping (default, when nothing is specified)

*Figure 5-26. Use of explicit mappings*

B2X mapping provides a way to create classes, attributes, or methods on your business model without impacting the XOM.

You customize the BOM-to-XOM mapping by using an explicit mapping for the reasons that are listed on the slide.

At run time, the rule engine uses the following order to find the right mapping:

- Explicit IRL mapping
- Explicit extender mapping
- Implicit mapping

If the rule engine does not find any mapping, an error is raised.



### Note

#### IBM ODM on Cloud

ODM on Cloud does not support B2X methods that make outbound calls to a database or a service.



## Create a method using ARL mapping code

New method on Loan class

**Member getLoanAmount (class: miniloan.Loan)**

**General Information**

|  |  |           |
|--|--|-----------|
| Name: <input type="text" value="getLoanAmount"/> | Type: int                                    | Browse... |
| Class: miniloan.Loan                             | Browse...                                    |           |
| <input type="checkbox"/> Static                  | <input type="checkbox"/> Final               |           |
| <input type="checkbox"/> Deprecated              | <input type="checkbox"/> Update object state |           |

**Vocabulary**

- Member Verbalization
  - ✖ Remove the verbalization.
  - + Create a navigation phrase.
  - >Edit the subject used in phrases.
- Navigation : "the loan amount of a loan" ✖
  - Template:

**Arguments**

**B2X ARL method**

**BOM to XOM Mapping**  
Edit the mapping between this BOM member and the XOM.

**Edit the imports**

**Body (in ARL)**

```
1 return amount;
```

Developing object models

© Copyright IBM Corporation 2019

Figure 5-27. Create a method using ARL mapping code

You use rule language mapping code to access the engine, ruleset variables, parameters, and rules directly from the BOM Editor.

In the example on this slide, the new getLoanAmount method is defined on the Loan class, and a default verbalization is created. The code is defined in the **Body** field of the **BOM to XOM Mapping** section in Advanced Rule Language (ARL):

```
return amount ;
```

## Testing instances of an execution class (1 of 2)

- Use B2X Tester to test instances of execution classes by mapping a business class to execution class
- To map a business class to an execution class:
  1. Specify the name of the execution class in the BOM editor
  2. Define the execution class in the **Execution name** field of the **BOM to XOM Mapping** section of the BOM Editor
  3. Define a tester, for complex mapping, to test instances of the execution class
    - When you run the rules, the execution class is used instead of the business class whenever needed
- For best results, apply the following rules to the mapping:
  - When the business class is a utility class, map it to `void`
  - When the business class is an enumerated class, do not provide a tester
  - When you do provide a tester, write it carefully so that it filters out all non-matching class instances, by returning `false`

*Figure 5-28. Testing instances of an execution class (1 of 2)*

B2X mapping can be used to test the instances of an execution class. You can define a business class that corresponds to an execution class, and use the B2X tester method to test instances.

By default, the elements of the business class are mapped to the elements of the execution class. The BOM-to-XOM mapping helps to deduce the business class from a particular class instance, such as for testing and simulation.

## Testing instances of an execution class (2 of 2)

- If members of the business class are not explicitly mapped, the BOM-to-XOM mechanism assumes that they are the same as the members of the execution class
- The following table describes how the BOM-to-XOM mechanism uses these members:

| Member of business class<br>BusinessClass                      | Case  | Mapping to execution member of<br>class ExecutionClass |
|--|---|--|
| Constructor<br>BusinessClass(MyBizClassB,<br>MyBizClassC)      | Call by new   | Constructor<br>ExecutionClass(MyClassB, MyClassC)      |
| Attribute MyBizClassA attr                                     | Assignment  | Attribute attr (not read-only)                         |
|  | Access  | Attribute attr (not write-only)                        |
| Method MyBizClassA<br>myBizMethod(MyBizClassB,<br>MyBizClassC) | Invocation  | Method MyClassA<br>myMethod(MyClassB, MyClassC)        |
| BusinessClass is used  | Use of operator<br>InstanceOf, or cast,<br>or classification in<br>conditions | ExecutionClass   |

Figure 5-29. Testing instances of an execution class (2 of 2)

On this slide, you see how the B2X mechanism maps business class members.

## Example: Mapping a business class to an execution class (1 of 2)

- Consider the Miniloan decision service project, which includes these business classes:
  - Borrower
  - Loan
- The `miniloan-xom` includes two execution classes:
  - Borrower
  - Loan
- Without modifying the Java XOM, you can create another business class on the BOM, **BigLoan**, that you can use to distinguish large loans from other loans
  - Business class **BigLoan** corresponds to execution class **Loan**
  - **BigLoan** includes an ARL method as Tester:

```
return amount > 100000000 ;
```

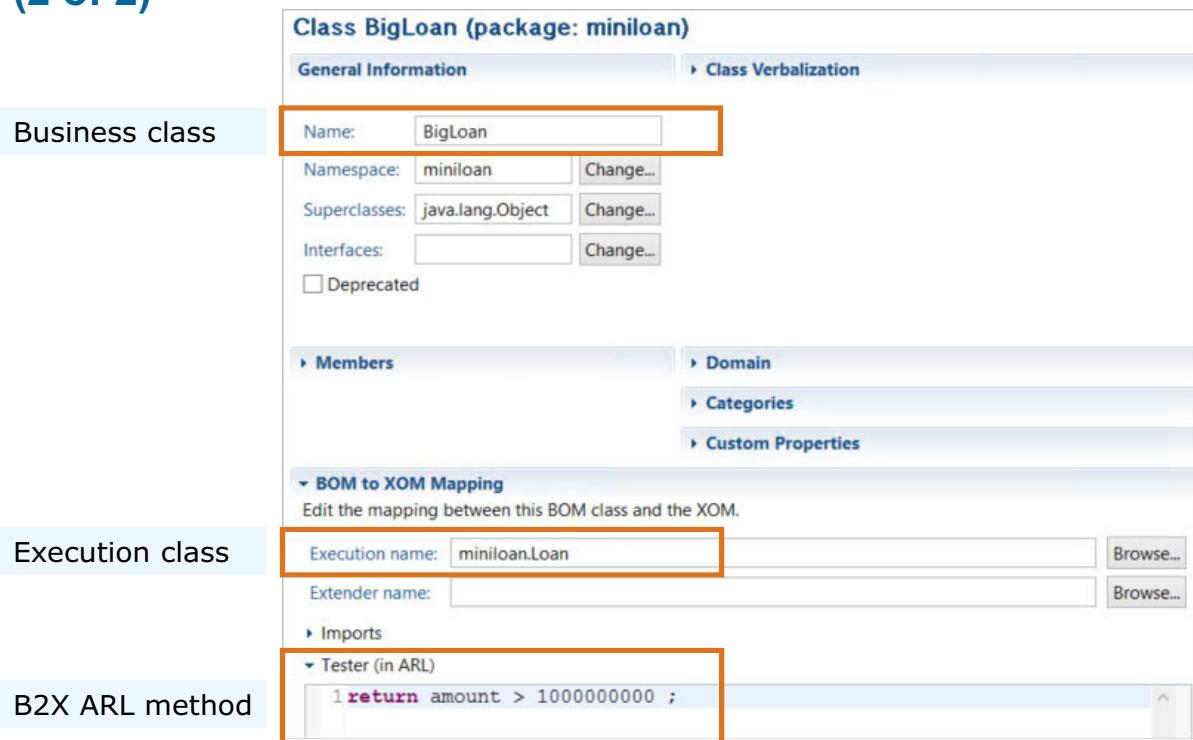
*Figure 5-30. Example: Mapping a business class to an execution class (1 of 2)*

To illustrate how you map a business class to an execution class, let's look at the example here with the Miniloan decision service.

The XOM includes the Borrower and Loan execution classes. If you needed to create another business class, but you didn't want to modify the XOM, you can simply create the class on the BOM, and map it to the XOM class. In this case, you create a specialized Loan class called BigLoan with a B2X method in ARL.

IBM Training 

## Example: Mapping a business class to an execution class (2 of 2)



The screenshot shows the BOM editor interface with three main sections:

- Business class**: A panel on the left containing the class definition.
- Execution class**: A panel in the center showing the mapping configuration.
- B2X ARL method**: A panel at the bottom containing the B2X ARL code.

**Business class (Top Left):**

- Name: BigLoan
- Namespace: miniloan
- Superclasses: java.lang.Object
- Interfaces: (empty)
- Deprecated

**Execution class (Center):**

- Execution name: miniloan.Loan
- Extender name: (empty)

**B2X ARL method (Bottom Right):**

```
1 return amount > 1000000000 ;
```

The fields for the execution class and the B2X ARL method are highlighted with orange boxes.

Developing object models

© Copyright IBM Corporation 2019

Figure 5-31. Example: Mapping a business class to an execution class (2 of 2)

In the BOM editor, you specify the execution class that you are mapping to, and you define the B2X method in the **Tester** section.

## Explicit extender mapping

- Designed to use Java rather than rule language
- In a Java extender class, you create static elements that have the same name as the BOM element
- Define an extender class name for a class in the **Extender name** field of the **BOM to XOM Mapping** section of the BOM Editor
  - The BOM-to-XOM mapping mechanism then looks up extender elements that have the same name as your business elements in the extender class

*Figure 5-32. Explicit extender mapping*

To create an explicit BOM extender mapping, define a static member in the XOM that has the same name as the member in the BOM.

To map a business element to the XOM by using extender mapping:

1. In the Outline view, click the class that contains the business element that you want to map.
2. In the BOM editor, in the **BOM to XOM Mapping** section, specify the name of your extender class in the **Extender name** field.
3. Define the extender class to provide the mappings for all members of the BOM class.

For example, say that your BOM has a business class that is named `ShoppingCart`, and you must map it to an execution class called `Cart` in the XOM. To do this mapping, select the `ShoppingCart` BOM class and specify `Cart` as its **Extender name** in the BOM editor. The `Cart` class in the XOM must define static members with the same names as the members of the `ShoppingCart` BOM class.

## 5.5. Working with the vocabulary

In this topic, you learn how to set up the vocabulary through verbalization.

## Working with the vocabulary

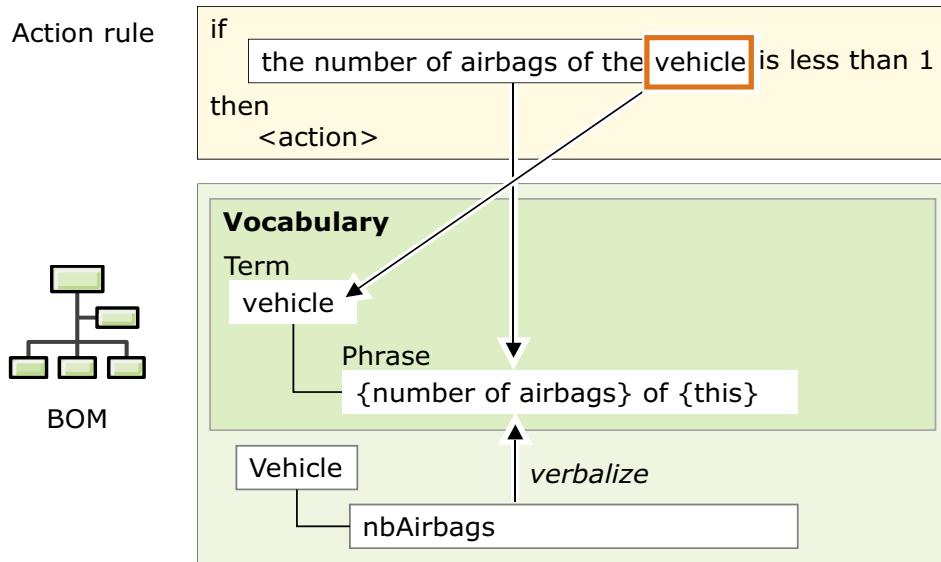
Developing object models

© Copyright IBM Corporation 2019

*Figure 5-33. Working with the vocabulary*

## Rule vocabulary

- Vocabulary in the rules comes directly from the BOM
- When you write rules in the rule editors, the vocabulary that you select is a set of terms and phrases that are attached to BOM members



Developing object models

© Copyright IBM Corporation 2019

Figure 5-34. Rule vocabulary

Generally, rule authors do not work directly with the BOM, so the default names of a BOM class or member might not be meaningful to them when they are authoring rules, especially when they are not familiar with the syntax of programming languages.

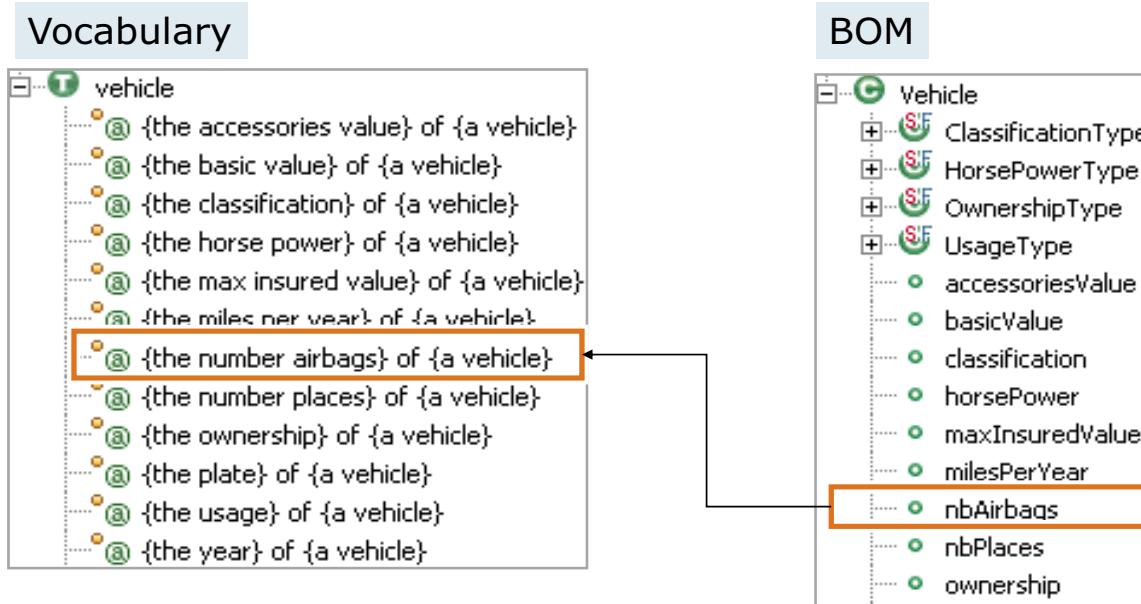
To facilitate rule authoring, you *verbalize* the BOM by attaching natural-language *vocabulary* to its classes, methods, and attributes. The set of *terms* and phrases that are attached to the elements of the BOM becomes the vocabulary that is used in the rule editors.

Verbalization makes it possible for business users to author the rules in natural language instead of programming language.

For example, on this slide, the rule uses the phrase "the number of airbags of the vehicle" instead of "vehicle.nbAirbags".

## Vocabulary and verbalization

- Attach meaningful terms to BOM classes, methods, and attributes



Developing object models

© Copyright IBM Corporation 2019

Figure 5-35. Vocabulary and verbalization

After you verbalize the BOM, the vocabulary becomes available in the rule editor selection lists.

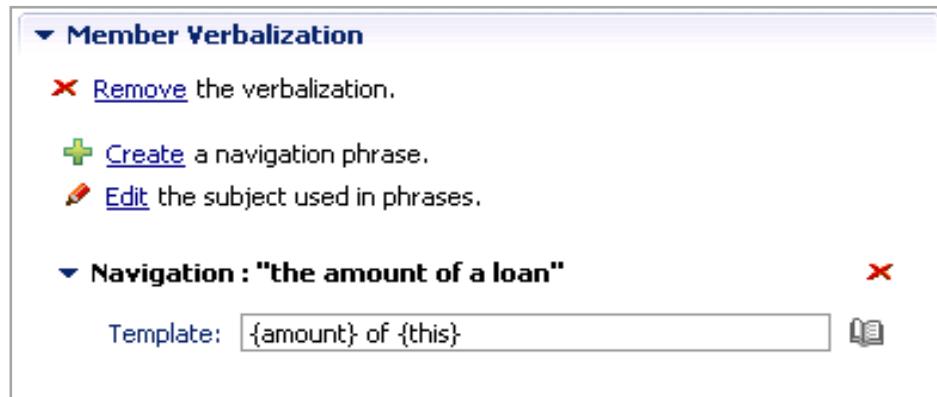
Rule Designer can generate a default verbalization for you when you create the BOM. You must review the default vocabulary to make sure that it makes sense and uses easily understood terms.

Keep in mind that this vocabulary layer is used for BAL rules. Later, when you learn about technical rules that are written in IRL, you see the difference in vocabulary that is used.



## Verbalization in the BOM editor

- Depends on the selected BOM element
- Example of a BOM attribute



*Figure 5-36. Verbalization in the BOM editor*

You edit the vocabulary for the BOM elements in the Verbalization section of the BOM editor.

In the BOM editor, you manage the verbalization of:

- A BOM class in its Class Verbalization section
- A BOM member in its Member Verbalization section

The options in the Verbalization section vary according to the BOM element that is selected. For example, the Member Verbalization section for the **amount** BOM attribute is shown here.

## Verbalization (1 of 2)

- Verbalizations are included in the vocabulary lists of rule editors
  - BOM must be defined and verbalized before you can start writing rules in the rule editors
  - Only verbalized business elements are accessible in the rule editors, and can be used in your business rules
- Rule Designer can provide a default verbalization of BOM elements
  - Review default vocabulary manually for clarity or customization

Developing object models

© Copyright IBM Corporation 2019

Figure 5-37. Verbalization (1 of 2)

Verbalized BOM members are listed as vocabulary elements in the rule editors. What about BOM elements that are not verbalized? Those elements do not show up in the vocabulary lists so they cannot be used to formulate BAL rules.

When you create a BOM entry, you can choose to create a vocabulary by default. A default verbalization is then applied to all members, getters, setters, and static references in the BOM entry. This default verbalization is not always relevant to the business policies, so you might need to manually review it and modify it to make sure that it is more business friendly.



### Hint

You can define several navigation or action phrases for a BOM element so that business users can use different wordings while using the same BOM element.

## Verbalization (2 of 2)

- Class:

- Class verbalizations are called terms
- Terms can be edited to correct plural form and articles
- Defaults typically require attention for irregular English-language nouns and non-English-language articles
- Examples of default verbalization:  
`LoanReport => loan report, loan reports`  
`branch => branch, branches`

- Members:

- Member verbalizations consist of subjects and phrases
- Defaults typically require attention on methods
- Examples of default verbalization:  
`getIncome() / setIncome() => income => the income of ...`  
`getYearlyIncome() / setYearlyIncome() => yearlyIncome => the yearly income of ...`  
`{this}.applyDiscount({0}) => apply a discount of {0} to {this}`

Developing object models

© Copyright IBM Corporation 2019

Figure 5-38. Verbalization (2 of 2)

On this slide, you see some of the terminology that is used to describe the vocabulary.

A vocabulary is composed of a set of business terms, phrases, and constants.

- A *business term* is the verbalization of a class.
- A *navigation phrase* is the verbalization of the getter of a member or a method that does not have a void return type.
- An *action phrase* applies an action to an object. It can be the verbalization of the setter of a member or a method that has a void return type.
- A *constant* is the verbalization of the public static final member of a class with same type as this class.

You see that these terms are used in the BOM editor.

## Placeholders

- Vocabulary phrase templates contain placeholders
- Placeholders represent gaps in phrases that can be completed automatically or manually when editing rules
- Braces identify the placeholders: {}

Figure 5-39. *Placeholders*



## Verbalizing BOM members

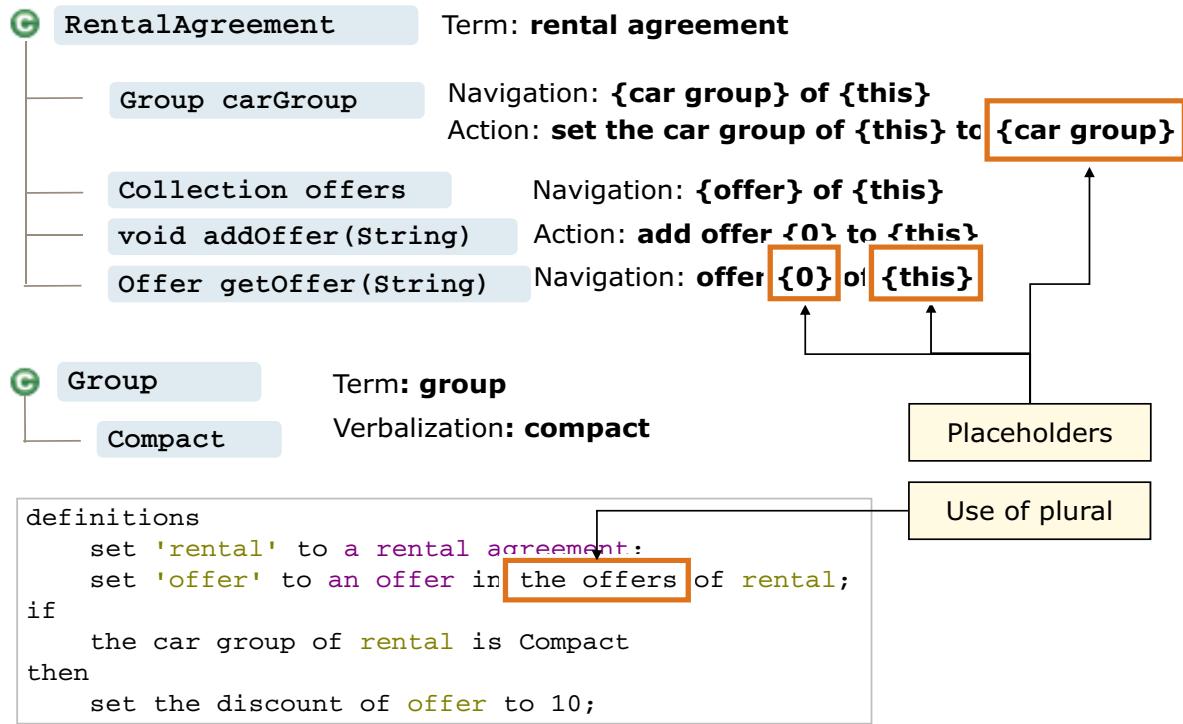


Figure 5-40. Verbalizing BOM members

Defining the vocabulary requires an understanding of business terms, phrases, and placeholders.

This slide shows examples of verbalization of BOM members so that you can recognize the relationship between the vocabulary that appears in the rule editors, and the BOM, and the final rule statement.

Notice how classes, methods, attributes, and constants are verbalized:

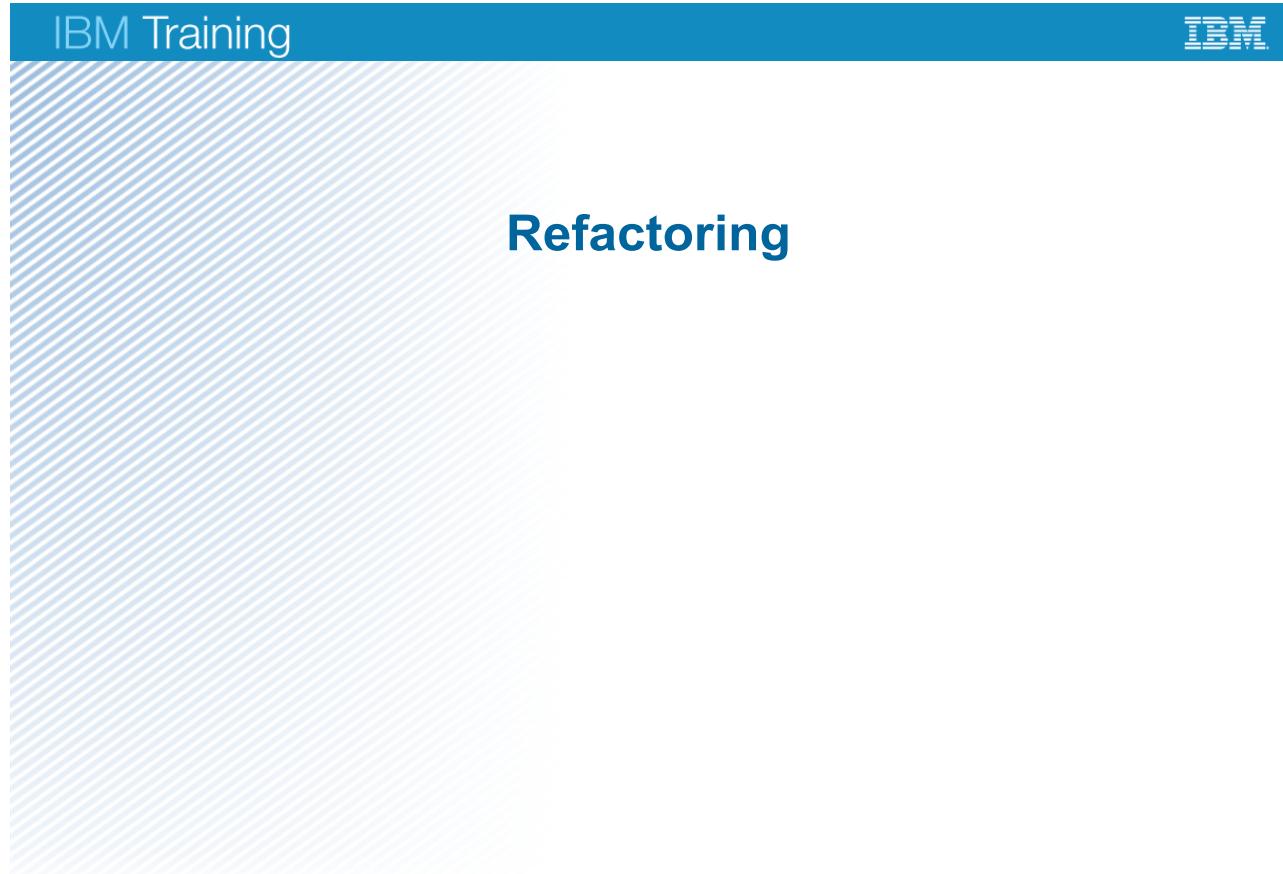
- The class **RentalAgreement** is verbalized with the term "rental agreement"
- The **carGroup** attribute of the class **RentalAgreement** (a **Group** object) is verbalized with two phrases:
  - The navigation phrase "{car group} of {this}" allows rules to read the value of this attribute
  - The action phrase "set the car group of {this} to {car group}" allows rules to write the value of this attribute
- The **offers** attribute of the class **RentalAgreement** (a **Collection** object) is verbalized with only the navigation phrase "{offer} of {this}"
- The **addOffer** method of the class **RentalAgreement** has a **String** argument, and returns **void**. It is verbalized with only the action phrase: "add offer {0} to {this}"

- The `getOffer` method of the class `RentalAgreement` has a `String` argument, and returns an `Offer`. It is verbalized with only the navigation phrase: "offer {0} to {this}"
- The class `Group` is verbalized with the term "group"
- The `Compact` constant object of the `Group` class is verbalized with the label "Compact"

After you define terms, you can use their plural forms, which are automatically defined for you. For example, "offers" is automatically available when you define "offer".

## 5.6. Refactoring

In this topic, you learn why and how you can keep the BOM and the XOM consistent with each other.



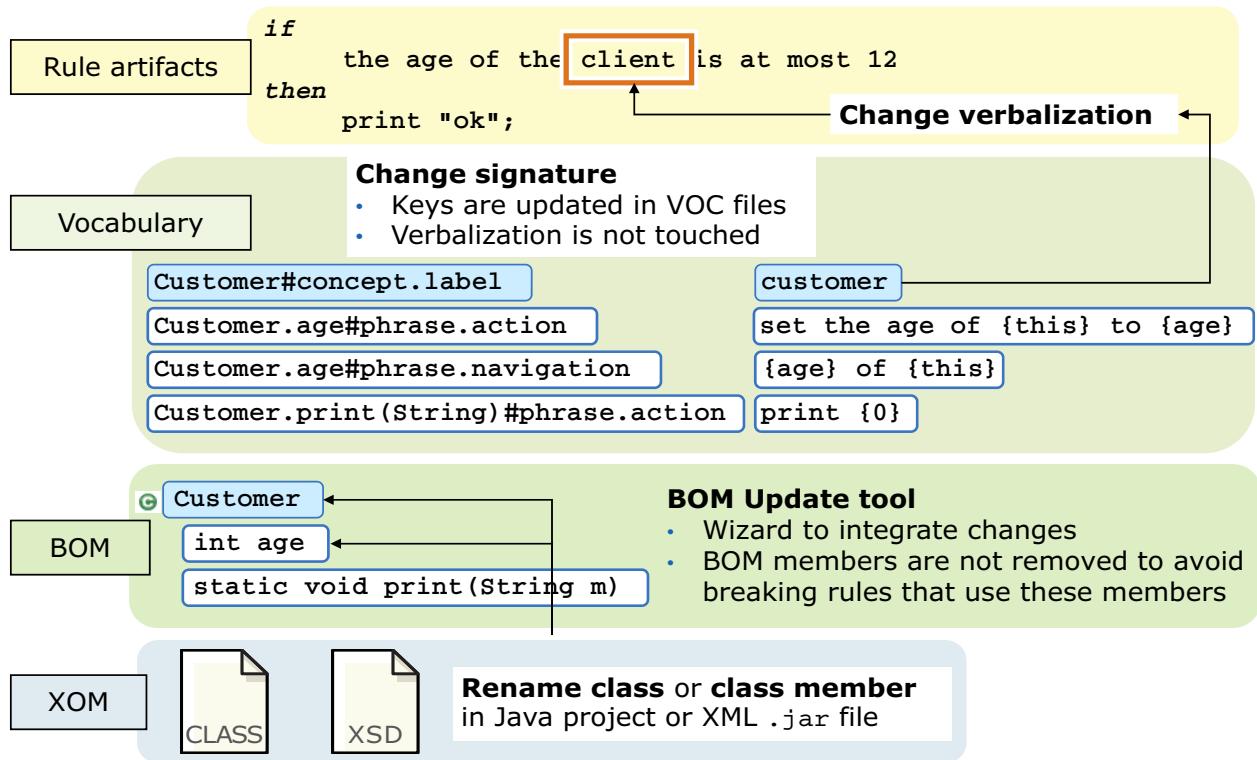
Developing object models

© Copyright IBM Corporation 2019

*Figure 5-41. Refactoring*



## Refactoring



Developing object models

© Copyright IBM Corporation 2019

Figure 5-42. Refactoring

The XOM, the BOM, and the vocabulary evolve as you and business users develop the business rule application. You can ensure that the rule project maintains consistency with such evolutions by *refactoring* the affected part of your rule project.

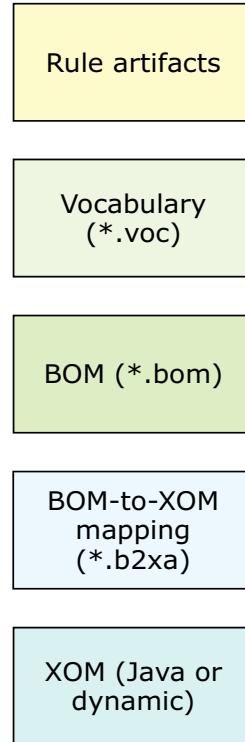
For example, you refactor when you change the verbalization of BOM members, the definition of BOM or XOM classes, or change the parameters or ruleset variables that are used in the rules.

Refactoring propagates the changes, while maintaining a valid state:

- XOM changes are propagated to BOM
- BOM changes are propagated to vocabulary files
- Vocabulary changes are propagated to rule artifacts

## From the XOM to the rules

- To understand the impact of evolution on rules, look at the various abstraction layers that separate the rules from the XOM
  - Rule artifacts
  - Vocabulary
  - BOM
  - BOM-to-XOM mapping
  - XOM



Developing object models

© Copyright IBM Corporation 2019

*Figure 5-43. From the XOM to the rules*

To understand the effects of change, you must understand the various abstraction layers that separate the rules from the XOM.

- The rules: Expressions of business logic that are written with business terminology
- The vocabulary: The terminology that is used for BOM elements in the rules
- The BOM: The abstract object model that represents the business view of the data
- The BOM-to-XOM mapping: How the BOM maps to the XOM
- The XOM: Where the rules execute



## BOM and XOM evolution

- As vocabulary requirements change, the XOM and the BOM members can change
- BOM-to-XOM mapping can shield the BOM from XOM changes within its operating range
- Vocabulary can protect the rules from changes in the BOM
  - Vocabulary refactoring propagates vocabulary changes to rules

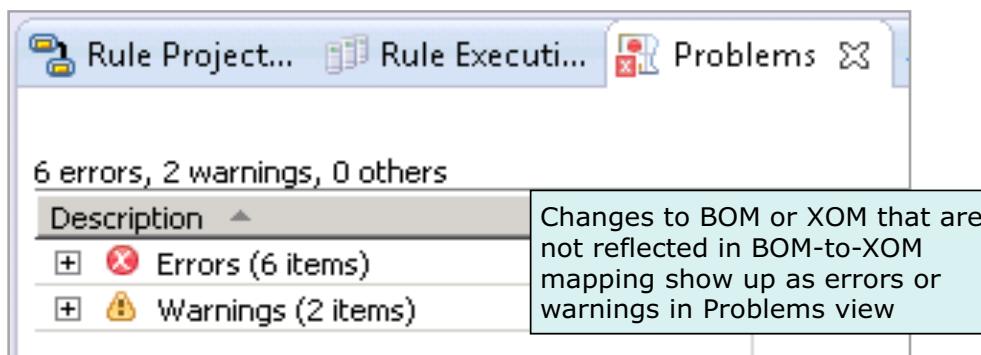


Figure 5-44. BOM and XOM evolution

The BOM-to-XOM mapping and the vocabulary, which sit between the XOM and the rules, shield the rules from many of the changes that are made either to the BOM or the XOM.

## XOM changes (1 of 3)

- If you use the Eclipse refactor menu to rename a Java class in the Java project, the BOM of the rule project is automatically refactored and the corresponding BOM class is renamed
  - The verbalization of the class is also changed
- The verbalization of the class is not changed when you rename only BOM elements

Figure 5-45. XOM changes (1 of 3)

If you use the Eclipse refactor menu to rename a Java class in a Java project that is used in your rule project, the BOM is automatically refactored and the corresponding BOM class is renamed. The verbalization of the class is also updated. The verbalization is not changed when you rename only BOM elements. Rule Designer changes only the indexing in the vocabulary files.

## XOM changes (2 of 3)

### Additions:

- If you add XOM elements, you can export them to the BOM when you use **BOM Entry > Update** to resynchronize the BOM with the XOM

### Removals:

- If you remove a XOM element, the corresponding BOM element is left without a XOM implementation
  - The BOM shows errors
  - You can fix it by mapping the BOM element to something else in the XOM or by also removing the BOM element

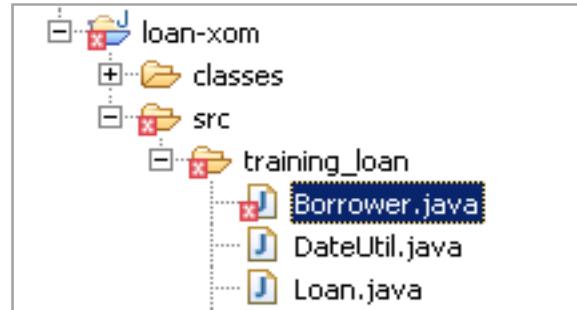


Figure 5-46. XOM changes (2 of 3)

If you add XOM elements, you can export them to the BOM when you resynchronize the BOM with the XOM by using the BOM with the BOM Update tool.

If you remove a XOM element, the corresponding BOM element is left without any implementation. When you resynchronize the BOM with the XOM, the BOM element is marked as deprecated. You can fix it by mapping the BOM element to something else in the XOM or also deleting the BOM element. However, such decisions require input from the business users.

## XOM changes (3 of 3)

- Renaming done manually: Consider such renaming as an addition or a removal, and map the BOM member name to the new XOM member name so as not to break existing rules
- Renaming is done through the refactor menu, with the BOM project open; the change is propagated to the BOM and the appropriate part of the vocabulary
- Other kinds of refactoring:
  - As much as possible, use **BOM Update** to propagate changes
  - Fix the rest manually

Figure 5-47. XOM changes (3 of 3)

When you manually rename a Java class, you should treat the renaming as an addition or a removal. You should then map the BOM member name to the new XOM member name so that existing rules do not break. By renaming through the refactor menu with the BOM project open, the change is automatically propagated to the BOM and the appropriate vocabulary.

## BOM changes

- Additions are mostly non-problematic
  - Unless the addition creates ambiguity in rules
- Removals can be problematic if the removed elements are used in rules
  - Consider deprecation instead of removal
- Renaming has no effect because the vocabulary hides the change
- Method signature changes generally break existing rules

Figure 5-48. BOM changes

When you change elements in the BOM, such as renaming an element, this change has no effect on the vocabulary because the vocabulary hides the change. This feature protects the rules so that they are not broken by the change. Even if the name of the BOM element changes, its associated phrases stay the same so the rules are not affected.

If you change the signature of a BOM method, this change might break existing rules because adding or removing parameters breaks verbalization, which in turn, breaks existing rules that use that method.

## Vocabulary changes

- If you change the verbalization of the BOM elements or variables, the rules that reference these elements can be automatically refactored
- A verbalization change is propagated to a rule that uses the term when the rule is:
  - In the current project
  - OR
  - In another open rule project of the same workspace that references the BOM project where the verbalization change occurs

Figure 5-49. Vocabulary changes

If you change the verbalization of the BOM elements or variables, the rules that reference these elements can be automatically refactored. After you save your change in the BOM editor, Rule Designer prompts you to specify whether you want to refactor the rules that use the business element to take your changes into account.

- If you accept, Rule Designer propagates the changes and maintains a valid state.
- If you decide not to refactor, the rules are not modified, and syntax errors are reported in the Problems view.

The changes are propagated to rules that are both saved and syntactically correct.

## Unit summary

- Describe the association between the BOM and the vocabulary that is used in rules
- Define the XOM
- Work with BOM-to-XOM mapping
- Use refactoring tools to maintain consistency between the BOM and XOM

Figure 5-50. Unit summary

## Review questions

1. True or False: The BOM is the business view of the model that defines the actions and entities that are used in business rule artifacts.
2. True or False: Before you can author rule artifacts, you must define the BOM, which is the source of vocabulary in the rule editors.
3. True or False: The XOM is the model against which rules are executed.
4. How do you define the translation of BOM elements into XOM elements?
  - A. As ruleset properties
  - B. As rule project properties
  - C. With the BOM editor by using B2X



Developing object models

© Copyright IBM Corporation 2019

Figure 5-51. Review questions

Write your answers here:

- 1.
- 2.
- 3.
- 4.

## Review answers (1 of 2)

1. True or False: The BOM is the business view of the model that defines the actions and entities that are used in business rule artifacts.  
The answer is True.
  
2. True or False: Before you can author rule artifacts, you must define the BOM, which is the source of vocabulary in the rule editors.  
The answer is True. Before you can create and edit rule artifacts, you must define a BOM to define the classes and methods that your rule artifacts act on.
  
3. True or False: The XOM is the model against which rules are executed.  
The answer is True.



Figure 5-52. Review answers (1 of 2)

## Review answers (2 of 2)

4. How do you define the translation of BOM elements into XOM elements?
  - A. As ruleset properties
  - B. As rule project properties
  - C. [With the BOM editor by using B2X](#)

The answer is C.



Figure 5-53. Review answers (2 of 2)

## Exercise: Working with the BOM

Developing object models

© Copyright IBM Corporation 2019

*Figure 5-54. Exercise: Working with the BOM*

## Exercise introduction

- Generate a BOM from an existing XOM
- Verbalize the BOM with natural-language vocabulary



*Figure 5-55. Exercise introduction*

## Exercise: Refactoring

Developing object models

© Copyright IBM Corporation 2019

*Figure 5-56. Exercise: Refactoring*

## Exercise introduction

- Refactor vocabulary changes
- Manage inconsistency issues after updating the XOM and BOM



Developing object models

© Copyright IBM Corporation 2019

*Figure 5-57. Exercise introduction*

# Unit 6. Orchestrating ruleset execution

## Estimated time

00:45

## Overview

This unit describes how to orchestrate rule execution through ruleflows. You also learn about rule engine execution modes.

## How you will check your progress

- Review
- Exercise

## Unit objectives

- Design ruleflows to organize the execution of the rule artifacts in a ruleset
- Configure how rules are selected for execution at run time
- Explain rule engine execution modes

Orchestrating ruleset execution

© Copyright IBM Corporation 2019

*Figure 6-1. Unit objectives*

This unit teaches you how to orchestrate rule execution through ruleflows. You also learn about rule engine execution modes.

## Topics

- Controlling rule execution: Overview
- Designing ruleflows
- Controlling rule selection for execution
- Controlling rule order during rule execution

*Figure 6-2. Topics*

## 6.1. Controlling rule execution: Overview

## Controlling rule execution: Overview

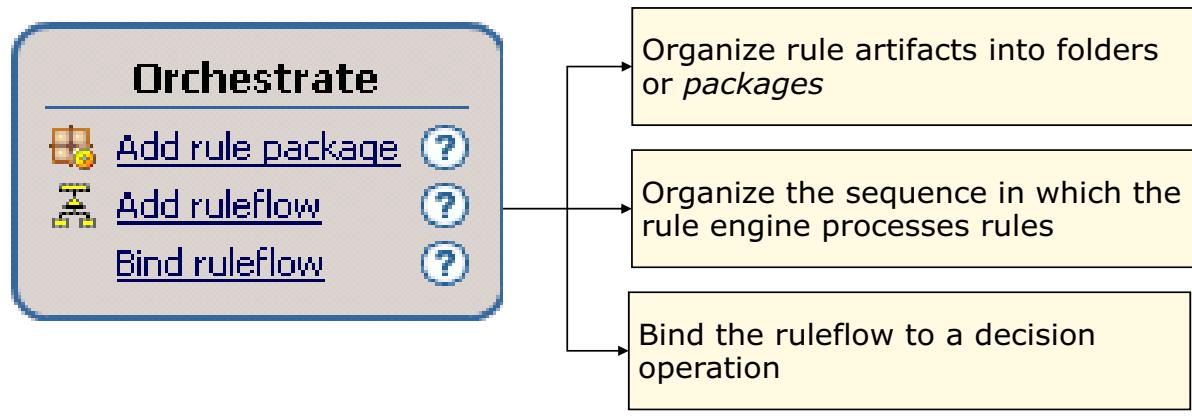
Orchestrating ruleset execution

© Copyright IBM Corporation 2019

*Figure 6-3. Controlling rule execution: Overview*

## What is orchestration?

- Rules implement business policy, but have no notion of sequence
- *Ruleflows* organize rules into a series of smaller decisions and define a routing logic to produce the overall decision
- Operation Map



Orchestrating ruleset execution

© Copyright IBM Corporation 2019

Figure 6-4. What is orchestration?

Rules are designed to produce decisions, but individually they have no notion of sequence. Therefore, the ruleflow is used to enforce the sequence in which rules are selected. It also defines which method the rule engine should use to evaluate those rules.

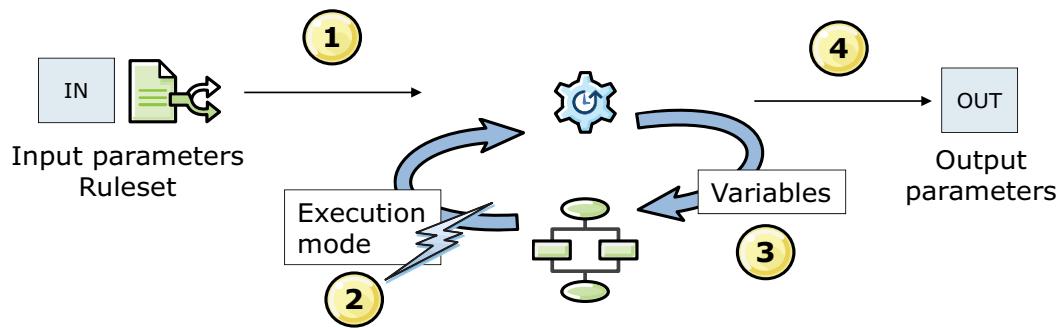
A ruleset is designed to make a single business decision. The individual rules in the ruleset can be evaluated as a series of smaller decisions that lead to the overall business decision.

A routing logic can determine the appropriate sequence of evaluation from a set of possible paths through the ruleset. You use the ruleflow to define the routing logic that determines the execution order of rule artifacts within the context of the ruleset.

## Key elements for orchestration

The ruleflow coordinates rule selection and execution within the ruleset

1. Parameters are passed (with the ruleset) to the rule engine
2. Based on information in the ruleflow, the rule engine knows which rules to select and what algorithm to use to evaluate the rules
3. During the evaluation, intermediate values or computations are stored in ruleset variables and can be transferred from one task to the next
4. The rule engine's evaluation results are returned to the calling application in a parameter



Orchestrating ruleset execution

© Copyright IBM Corporation 2019

Figure 6-5. Key elements for orchestration

To outline the flow of rule execution and produce the correct decision, you need to work with the business analysts and policy managers. Together you identify the type of information that is required to make the decision, and what type of information should be included as part of the decision result.

The key elements in your decision service that are required to orchestrate execution of your business rules include: ruleflows, rule packages, rulesets, parameters, and ruleset variables. You work with each of these artifacts during the exercise.

## 6.2. Designing ruleflows



Orchestrating ruleset execution

© Copyright IBM Corporation 2019

*Figure 6-6. Designing ruleflows*

You learned about the concepts that are required to orchestrate the execution of the rules in your decision service.

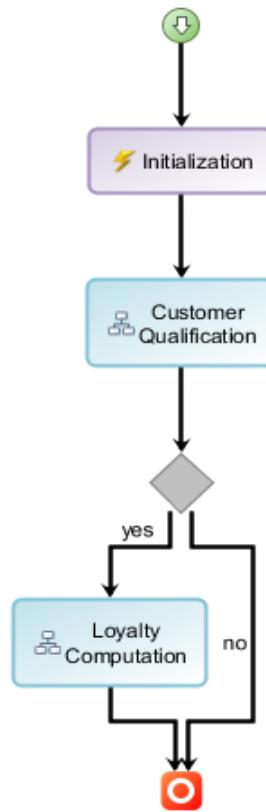
You now learn how to graphically design ruleflows and set their properties by using the Ruleflow Editor in Rule Designer.

# IBM Training



## Ruleflows

- Represent the business logic
- Sequences of rules that are grouped in tasks
- Organize rules into a sequence of decisions to produce a single decision



Orchestrating ruleset execution

© Copyright IBM Corporation 2019

Figure 6-7. Ruleflows

Here you see an example ruleflow. The ruleflow represents the application business logic. It defines the rule sequence and rule selection at a high level. In the ruleflow, you define smaller decisions as ruleflow *tasks*. You specify the *transitions* between the tasks and define the conditions under which these transitions are executed.

With the ruleflow, you do not have to write procedural code to control the flow of execution of your business rules. You construct ruleflows graphically in a Ruleflow editor and specify how tasks are chained together.

When objects are passed from the client application to the rule engine, the rule engine evaluates those objects against the rules in each task according to sequence defined in the ruleflow. Based on the results after evaluating one task, the routing logic determines which task to evaluate next. The final decision result, after the ruleflow completes, is then returned to the client application.

Objects can be passed to the rule engine as parameters or be accessed from the working memory. Within the ruleflow, intermediate results can be passed between tasks as ruleset variables.

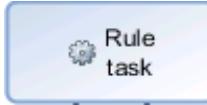
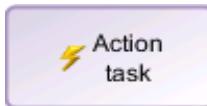
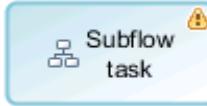
A decision service can contain several ruleflows. However, your ruleset must identify one ruleflow as the *main* ruleflow for the project. Additional ruleflows are accessed as subflow tasks.

Within the ruleflow, you can also define properties to control how the rules are selected and evaluated at run time.

## Ruleflow elements

- One start node  

- One or more end nodes  

- Tasks:
  - **Rule task:** Set of rules to be executed at that point in the ruleflow  
You can specify rule order, rule selection, and execution modes on each task  

  - **Action task:** Rule action statements in BAL or IRL code that are executed each time that the task is executed  

  - **Subflow task:** Reference to another ruleflow in the decision service or in a referenced rule project  

- Transitions connect the tasks in a ruleflow and define the sequence of the ruleflow from one task to another  


Orchestrating ruleset execution

© Copyright IBM Corporation 2019

*Figure 6-8. Ruleflow elements*

A ruleflow has one start node and one or more end nodes. It is composed of rules that are grouped into tasks. Transitions are used to connect tasks.

## Rule tasks

- A rule task should correspond to a single rule package
- Each rule in the rule task (package) is evaluated before the ruleflow moves to the next rule task
- Add or remove a rule in the package without modifying the ruleflow
  - All changes to the rule package are included the next time that the ruleset is deployed

Figure 6-9. Rule tasks

As a good practice, each rule task should correspond to a single rule package. When rule tasks include all the rules of a package, you can add or remove a rule in the package without modifying the ruleflow. All changes to the rules package are included the next time that ruleset is deployed.

## Initial and final actions

- Initial and final actions are optional
- At the ruleflow level
  - Initial actions: Specified in the start node and executed before the rest of the ruleflow
  - Final actions: Specified in the end nodes and executed after the rest of the ruleflow
  - The final actions are the same for all end nodes
- At the task level
  - Initial actions: Executed before the task body
  - Final actions: Executed after the task body

Figure 6-10. Initial and final actions

You can define a set of initial actions and final actions on a ruleflow or on an individual task within the ruleflow.

At the ruleflow level, initial actions that are specified in the start node are executed before the rest of the ruleflow begins. Final actions that are specified in the end nodes are executed after the full ruleflow is finished. If you have more than one end node, the final actions are the same for all end nodes.

At the task level, initial actions are executed before the task body, and final actions are executed after the task. Task execution consists of executing the initial actions, then its body, and then its final actions.

Initial and final actions are not mandatory and can be used independently of each other.

## Transitions

- Transitions
  - Unidirectional arrows that connect tasks and determine the “flow” of the ruleflow
- Transition conditions
  - Boolean condition for controlling the flow from task to task
- Can include an “**else**” condition
  - For all cases that do not match other conditions in the transition
  - Maximum one “**else**” per transition

Figure 6-11. *Transitions*

Transitions are directed arrows that connect the tasks and other components in a ruleflow.

For a single transition from one task to another, you specify a single transition arrow without conditions.

If you have a choice, and need to specify several transition arrows, for each transition, you must indicate under what conditions this transition can be selected. The transition conditions really define the routing logic through the ruleflow.

When multiple transitions that originate from the same task define overlapping conditions, the path that is taken to execute the ruleflow can be unpredictable. Make sure the conditions that you define for multiple transitions do not overlap.

## Forks and joins

- Use forks and joins to create multiple, parallel paths in your ruleflow
- A fork is a node that splits the execution flow into several parallel paths 
- A join is a node that combines all the paths that are created from a fork when the parallel paths are all completed 
- Although the rule engine executes transitions in a path sequentially, one after another, the order of execution of the different paths between a fork and a join is not certain

*Figure 6-12. Forks and joins*

The path that is taken through the ruleflow is a series of tasks, and transitions between these tasks, and might include a fork and a join.

The transitions that are created from a fork do not have conditions because the ruleflow follows all paths in parallel between the fork and the join.

Using a sequential or a parallel structure (with forks and joins) in the ruleflow makes no difference in terms of runtime performance or memory consumption.

## Expression of initial or final actions, and conditions

- Express initial actions, final actions, and transition conditions in:
  - Business Action Language (BAL)
  - ILOG Rule Language (IRL)
- Typical examples of actions:
  - Set up initial values of output parameters and variables
  - Check initial values of input parameters
- Typical examples of transition conditions:
  - Test values of parameters and variables

Figure 6-13. Expression of initial or final actions, and conditions

When you work with initial actions, final actions, and transition conditions in ruleflows, you can express them in Business Action Language (BAL).

Transitions can also be expressed in ILOG Rule Language (IRL).

During the exercise, you see how to use these actions and transition conditions.

## Main ruleflow

- You must indicate which ruleflow the rule engine must consider as the main ruleflow in your ruleset, that is, the one from which ruleset execution starts
- To define the main ruleflow:
  - In the Properties view of the ruleflow that you want to be the main ruleflow, set the **main flow task** property to: `true`
  - In the Properties view of each other ruleflow, set the **main flow task** property to: `false`

Figure 6-14. Main ruleflow

As mentioned earlier, you can have several ruleflows in your project, but you must define one as your main ruleflow. You define a main ruleflow in the Properties editor.

You learn how to define a main ruleflow during the exercise.

## 6.3. Controlling rule selection for execution

## Controlling rule selection for execution

Orchestrating ruleset execution

© Copyright IBM Corporation 2019

*Figure 6-15. Controlling rule selection for execution*



## Rule selection pipe

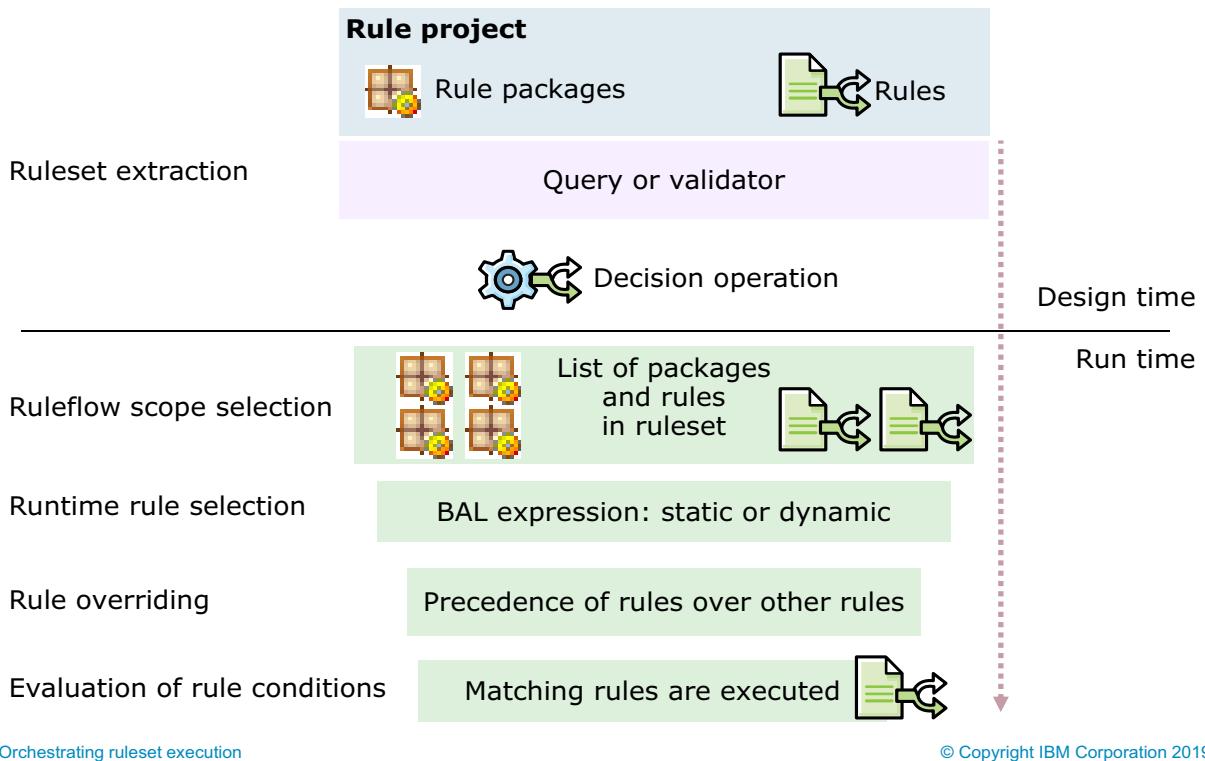


Figure 6-16. Rule selection pipe

After extracting the rule artifacts into a ruleset archive, you pass the archive to the rule engine for execution. The rule engine does the next steps at run time to select which rule artifacts to execute and in which order, by using *ruleflow scope selection*, *runtime rule selection*, *rule overriding*, and *rule conditions evaluation*.

You learn now how you can design your decision service to control how the rule engine does these steps on your rule artifacts.

## Ruleflow scope selection

- At run time, ruleflow scope selection generates the list of the rule packages and of the rules that each task defines
  - This list is the scope of the rule task
- When the rule engine executes the task, it considers only the rules within its scope

Figure 6-17. Ruleflow scope selection

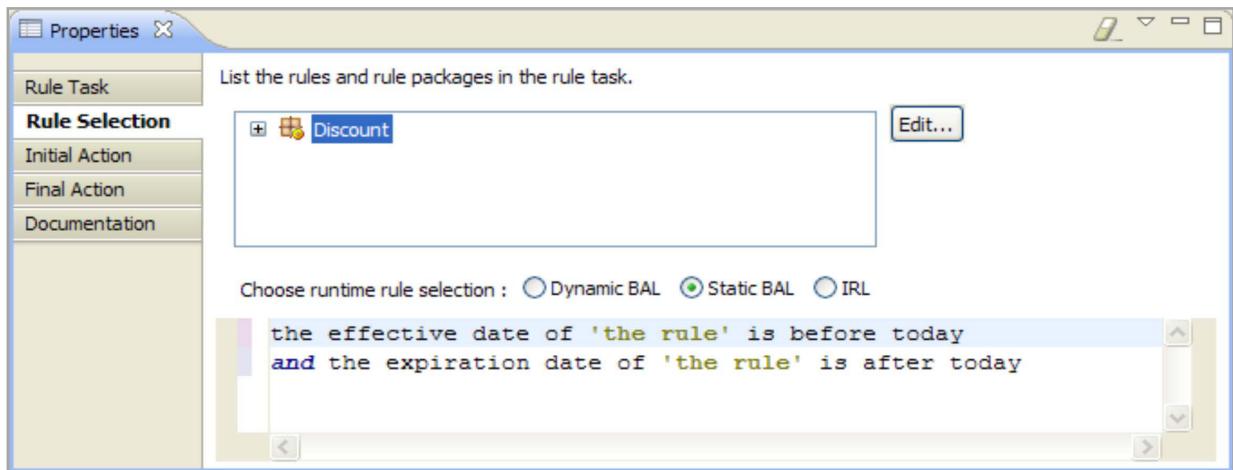
At run time, a list is generated of all the rule packages and rules that you defined for each ruleflow task. While the task is being evaluated, only the rules in the scope of the ruleflow task are considered for execution.

You define the *ruleflow scope* on the **Rule Selection** tab of the Properties view of the rule task. On this tab, you can edit and select the rule artifacts or rule packages that you want to include. You can further specify the order in which those artifacts are placed within the ruleset archive.



## Runtime rule selection

- **Dynamic BAL:** Each time the task is executed
- **Static BAL:** The first time the task is executed
- **IRL:** A rule filter in IRL with “body = . . .”



Orchestrating ruleset execution

© Copyright IBM Corporation 2019

Figure 6-18. Runtime rule selection

During task execution, you can further filter which rules to evaluate through *runtime rule selection*, which is a filter that is defined on the rule task. It constrains which rules from the task can be used. You can define this filter in either BAL or IRL. You define it on the **Rule Selection** tab of the Properties view for that rule task.

Runtime rule selection can be either dynamic or static. Your choice changes the way that the filter is applied to the rules. If you use static, the filter is called only at the first execution of the rule task. If you use dynamic, the filter is used each time the rule task is executed, so candidate rules are reselected if the state of an object was updated.

## Rule hierarchies

- Can be used to further refine rule selection
- Define a tree of values
- Are customizable
- Are typically combined with overriding

Figure 6-19. Rule hierarchies

You can further refine runtime rule selection by using specific BAL constructs that define rule hierarchies. To use hierarchies, you must extend the rule model. You do not work with hierarchies during this course, but you can find more information in the product documentation.

## Rule overriding

- Is typically used for local or specific rules to override general rules
  - A rule can override one or more other rules
- Is defined with the `overriddenRules` rule property
- Removes rules that other rules in the ruleset are overriding

Figure 6-20. Rule overriding

With rule overriding, you can assign rules precedence over other rules. For example, you can set one decision table to override another decision table.

When you override a rule, it means that the rule engine executes one rule instead of another. This type of overriding is often used together with rule hierarchies. For example, when considering locations, you might use a local rule to override a global rule, which means that the global rule is ignored and the local rule is executed instead.

## Rule condition evaluation

- Conditions of remaining rules are evaluated
- Only rules with matching conditions are fired

Figure 6-21. Rule condition evaluation

The final rule selection method is the evaluation of rule conditions. The other filters limit the scope of which rules the engine even looks at. When it comes to rule conditions, this filter is on the remaining rules that you *do* want the rule engine to evaluate. However, the engine itself filters which rules to execute by looking at the conditions in those rules. If the rule conditions do not match any of the objects that were passed from the application, the engine ignores that rule.

## 6.4. Controlling rule order during rule execution

## Controlling rule order during rule execution

Orchestrating ruleset execution

© Copyright IBM Corporation 2019

*Figure 6-22. Controlling rule order during rule execution*

## Introduction

- After the rules are selected, you can more finely control the order in which the rules are fired by setting more properties:
  - Rule priority
  - Rule task execution order
  - Execution modes

Figure 6-23. Introduction

You can set further properties on the business rules or on the rule tasks, like the execution mode. In this way, you can give finer control to the execution order among the business rules that are candidates for execution after rules selection.

## Rule priority

- With the **priority** rule property, you can specify that a rule has priority over other rules to execute if these rules have a lower priority
- You can also define dynamic priorities that are evaluated at run time
  - You define a dynamic property by setting the **priority** property to an expression rather than to a static value

Figure 6-24. Rule priority

You can assign a level of priority to individual rules to ensure that the rule engine evaluates them. Like the rule overriding property, rule priority is set at the rule level. Priority can be static or dynamic. For a static priority value, you set the priority property to an integer between -109 and 109. The larger the number, the higher the execution priority of the rule.

For dynamic priorities, you set them to an expression that evaluates to an integer at run time.

## Rule task execution order (1 of 2)

- Enforce a specific execution order by defining the ordering of rules in a rule task with properties:
  - **Ordering**
  - **Exit Criteria**
- **Ordering** property: To specify the order in which rules are executed in a rule task
  - **Default**: Depends on the execution mode that is selected
  - **Literal**: Order in which the rule task lists the rules
  - **Priority**: Sorts rules according to the execution mode in operation

Figure 6-25. Rule task execution order (1 of 2)

You can force a specific execution order by defining the ordering of rules within a task.

You define the rule order in the rule task by setting the **Exit Criteria** and **Ordering** properties.

You can set an **Ordering** property to one of these values. If you use **Default**, the ordering of rules depends on the execution mode. Or, you can choose **Literal**, in which case the rules are executed following the order in which they are listed in the task. If you use **Priority**, the rules are sorted according to the execution mode in operation.

## Rule task execution order (2 of 2)

- **Exit Criteria** property: Use to specify how rules are executed before the task terminates
  - **None**: All rules are executed until conditions terminate execution
  - **Rule**: The execution terminates after the chosen rule is executed
  - **RuleInstance**: Applicable for RetePlus and Fastpath only; a single instance of one rule is executed
- Combinations of these properties have different effects, depending on the applicable execution mode

Figure 6-26. Rule task execution order (2 of 2)

You can set an **Exit Criteria** property on a rule task to specify how rules are executed before the task terminates (all rules, one rule, or one rule instance).

You can set this property to **None**, which means all rules are executed. Or, you can set it to **Rule**, which means execution terminates after the chosen rule is executed, according to the algorithm. Or, you can set it to **RuleInstance**, which means a single instance of one rule is selected (according to the rule order) and executed. This value is applicable for RetePlus and Fastpath algorithms.

Combinations of these properties have different effects, depending on the applicable execution mode. For more information, see the product documentation.

## Execution modes

- Control how the rule engine processes rules by setting the execution mode for each rule task in a ruleflow
- Execution mode specifies the algorithm that the rule engine uses and the order in which rules in the task are evaluated
- The rule engine provides three execution modes:
  - Fastpath
  - RetePlus
  - Sequential
- Each ruleflow task can use a different execution mode
  - Choose the mode that is most appropriate for the type of rules in the rule task and according to the way the objects are passed to the rule engine
  - Default mode: Fastpath

Figure 6-27. Execution modes

The rule engine uses three algorithms: RetePlus, Sequential, and Fastpath.

You can use the ruleflow to define which algorithm to use. Within the ruleflow, you can further specify whether you want to use a different algorithm for a specific task. The algorithm changes the way the rule engine evaluates the rules.

## Unit summary

- Design ruleflows to organize the execution of the rule artifacts in a ruleset
- Configure how rules are selected for execution at run time
- Explain rule engine execution modes

Figure 6-28. Unit summary

## Review questions

1. True or False: A ruleflow is a graphical representation of a business decision.
2. True or False: A ruleflow has one start point, one or more end points, and groups rules into rule tasks, action tasks, or subflow tasks.
3. Rule tasks use which execution mode as the default?
  - A. RetePlus
  - B. Sequential
  - C. Fastpath



Figure 6-29. Review questions

Write your answers here:

- 1.
- 2.
- 3.

## Review answers

1. True or False: A ruleflow is a graphical representation of a business decision.  
The answer is True.
  
2. True or False: A ruleflow has one start point, one or more end points, and groups rules into rule tasks, action tasks, or subflow tasks.  
The answer is True.
  
3. Rule tasks use which execution mode as the default?
  - A. RetePlus
  - B. Sequential
  - C. FastpathThe answer is C.



Figure 6-30. Review answers

## Exercise: Working with ruleflows

Orchestrating ruleset execution

© Copyright IBM Corporation 2019

Figure 6-31. Exercise: Working with ruleflows

## Exercise introduction

- Describe the parts of a ruleflow
- Create a ruleflow
- Orchestrate rule selection and execution through the ruleflow



Orchestrating ruleset execution

© Copyright IBM Corporation 2019

*Figure 6-32. Exercise introduction*

---

# Unit 7. Authoring rules

## Estimated time

02:00

## Overview

This unit teaches you how to author rule artifacts that implement the business logic and policies of a business rule application.

## How you will check your progress

- Review
- Exercises

## Unit objectives

- Describe rule languages
- Use the various rule editors to author rule artifacts
- Define the objects that rule artifacts manipulate

Authoring rules

© Copyright IBM Corporation 2019

*Figure 7-1. Unit objectives*

This unit teaches you how to author rule artifacts that implement the business logic and policies of a business rule application.

## Topics

- From business policy to business rules
- Authoring action rules with BAL
- Authoring decision tables
- Advanced Rule Language (ARL)
- Technical rules
- Defining objects that are used in rules

Authoring rules

© Copyright IBM Corporation 2019

*Figure 7-2. Topics*

## 7.1. From business policy to business rules

## From business policy to business rules

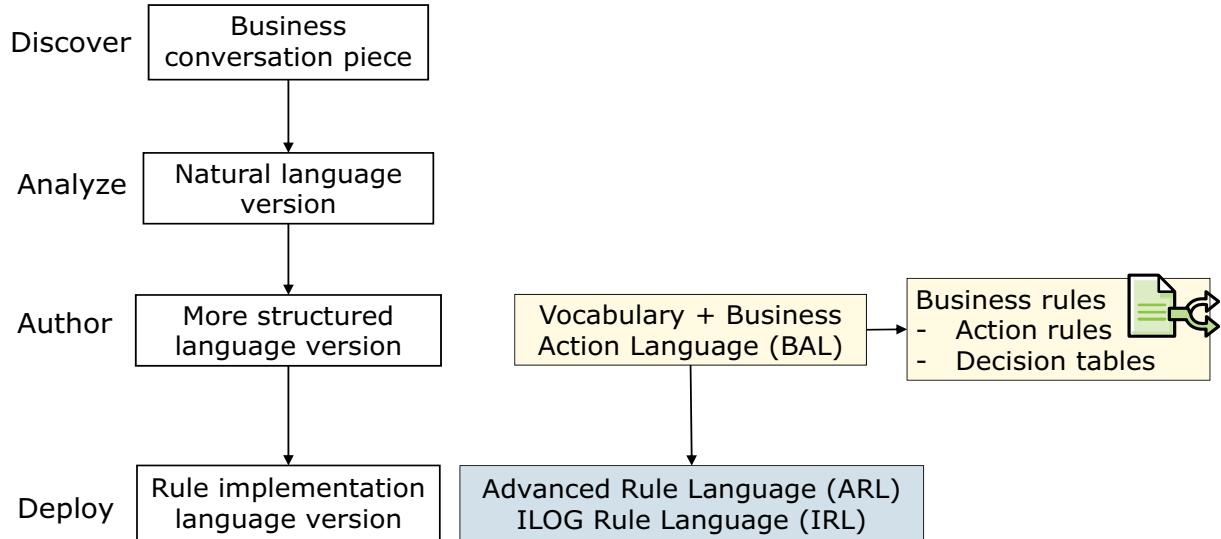
Authoring rules

© Copyright IBM Corporation 2019

*Figure 7-1. From business policy to business rules*



## Rule language evolves during a project (1 of 2)



Authoring rules

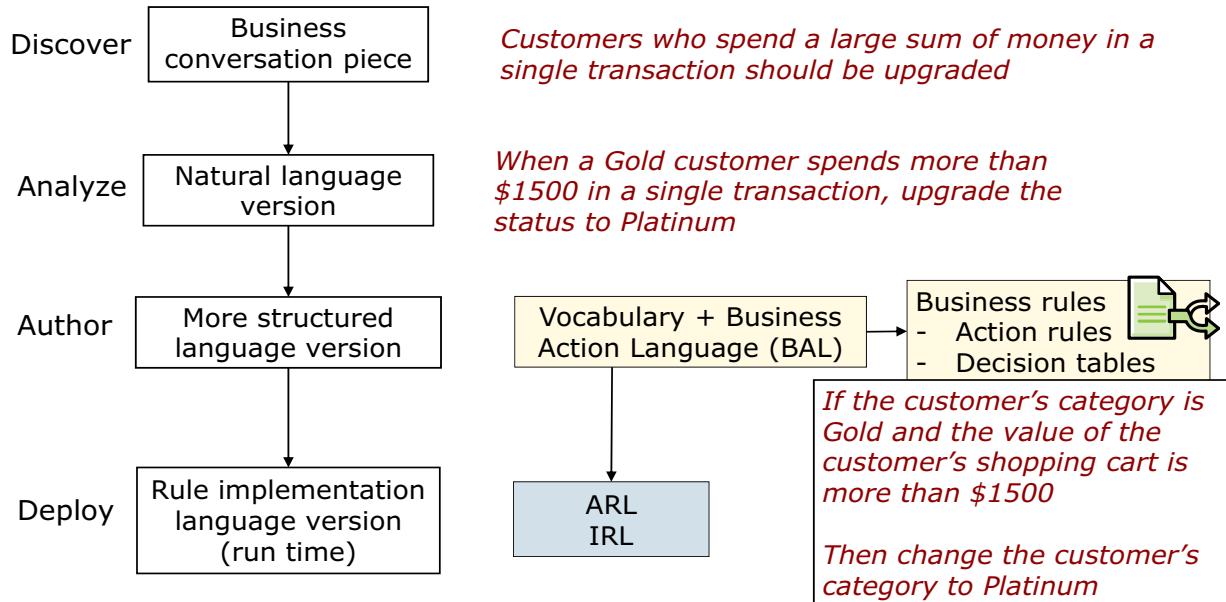
© Copyright IBM Corporation 2019

Figure 7-2. Rule language evolves during a project (1 of 2)

During the process of turning business policy into business rules, the language evolves as shown.

- **Discover:** During the discovery phase, the basis for rules is captured from interviews, business documents, and discussions.
- **Analyze:** During the analysis phase, business analysts work through what they discovered. Rules become more formalized, but they are still on paper and in natural language.
- **Author:** After analysis, developers and business analysts can work together to create and verbalize the business object model. Rule authoring can begin in the tools (including Rule Designer and Decision Center rule editors). Rules can be expressed with various rule artifacts, including action rules, decision tables, and technical rules.
- Rules are written in Business Action Language (BAL), which is a rule language that is used in Operational Decision Manager. BAL is a quasi-natural language that combines rule constructs with terms and phrases that are expressed in the vocabulary. As you learned earlier, the vocabulary is created through verbalization of the business object model in Rule Designer.
- **Deploy:** When rules are deployed, the rule statement, which is written in BAL from the BOM vocabulary, is translated into the rule implementation language, which is what the rule engine sees at run time. This language is called the ILOG rule language, or IRL.

## Rule language evolves during a project (1 of 2)



Authoring rules

© Copyright IBM Corporation 2019

Figure 7-3. Rule language evolves during a project (1 of 2)

Let's take another look at the language during these phases to see how a business policy evolves.

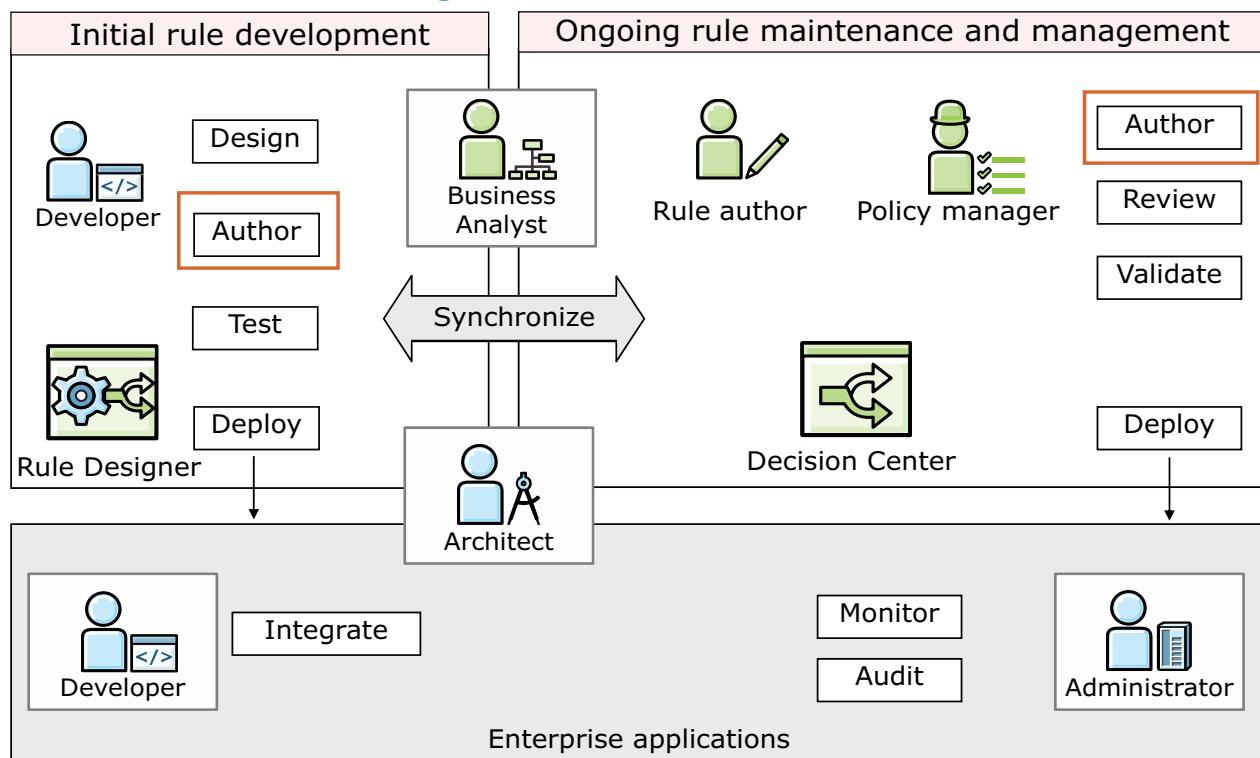
During discovery, you see the rule expressed as a natural language statement.

That rule becomes more formal during the analysis phase. The rule now associates specific vocabulary terms to the customer that are based on purchase amounts.

Next, during the authoring phase, the rule becomes an if-then statement in BAL so that it can be entered in the authoring tool.



## When rule authoring occurs



Authoring rules

© Copyright IBM Corporation 2019

Figure 7-4. When rule authoring occurs

Rule authoring occurs at various phases of the project.

In particular, initial rule development at the beginning of a project requires a large-scale effort to move the business policies into the rule authoring tools. It can take some time for all the business rules that were identified during discovery and analysis to be moved into the tools from English into BAL.

Later as the project evolves, these rules continue to change and require updates or maintenance. Rule maintenance becomes a regular task for some business users as the business policies shift, or as more rules are identified.

## Who writes rules?

- Business analysts
  - Typically involved heavily during initial rule development
  - Might author many rules
- Policy managers and rule authors
  - Act as subject matter experts (SMEs) during early stages
  - Involved in reviewing and validating rules
  - Responsible for maintaining rules (write and edit) after the system is rolled out
- Developers
  - Involved heavily during initial rule development
  - Create technical rules and complex rules

Authoring rules

© Copyright IBM Corporation 2019

Figure 7-5. Who writes rules?

While rule authoring is usually done by the business team, developers also get involved during initial rule development and when more complex, technical rules are required.

## Where are rules written?

- Decision Center Business console and Enterprise console
  - Business console is main rule authoring and management environment for business users
  - Includes features for rule administrators
- Rule Designer
  - Eclipse-based development environment for developers and technical business analysts
  - Includes tools to manage synchronization with Decision Center

Authoring rules

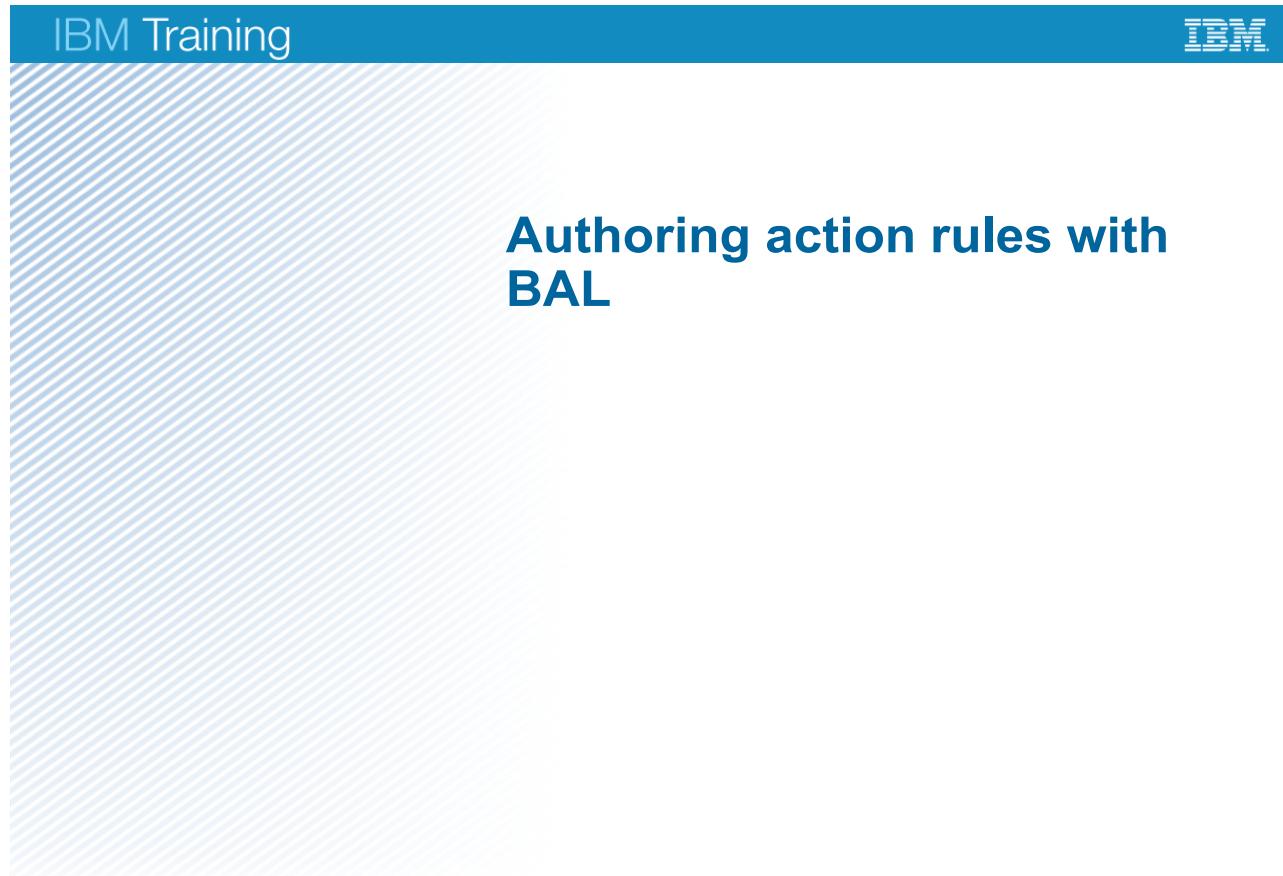
© Copyright IBM Corporation 2019

Figure 7-6. Where are rules written?

As developers, you work in Rule Designer. The rule editors that are available to you include the Intellirule editor, into which you can type directly or copy and paste. It also has an auto-completion feature.

Rule Designer also includes the Guided editor, the decision table editor, the technical rule editor, and the function editor.

## 7.2. Authoring action rules with BAL



*Figure 7-1. Authoring action rules with BAL*

## Business Action Language (BAL)

- Defines a simple syntax to express rules
- Provides constructs for:
  - Defining variables your rules work on
  - Expressing conditions and actions on your business objects
- Can be used throughout Operational Decision Manager
  - In Rule Designer
  - In Decision Center Business and Enterprise consoles

[Authoring rules](#)

© Copyright IBM Corporation 2019

*Figure 7-2. Business Action Language (BAL)*

Action rules are written with the Business Action Language (BAL), which provides a simple syntax to express rules.

BAL provides constructs for defining variables and for expressing conditions and actions on your business data.

BAL is designed to cover most of the common requirements when authoring action rules. You can author action rules in BAL in Rule Designer and Decision Center consoles.

## Rule structure

|                            |  |
|----------------------------|--|
| <b>Definitions</b>         | <ul style="list-style-type: none"> <li>Used to declare rule variables that are used in the rule and tests on these variables, if necessary</li> <li>Definitions are optional</li> </ul>  |
| <b>Conditions<br/>(if)</b> | <ul style="list-style-type: none"> <li>Used to define the conditions that determine when a rule is executed</li> <li>Rules with no conditions are executed under all circumstances</li> <li>Conditions are optional</li> </ul>                             |
| <b>Actions<br/>(then)</b>  | <ul style="list-style-type: none"> <li>Used to define the actions that are done when all conditions are met</li> <li>At least one action is mandatory</li> </ul>   |
| <b>Actions<br/>(else)</b>  | <ul style="list-style-type: none"> <li>Statements after the keyword <code>else</code> define the actions when conditions are not met</li> <li>Keyword <code>else</code> is optional, and valid only when the keyword <code>if</code> is present</li> </ul> |

Authoring rules

© Copyright IBM Corporation 2019

Figure 7-3. Rule structure

Business rules are structured into three main parts:

1. The definitions part, introduced with the keyword `definitions`, is where you declare rule variables that are used in the rule and tests on these variables, if necessary.  
Definitions are optional.
2. Conditions are introduced with the keyword `if`, and they define the tests on your data that determine whether the rule is executed. If no conditions are defined in your rule, the rule is always executed.
3. Actions are introduced with the keyword `then`, and define the actions that are done when all conditions are met. You must have at least one action in a rule.

You can also include an `else` statement, which defines what happens when the conditions are not met. The `else` statement is optional, and it applies only when the rule has a condition.

## Example of action rule

```

definitions
  set 'loyal customer' to a customer
    where the category of this customer is Gold; 1

if
  the age of 'loyal customer' is more than 60 2

then ← a
  give a 10% discount to 'loyal customer';
else ← b
  give a 5% discount to 'loyal customer';

```

Figure 7-4. Example of action rule

Look at this example rule to see how it is broken down.

1. Notice the definitions, where the variable `loyal customer` is defined to be a customer of a certain category. At run time, the rule engine first checks to see whether the rule has a customer object where the category of this customer is gold. If so, then the variable, `loyal customer`, can be defined, and the rest of the rule can be evaluated.
2. The `if` part gives the conditions of the action rule.
3. The `then` part that starts with the keyword `then` defines the actions to take when the conditions are met
  - a. If the `loyal customer` object matches the test in the `if` statement, the `then` statement, or the `action` statement, can then be executed.
  - b. However, in this case, because the rule has an `else` statement, if the condition does not prove true, the `else` statement can be performed on the `loyal customer` object.
    - a. Actions are required in every rule.
    - b. The `else` part is optional. It can be used to define the actions to take when the conditions are not met.

The following slides give more details about these parts.

## Rule definitions: Overview

- Introduce rule definitions with the `definitions` keyword
- Define a rule variable with the `set` keyword and the rule variable name
- Assign a value to a rule variable with the `to` keyword
- As required, introduce constraints with the following keywords:
  - `in <list>`
  - `from <object>`
  - `where <test>`

Figure 7-5. Rule definitions: Overview

You saw on the previous slide how the rule variable is defined in the `definitions` part of the rule.

You use the key keyword `set` and create a variable name. You then assign a value to a rule variable by using the keyword `to`.

You can add other filters on which objects or data can be assigned to that variable by using the keywords `in`, `from`, or `where`.

## Rule definitions: Values

- You can assign the following kinds of values to your rule variable:
  - A constant
  - An expression
  - An object
  - A collection of objects

- Example:

```
definitions
  set 'c1' to 15 %;
  set 'c2' to 5 * 'c1';
  set 'agreement' to 'the current rental agreement';
  set 'categories' to { GOLD, SILVER };
```

Figure 7-6. Rule definitions: Values

When you define a variable, you can define the value to various types. For example, in this definition statement, you see the variable `c1` is set to a constant, `c2` is set to an expression, the `agreement` variable is set to an object, and the `categories` variable is set to a collection of objects.

## Rule definitions: Constraints

- set 'customer' to a customer from the customer of 'agreement'  
**where** the category of this customer is one of 'categories';
- set 'offer' to an offer in the offers of agreement  
**where** the discount of this offer is at least 'c1';
- set 'offerlist' to all offers in the offers of agreement  
**where** the discount of each offer is less than 'c1';

*Figure 7-7. Rule definitions: Constraints*

When you want to filter which values can be assigned to your variable, you can use keywords such as *in* or *where*.

The *in* keyword allows you to access objects in a collection. Notice in the second example, the *offer* variable is set to an offer *in* the offers of agreement. So that means offers of agreement is a collection, such as an array, which includes several offers. This *offer* variable will only be assigned an *offer* object from that collection.

You can also use the keyword *from*, which retrieves an object from another object. The *from* keyword is usually used when you have an inheritance relationship.

You can also use the keyword *where* to further add a test on the objects that must match. Look again at the first example, the *customer* variable is assigned a particular value only if the category of this customer matches certain criteria. Or look at the last example. The *offerlist* variable is only assigned a value when the discount of each offer is less than a certain value.

So in these examples, the variable is only assigned a value when the *where* condition tests true.

## Multiple variables in BAL

- You can define multiple rule variables to denote objects that are instances of the same class
- BAL ensures that the rule variables correspond to different instances
- For example, in the following BAL action rule, `customer1` and `customer2` are different:

```
definitions
  set 'customer1' to a customer;
  set 'customer2' to a customer;
if
  ...

```

- This behavior is not the same for IRL, as you see later

*Figure 7-8. Multiple variables in BAL*

You can define multiple rule variables to denote objects that are instances of the same class. BAL ensures that the rule variables are assigned different instances as values. In the example that is shown here, `customer1` and `customer2` are assigned two distinct instances of the `Customer` class.



### Attention

This behavior differs for technical rules that are written in IRL, as you learn later.

## Rule conditions: Introduction

- Rule conditions are introduced with the `if` keyword and define tests on objects on which your rule works

- Example:

```

definitions
  set 'offer' to an offer in the offers of agreement;
  set 'offerlist' to all offers in the offers of agreement;
if
  any of the following conditions is true:
    - the category of the customer is GOLD
    - the discount of offer is at least 5 %
then
  remove offer from the offers of agreement;

```

*Figure 7-9. Rule conditions: Introduction*

Condition statements are introduced with the keyword `if`, and they define tests on objects to see whether the rule actions should be executed.

The conditions can include one or more logical expressions, but the result of a condition statement always produces a single Boolean result of true or false.

You can use conditions to compare values or objects, test for the existence of an object, count objects, or determine whether an object is a member of a particular group.

The following slides give examples for each of these types of tests.

## Rule conditions: Comparison test

- Compares or establishes relationships between different terms that are found in rule statements
- Examples:
  - Numbers:  
the credit score of 'the borrower' is between 650 and 800
  - Strings:  
the family name of 'the borrower' contains "Smith"
  - Dates:  
the birth date of 'the borrower' is after 1/1/1978
  - Objects:  
the spouse of 'the borrower' is 'some other borrower'

Figure 7-10. Rule conditions: Comparison test

Comparison conditions can test relationships between different values or different objects. As you see in these examples, the test can compare number values, strings, dates, or objects.

## Rule conditions: Membership test

- Tests whether an object belongs to a set
- Examples:
  - if the classification of the vehicle is one of {High-end luxury , Ultra luxury}
  - if the ownership of the vehicle is one of {Financed by credit, Leased}
  - the latest bankruptcy chapter of 'the borrower' is one of {7,11,13}
  - the spouse of 'the borrower' is one of the borrowers of 'the report'

Figure 7-11. Rule conditions: Membership test

Membership tests determine whether an object belongs to a set of values. As you see in these examples, you can check whether the object or attribute is included in a list or in a domain.

## Rule conditions: Existence test

- Tests whether an object is present among the set of objects that are passed to the rule engine
- Examples:
  - if there is at least one customer
  - if there are at least 3 items in the items of the shopping cart
  - if there is at least one item in the list of items in the shopping cart
  - if there is at least one item in the list of items in the shopping cart
    - where the price of this item is more than 1000
  - there is no borrower in the borrowers of 'the report'
    - where this borrower has had a bankruptcy
  - there is at least one borrower in the borrowers of 'the report'

Figure 7-12. Rule conditions: Existence test

Existence tests determine whether an object is present among sets of objects, as shown in the list of examples here.

The BAL constructs to test for existence are: there is at least one **or** there is no

## Rule conditions: Count test

- Counts the number of occurrences of something
- Examples:
  - there are at least 5 borrowers
  - there are at most 3 trucks
  - there are less than 10 drivers
  - there are more than 7 loans
  - if the number of drivers in the request is more than 6

Figure 7-13. Rule conditions: Count test

Count tests can count the number of occurrences of something, as shown in these examples.

## Multiple conditions

- all of the following conditions are true:
  - the yearly income of 'the borrower' is more than 60000
  - the credit score of 'the borrower' is more than 650
- any of the following conditions is true:
  - 'the borrower' is a US citizen
  - 'the borrower' is a permanent resident
- none of the following conditions is true:
  - 'the borrower' is a permanent resident
  - one of the resident coborrowers is a US citizen

[Authoring rules](#)

© Copyright IBM Corporation 2019

*Figure 7-14. Multiple conditions*

A single rule can contain multiple conditions to further filter which objects match the condition tests.

Notice that when you use multiple conditions, you still have only one Boolean result.

## Avoiding ambiguity with parentheses (1 of 2)

- Parentheses can be used to clarify situations that might otherwise be ambiguous

Example:

`if`

```
all of the following conditions are true :
- the category of the customer is Gold
- the age of the customer is at most 15
or the salary of the customer is more than 1000
```

- Is the final “`or`” statement part of the previous phrase “the age of the customer is at most 15,” or a separate item?

- Parentheses can be used for clarity:

`if`

```
all of the following conditions are true :
- the category of the customer is Gold
- (the age of the customer is at least 15
or the salary of the customer is more than 1000)
```

Figure 7-15. Avoiding ambiguity with parentheses (1 of 2)

When you use multiple conditions, you might need to use parentheses to avoid ambiguity in the condition statements. As shown in this example, there might be a question of where the `or` statement applies. Is it a separate condition in addition to the first two listed conditions, or is it the second half of the second bulleted item?

## Avoiding ambiguity with parentheses (2 of 2)

- Although parentheses can clarify complex rules, evaluate whether the rule would make more sense as two rules
- Consider: Do you want to tie two policies together?
  - Example: In the previous rule example, is the age of the customer related to the salary, or are they included together only because they share an action?
- Use ANDs and ORs to group things that are related
- Do not use OR just because the actions are the same, and you want to reuse the actions

Figure 7-16. Avoiding ambiguity with parentheses (2 of 2)

If the rule conditions become too complex, you might want to work with the business analysts to determine whether the rules should be split into two rules. In most cases, atomic rules that produce one result are easier to maintain than combined rules.

When you do need to combine condition tests, you can use Boolean logic operators, AND and OR.

## Use of commas (,) in rule conditions

```
if  
any of the following conditions is true:  
- the category of the customer is GOLD  
- the discount of offer is at least 5 % and (the car group  
upgrade of offer is at least 12 or the price of offer is less  
than 12),  
and there are at most 5 offers in offerlist where the  
discount of each offer is less than 15,  
and the number of offers in offerlist is at least 3
```

Figure 7-17. Use of commas (,) in rule conditions

You can also use commas in the rule conditions to avoid ambiguities. For example, as shown ion this slide, adding a comma is useful when conditions are separated only with the logical operators (and, or).

## Rule actions

- Actions state what the rule does
  - Actions are the executable part of the rule
- A rule must specify at least one action statement
- Actions are executed when all of the conditions and preconditions are met
  - If a rule has no conditions or preconditions, the rule actions are executed if an object that matches is found
  - Example:  
 set the amount of "the account" to 95  
 If "the account" does not exist, the action is not executed
- A rule can be composed of an action statement only
  - When rules do not include definition or condition statements, the actions are always executed

[Authoring rules](#)

© Copyright IBM Corporation 2019

Figure 7-18. Rule actions

Action statements are the executable part of the rule, and can include the `then` and `else` statements. Generally, using an `else` statements is harder to maintain.

- The `then` block of action statements is mandatory and contains all the action statements that are found between the keyword `then`, and either the `else` keyword (if it exists) or the end of the rule artifact.

The `then` statements are executed when the condition statements test true.

- The `else` block of action statements is optional and contains all the action statements that are found after the `else` keyword. If the condition statements test false, then the `else` block executes.

When a rule contains both definitions and conditions (`if`), the action in the `else` block executes when the definitions are satisfied and the conditions (`if`) test false.

In the following rule, a discount is always applied, and any customer receives at least a 5% discount:

```
if
  the category of the customer is Gold and the value of the customer's shopping
  cart is more than $100
then
  apply a 15% discount
else
  apply a 5% discount
```

If you adjust the rule by adding a definition, a discount is applied only for customers in the Gold category:

```
definitions
  set applicant to a customer where the category of this customer is Gold
if
  the value of the applicant's shopping cart is more than $100
then
  apply a 15% discount
else
  apply a 5% discount
```

## Examples of rule actions

- Rule actions can modify objects, attributes, or variables by using “set” statements

```
    set the membership of the customer to GOLD
```

- Actions can execute a method or a function, such as “apply a 10% discount”:

```
if
    the value of the shopping cart is more than $100
then
    apply a 10% discount on the shopping cart
```

- Action that sets Boolean values:

- make it true that
- make it false that

- Action rules with no definitions or conditions always execute:

```
then set the total price of 'the policy'
    to (100 + 'tax rate percentage') * the total price before tax
    of 'the policy' / 100;
```

*Figure 7-19. Examples of rule actions*

As a result of an action statement, objects can be modified.

Here are some examples of rule actions and how they can manipulate objects. Notice that values can be changed, Boolean expressions can be set, and values can be calculated.

## BAL number operators

| BAL construct                               | Operator |
|---|----------|
| Is more than                                | >        |
| Is at least                                 | $\geq$   |
| Is less than                                | <        |
| Is at most                                  | $\leq$   |
| Equals                                      | =        |
| Does not equal                              | $\neq$   |
| Is between <number> and <number>            | [ , ]    |
| • Includes endpoints                        |          |
| Is strictly between <number> and <number>   | ] , [    |
| • Excludes endpoints                        |          |
| Is at least <number> and less than <number> | [ , [    |
| Is more than <number> and at most <number>  | ] , ]    |

Figure 7-20. BAL number operators

BAL constructs and operators, as you see listed here, are vocabulary that is associated with what is referred to as the *boot BOM*. The boot BOM is a verbalized system vocabulary that is visible in the Vocabulary view of your project. You can work with the boot BOM to change its verbalization if, for example, you need to localize your rules.

## BAL arithmetic operators

| BAL construct       | Description  |
|---------------------|--|
| <number> + <number> | Adds a number to another number  |
| <number> - <number> | Subtracts a number from another number   |
| <number> / <number> | Divides a number by another number   |
| <number> * <number> | Multiplies a number by another number  |
| <text> + <text>     | Concatenates two strings<br><b>Example:</b> set 'full name' to<br>'first name' + ' ' + 'last name' |

Figure 7-21. BAL arithmetic operators

You can find more information about BAL in the *BAL Reference* section of the production documentation. The *BAL Reference* provides a full list of constructs and operators that you can use in your rules, along with explanations of how they work and various examples.

## 7.3. Authoring decision tables

## Authoring decision tables

Authoring rules

© Copyright IBM Corporation 2019

*Figure 7-1. Authoring decision tables*

## Decision tables

- Concise way of viewing a set of action rules in the form of a spreadsheet or tree structure
- Suitable for representing sets of action rules that have a uniform structure
- Advantages of decision tables
  - Preconditions that apply to an entire decision table
  - Built-in error checking that verifies the structure of your data for gaps and overlaps

Authoring rules

© Copyright IBM Corporation 2019

*Figure 7-2. Decision tables*

Decision tables provide a concise way of viewing a set of business rules in the form of a spreadsheet structure. They are suitable for representing sets of business rules that have a uniform structure.

## Preconditions

- A precondition can be added to a decision table
  - To limit the scope of the rules in decision tables
  - To define variables that can be used in all the rules of the decision table
- The precondition of a decision table applies to all the rules that this decision table defines

*Figure 7-3. Preconditions*

When working with decision tables, you can add preconditions that apply to the entire table.

You can use preconditions to limit the scope of the rules, or to define variables.

## Error checking in the editors

- Decision table editor can verify your data and the structure of your data
- Decision tables verify:
  - Expressions in cells
  - Overlapping data with hierarchical discrimination
  - The symmetry and the contiguity of partition items

*Figure 7-4. Error checking in the editors*

As you edit decision tables, the editor can verify your data and the structure of your data, and highlight any problem so that you can correct it immediately.

The decision table editor can automatically check expressions in cells. It can check for overlaps in the data, and it can check the symmetry and contiguity of partition items.

## About decision tables

- Convenient way to view and manage large sets of similar business rules
- Composed of rows and columns:
  - Each row corresponds to a single rule
  - Columns define the conditions and actions
- Other benefits:
  - Can apply preconditions to all rules
  - Easy to see overlaps or gaps
  - Consistency checking features (static rule analysis)
- When to use decision tables:
  - Common condition tests
  - Similar actions

Figure 7-5. About decision tables

Decision tables provide a convenient way to view and manage large sets of similar business rules. They are composed of rows and columns. Each row corresponds to a single rule. The columns are used to define the conditions and actions.

A benefit of creating decision tables is that you can define preconditions that are applied to each row (or each rule) in the table. Also, you can easily see overlaps and gaps between rules. In addition, decision tables have built-in consistency checking features.

Decision tables are generally used when rules share common condition terms, and when they have similar actions.

## Example: Symmetrical rules

- Patterns are a good indicator that a decision table might be useful
  - If ***the miles per year of the vehicle*** is less than 4,000, then **set the third-party price before tax of the policy** to the third-party price before tax of the policy \* 0.6
  - If ***the miles per year of the vehicle*** is at least 4,000 and less than 8,000, then **set the third-party price before tax of the policy** to the third-party price before tax of the policy \* 0.8
  - If ***the miles per year of the vehicle*** is at least 8,000 and less than 15,000, then **set the third-party price before tax of the policy** to the third-party price before tax of the policy \* 1
  - If ***the miles per year of the vehicle*** is at least 15,000 and less than 25,000, then **set the third-party price before tax of the policy** to the third-party price before tax of the policy \* 1.2
  - If ***the miles per year of the vehicle*** is at most 25,000, then **set the third-party price before tax of the policy** to the third-party price before tax of the policy \* 1.3
- Each of these rules can become a row in a decision table

[Authoring rules](#)

© Copyright IBM Corporation 2019

Figure 7-6. Example: Symmetrical rules

How do you know when to use a decision table? Look for patterns in your rules.

Notice on this slide that each rule can become a row in a decision table because each rule uses the same information.

- All the conditions test the same attribute: ***the miles per year of the vehicle***
- Each action sets the same attribute: ***the third-party price before tax of the policy***

Look for this type of a pattern to determine whether the business logic belongs in a decision table.

## Example: Symmetrical rules in a decision table

- As a decision table, the same set of rules is much easier to read

|   | Vehicle miles per year | MPY FActor |
|---|------------------------|------------|
| 1 | < 4,000                | 0.6        |
| 2 | [4,000 8,000[          | 0.8        |
| 3 | [8,000 15,000[         | 1          |
| 4 | [15,000 25,000[        | 1.2        |
| 5 | ≥ 25,000               | 1.3        |

- Each row represents a rule
  - When conditions for a row are met, actions in that row are executed

Rule

**if**  
**all of the following conditions are true :**  
 - ( the miles per year of '**the vehicle**' is less than **4000** ) ,  
**then**  
 set the third party price before tax of '**the policy**' to the third party price before tax of '**the policy**' \* **0.6** ;

|   | Vehicle miles per year | MPY FActor |
|---|------------------------|------------|
| 1 | < 4,000                | 0.6        |

Authoring rules

© Copyright IBM Corporation 2019

Figure 7-7. Example: Symmetrical rules in a decision table

On this slide, you see that the same list of rules that were presented on the previous slide are now in an easy-to-read decision table. Each rule becomes a row.

Condition columns are on the left, and show the number of miles per year that a vehicle uses. The action column is on the right, and represents the calculation to set the third-party price before tax of the policy.

Because each row is a distinct rule, only one row of a decision table can execute. When conditions for one of the rows are met, the actions in that row are executed.



## Structure of a decision table

Diagram illustrating the structure of a decision table:

|               | Row header | Condition column       | Action column |
|---------------|------------|------------------------|---------------|
| Column header |            |                        |               |
| Rows = Rules  | 1          | Vehicle miles per year | MPY Factor    |
|               | 2          | < 4,000                | 0.6           |
|               | 3          | [4,000 8,000[          | 0.8           |
|               | 4          | [8,000 15,000[         | 1             |
|               | 5          | [15,000 25,000[        | 1.2           |
|               |            | ≥ 25,000               | 1.3           |

Authoring rules

© Copyright IBM Corporation 2019

Figure 7-8. Structure of a decision table

In the decision table editor, condition columns are always on the left, and action columns are always on the right.

You can change the labels for column headers to meaningful names to help you identify the condition or action that the column represents.

Row headers are automatically numbered to identify the specific row or rule. Row headers are useful when auditing decision results or other reports, because they specify which row of a table was used during rule execution.

## Columns for conditions and actions

- Condition columns are on the left, unshaded
- Action columns are on the right, shaded



|          | <b>Vehicle miles per year</b> | <b>MPY FActor</b> |
|----------|-------------------------------|-------------------|
| <b>1</b> | < 4,000                       | 0.6               |
| <b>2</b> | [4,000 8,000[                 | 0.8               |
| <b>3</b> | [8,000 15,000[                | 1                 |
| <b>4</b> | [15,000 25,000[               | 1.2               |
| <b>5</b> | ≥ 25,000                      | 1.3               |

Figure 7-9. Columns for conditions and actions

Notice that the action column has a shaded background, which helps to visually distinguish it from condition columns.

## Locking facilities

- Decision tables have locking facilities to protect your decision tables against editing by others
- Locking facilities apply to both the condition columns and action columns
- You can lock the structure of the decision table
- You can also lock any predefined data

Figure 7-10. Locking facilities

Decision tables have locking tools that allow you to protect the contents from editing by others. For both condition columns and action columns, you can lock the decision table structure, or data, or both.

## Decision table size

- Try not to create decision tables too large in terms of the number of rows and number of columns
  - A few hundred rows and 10 – 20 columns are acceptable values to keep the tools efficient
- If a decision table becomes too large, the time it takes to save and compile the data becomes too long

Authoring rules

© Copyright IBM Corporation 2019

Figure 7-11. Decision table size

When you create a decision table, consider that a few hundred rows and perhaps 10-20 columns are acceptable values to ensure that the decision table remains efficient. If a decision table is too large, saving and compilation data can take longer.



### Information

Compilation in Rule Designer occurs each time that the table is saved. In Decision Center consoles, it occurs when the ruleset is generated.

## 7.4. Advanced Rule Language (ARL)



*Figure 7-1. Advanced Rule Language (ARL)*

## Translation from BAL to ARL

- ARL is a Java-like language, designed for the decision engine
  - Rules and decision tables are written in BAL but translated to ARL for the decision engine
- Example:

|            |  |
|------------|--|
| <b>BAL</b> | <pre>if   the credit score of b is less than 200 then   add "Credit score below 200" to the messages of 'the loan';   reject 'the loan';</pre>   |
| <b>ARL</b> | <pre>when {   miniloan.Borrower() from \$EngineData.this.borrower;   evaluate ( \$EngineData.this.borrower.creditScore &lt; 200); } then {   \$EngineData.this.loan.addToMessages("Credit score below 200");   \$EngineData.this.loan.reject</pre> |

[Authoring rules](#)

© Copyright IBM Corporation 2019

Figure 7-2. Translation from BAL to ARL

The Advanced Rule Language is a Java-like language designed for the decision engine. This rule language has a similar syntax to Java 7, with some rule-focused additions.

Rules and decisions are written in BAL, but translated to ARL for the decision engine to process. On this slide, you see a rule written in BAL and its ARL counterpart.

The rule states that when a customer asks for a loan but the credit score is less than 200, the request is denied. The message 'Credit score below 200' is attached to the loan application.

The ARL translation example on this slide illustrates some of the differences from Java:

- Identifiers can be written in free text with back-quotes: `eligibility.minimum credit score`
- ARL keywords include: **rule**, **when**, **ruleset**, **signature**, **evaluate**, **from**, **ruleset**, **match**, many, **in**, **flowtask**, **ruletask**, and **aggregate**
- Use **\$EngineData.this** to reference BOM objects and ruleset parameters



## Viewing ARL

- You can view the ARL translation of a rule on the ARL tab of the rule editor in Rule Designer
  - The ARL tab is read-only; you edit rules only in BAL on the Intellirule tab

The screenshot shows the IBM Rational Rules Rule Designer interface. A rule named 'minimum credit score' is displayed in the main editor area. The code is written in ARL (Apache Rule Language). The ARL tab is highlighted with an orange border at the bottom of the window.

```

rule `eligibility.minimum credit score` {
    ilog.rules.business_name = "minimum credit score"
    ilog.rules.dt = ""
    ilog.rules.package_name = "eligibility"
    status = "new"
    when {
        miniloan.Borrower() from $EngineData.this.borrower;
        evaluate ( $EngineData.this.borrower.creditScore < 200);
    }
    then {
        $EngineData.this.loan.addToMessages("Credit score below 200");
        $EngineData.this.loan.reject();
    }
}

```

Authoring rules

© Copyright IBM Corporation 2019

*Figure 7-3. Viewing ARL*

You can view the ARL translation of a rule on the ARL tab of Rule Designer. The ARL tab is read-only. You cannot edit rules in ARL.

## 7.5. Technical rules



Authoring rules

© Copyright IBM Corporation 2019

*Figure 7-1. Technical rules*

## Writing technical rules in IRL

- Technical rules support constructs and features unavailable in BAL, including loops and explicit IRL mapping
- Technical rules are written using IRL
  - Java-like rule language that can be executed directly by the rule engine
- Technical rule structure:
  - Conditions
  - Actions
  - You can define variables within the condition or action statements
- Use technical rule actions to:
  - Control execution
  - Make loops
  - Branch
  - Handle exceptions
  - Discriminate between execution branches

[Authoring rules](#)

© Copyright IBM Corporation 2019

*Figure 7-2. Writing technical rules in IRL*

Technical rules are written by using the ILOG Rule Language (IRL). IRL is a Java-like rule language that can be executed directly by the rule engine.

A technical rule is made of a condition part and an action part. Technical rules and functions are written directly against the BOM elements, and do not use verbalized terms or phrases.

Technical rule actions control execution, make loops, branch, handle exceptions and discriminate between execution branches.

## IRL syntax

- Conditions begin with the keyword **when**
- Actions begin with the keyword **then**
- Variables:
  - Simple class condition variable:  
`variable : ClassName();`
  - Simple variable in a condition:  
`evaluate ( variable : <value> )`
  - Simple variable in the action:  
`int variable : <value>;`
  - The **?x** variable at the beginning of the condition identifies any object that matches this condition that you can use in other conditions or actions
- To import required elements, use the keywords **use** and **import**

[Authoring rules](#)

© Copyright IBM Corporation 2019

Figure 7-3. IRL syntax

On this slide, you see some of the IRL syntax. Technical rules are composed of IRL keywords, BOM elements in their code form, values, and constants.

- Conditions begin with the keyword **when**.  
 Conditions define both the variables that are used in the rules and any tests on these variables.  
 In BAL, you have the **definitions** and **if** keywords for this purpose.
- Actions begin with the keyword **then**.  
 Actions specify the actions to execute when the conditions are met.  
 You can specify **else** statements to execute when the rule conditions are not met.
- As in BAL, you can define variables in the conditions of a technical rule.  
 The presence of the **?x** variable at the beginning of a condition serves as a marker. It identifies any object that matches this condition. You can then look at the matched object in other conditions or in the actions of the rule. In this example, the first action asks for the removal of the objects that are referenced with the **?x** rule variable. As a result, any **Fish** object of color

green and of type shark is removed from the set of objects that are provided to the rule engine for execution.

```
when {  
?x: Fish(color==green; type==shark);  
} then {  
retract ?x;  
}
```

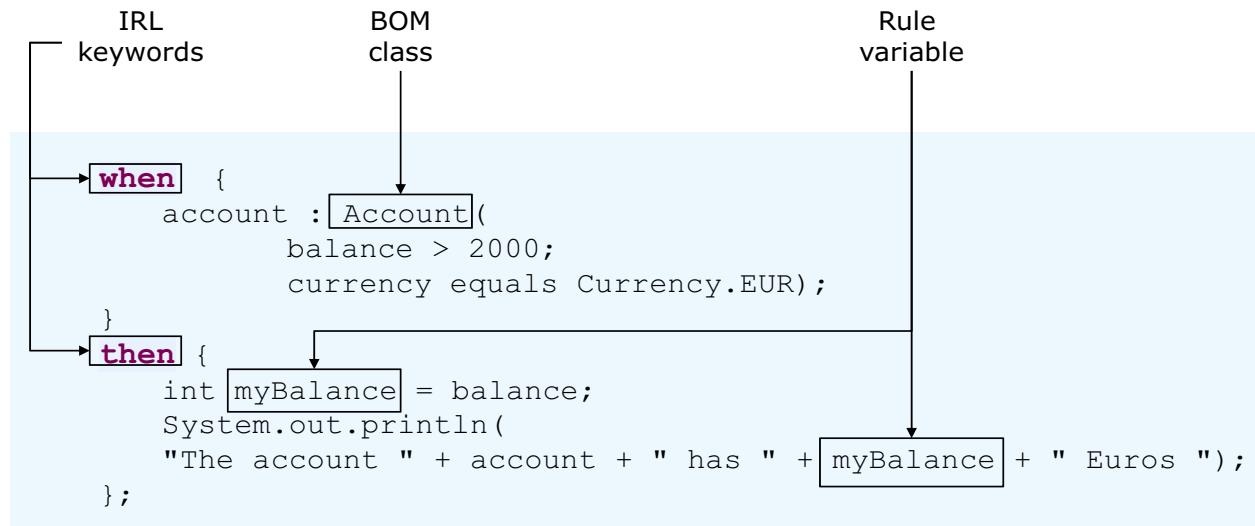
- To import required elements, use the following keywords:
  - use: To import ruleset elements (for example, variables)
  - import: To import BOM elements



### Information

The question mark in variable names such as ?x is optional.

## Technical rule structure



Authoring rules

© Copyright IBM Corporation 2019

*Figure 7-4. Technical rule structure*

On this slide, you can see the example structure and syntax of an IRL rule. You see the IRL keywords, the BOM class that it is operating on, and an example of a variable.

## Multiple variables in IRL

- If you define multiple rule variables to denote instances of the same class, they might denote the same instance when the rule executes

```
when {
customer1: carrental.Customer();
customer2: carrental.Customer();
} then {
...
}
```

- To make sure that these instances are different, add an IRL test

```
when {
customer1: carrental.Customer();
customer2: carrental.Customer(?this != customer1);
} then {
...
}
```

Figure 7-5. Multiple variables in IRL

In IRL, you can define multiple rule variables to denote instances of the same class. However, if you do not specify further constraints, they can denote the same instance.

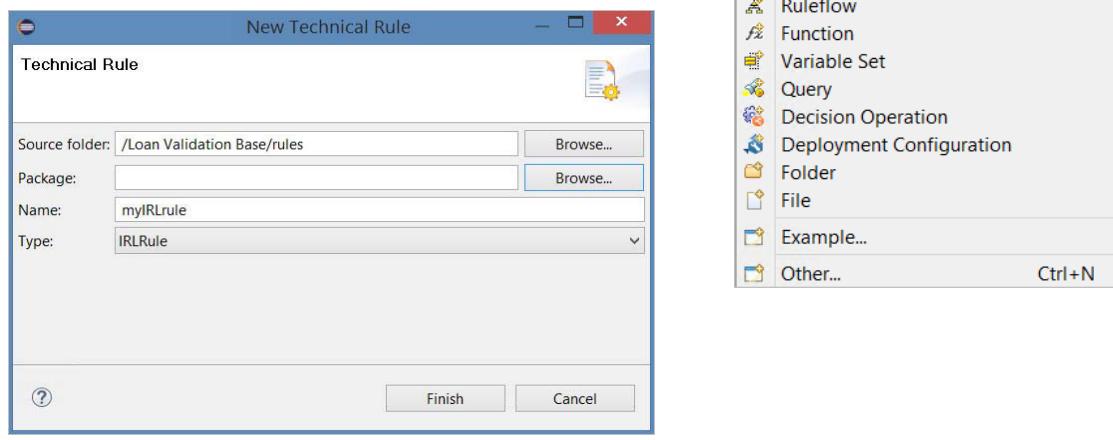
For example, the first IRL rule on this slide defines two variables `customer1` and `customer2` that denote instances of the `Customer` class in the working memory. However, in IRL, *these two variables can denote the same instance of the `Customer` class in the working memory.*

To make sure that these instances are different, you must explicitly add an IRL test. For example, you can add it in the definition of `customer2`, as you see on the slide.



## Create a technical rule

- From the **File** menu, click **New > Technical Rule**
- In the New Technical Rule wizard, enter the **Source folder**, **Package**, and **Name**
  - Type is **IRLRule**



Authoring rules

© Copyright IBM Corporation 2019

Figure 7-6. Create a technical rule

To create a technical rule in IRL, you work from the Rule perspective in Rule Designer and use the **File > New > Technical Rule** menu.

In the Technical Rule wizard, note that the type is specified as **IRLRule**.

## 7.6. Defining objects that are used in rules

## Defining objects that are used in rules

Authoring rules

© Copyright IBM Corporation 2019

*Figure 7-1. Defining objects that are used in rules*

## Introduction

- Business objects are made available to the rule artifacts and the rule engine by using:
  - Parameters
  - Ruleset variables
  - Objects in working memory
- After you define and verbalize these parameters or variables, they become vocabulary that you can use in the rule artifacts

Authoring rules

© Copyright IBM Corporation 2019

Figure 7-2. Introduction

To produce decisions, the rule artifacts are tested against objects that are passed from your client application.

The application can pass those objects as *parameters*, or as *objects* in the working memory. You manipulate those objects as *ruleset variables* in your decision service.

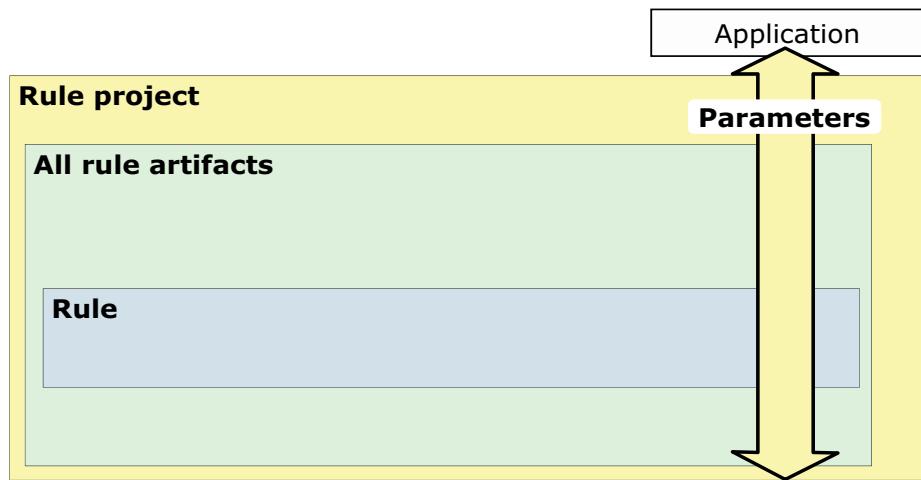


### Important

The objects that you model as parameters or ruleset variables might also play a role in orchestrating the execution of your rule artifacts.

## Parameters

- Pass objects between the application and the rule engine
  - Constitute the signature of the rulesets that are generated from the rule project
- Scope:
  - Available to all rule artifacts in the rule project or dependent projects
  - Accessible to the application



Authoring rules

© Copyright IBM Corporation 2019

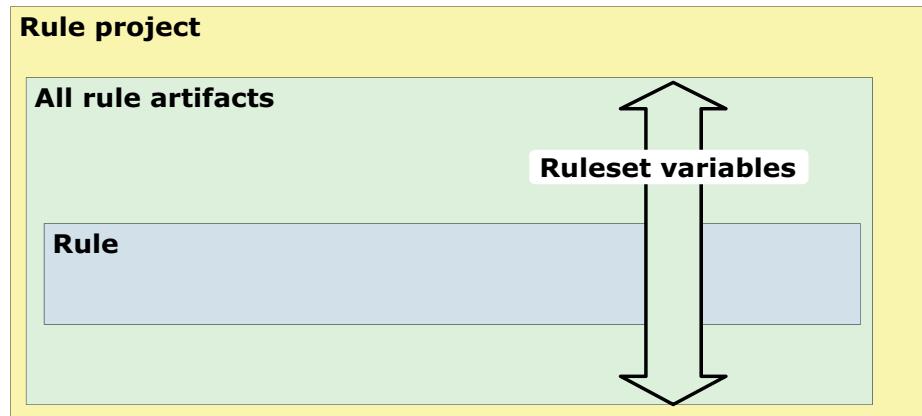
Figure 7-3. *Parameters*

Ruleset parameters can be used to pass object from the application to the rule engine, and can also be returned to the application with the decision result.

Because the parameters are accessible to the application, they form the signature of the ruleset. Their values that are passed to or from the client application, or in both directions, are the actual objects on which your rule artifacts work.

## Ruleset variables

- Pass data between rules and ruleflow tasks
- Grouped into variable sets that are stored in rule packages or at the project root
- Scope: Available to all rule artifacts in the ruleset



Authoring rules

© Copyright IBM Corporation 2019

Figure 7-4. Ruleset variables

Ruleset variables define objects internal to the ruleset. You design ruleset variables within a *variable set*.

Like the parameters, the ruleset variables are accessible to all rule artifacts in your rule project.

You bind parameters to ruleset variables, and then you can manipulate the ruleset variables to transfer values between tasks in a ruleflow, to store intermediate results, or to compute a condition.

## Objects in working memory

- The working memory contains objects that any rule artifact can reference
- You can add, remove, or modify objects in the working memory:
  - With a statement in the actions of a rule
  - By using the application programming interface (API)

[Authoring rules](#)

© Copyright IBM Corporation 2019

*Figure 7-5. Objects in working memory*

Your rule artifacts can also work on objects in the working memory. You can directly add, remove, or change objects in the working memory directly by using the application programming interface (API).

## Working with objects in rule artifacts

- Manipulation of objects in rule artifacts depends on the object type and the language used

| Type of objects                            | How to manipulate  |
|--|--|
| Parameters                                 | With the name, in IRL<br>With the verbalization, in BAL<br>With a rule variable, in IRL or BAL |
| Ruleset variables                          | With the name, in IRL<br>With the verbalization, in BAL<br>With a rule variable, in IRL or BAL |
| Objects in the working memory              | With a rule variable, in IRL or BAL<br>With an automatic variable, in BAL                      |
| Members (methods, attributes) of an object | With the name, in IRL<br>With the verbalization, in BAL<br>With a rule variable, in IRL or BAL |

Figure 7-6. Working with objects in rule artifacts

As you can see in this table, the way objects are manipulated depends on the type of object and whether the rule was written in IRL or BAL.

## Using IRL

- In rule artifacts that are written in IRL, such as technical rules or functions, you can manipulate all named elements with their names:
  - For example, a parameter that is called `report` can be manipulated in IRL as follows: `evaluate (report.approved);`
- Because objects in the working memory have no name, they cannot be manipulated that way

Figure 7-7. Using IRL

When you use IRL, parameters and ruleset variables have a name and can be manipulated with their names in rule artifacts that are written in IRL, such as technical rules or functions. For example, a ruleset parameter (or a ruleset variable) called `report` can be manipulated in IRL:

```
evaluate (report.approved);
```

Members (methods, attributes) of objects also have a name and can be manipulated in rule artifacts that are written in IRL.

Because objects in the working memory have no name, they cannot be manipulated that way.

## Using verbalized elements in BAL

- In rule artifacts that are written in BAL, you can manipulate all verbalized elements with their verbalization
  - For example, if the parameter called `report` is verbalized as 'the loan report', it can be manipulated in BAL as follows:  
`if 'the loan report' is approved;`
- Because objects in the working memory have no associated verbalization, they cannot be manipulated that way

*Figure 7-8. Using verbalized elements in BAL*

When your parameters or variables are verbalized, they can also be manipulated with BAL. Because the BAL rule editors work only with verbalized elements of the BOM, if you want a BOM class or member to show up in the rule editor, it must be verbalized.

For example, if a ruleset parameter (or a ruleset variable) called `report` is verbalized as 'the loan report', it can be used in BAL as follows:

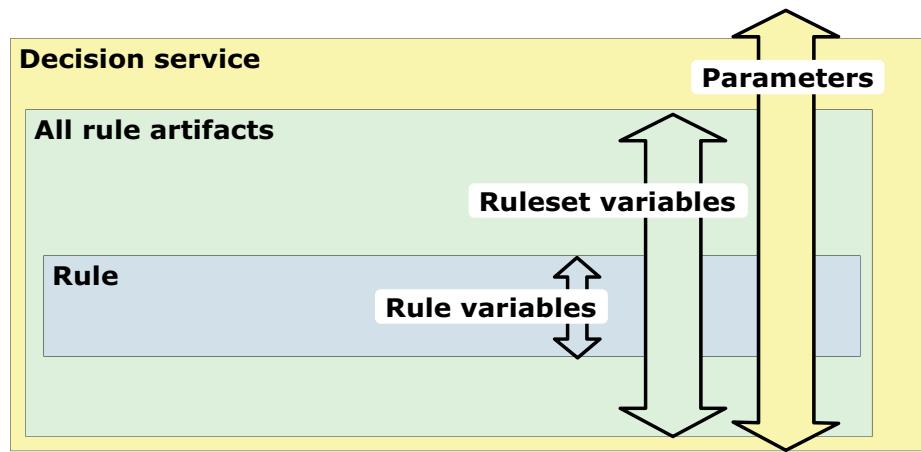
```
if 'the loan report' is approved;
```

*If they are verbalized*, members (methods, attributes) of an object can also be manipulated with their verbalization (navigation or action phrases) in rule artifacts that are written in BAL.

Because objects in the working memory have no associated verbalization, they cannot be manipulated that way.

## Rule variables (1 of 2)

- Defined in the *definitions* part of rule artifacts
- Used to manipulate the objects that are passed by a parameter or a ruleset variable, or an object in the working memory
- Can also be used to simplify long verbalizations for easier readability
- Scope: Available only within the rule that defines the rule variable



Authoring rules

© Copyright IBM Corporation 2019

Figure 7-9. Rule variables (1 of 2)

If you want to manipulate a parameter, a ruleset variable, or an object in the working memory, you can use a *rule variable*.

In BAL, rule variables are declared in the definitions part of the rule. In IRL, you define it in the condition part of the rule.

After the rule variable is declared, it can be used in any part of the rule statement, but it cannot be used outside of the scope of the rule.

## Rule variables (2 of 2)

- Because objects in the working memory cannot be named or verbalized, you must bind them to rule variables or automatic variables by defining a pattern in the definitions of the rule artifact to manipulate them
- You can also use rule variables to enhance the way that your rules are written

Authoring rules

© Copyright IBM Corporation 2019

Figure 7-10. Rule variables (2 of 2)

Because objects in the working memory cannot be named or verbalized, the only way to work with them is to bind them to rule variables or automatic variables.

Another use for rule variables is to simplify the wording in a condition or action statement. Simplifying the wording can be useful when you are supporting rule authors who find the wording of rule statements too complex or long to be understood easily.

## Using automatic variables

- Automatic variables are useful when you must execute rules on all the objects in the working memory
- An automatic variable is a rule variable that rule authors can use without the need to explicitly declare it in the rule definitions
- You declare automatic variables in the BOM editor
- Rule artifacts that use automatic variables are tested against all instances of this BOM class in the working memory

*Figure 7-11. Using automatic variables*

Automatic variables are designed to manipulate *objects in the working memory*.

You declare an automatic variable at the BOM class level. When you are in the BOM editor, you see the **Generate automatic variable** option. When you select the automatic variable option for a BOM class, it immediately becomes available in the rule editor, and is prefaced by: "the"

Rules that use automatic variables can evaluate every instance of that BOM class in the working memory. For example, say that you have a rule that uses an automatic variable for the `Customer` BOM class, and the working memory contains a group of `Customer` objects. The rule then iterates over every `Customer` instance in the memory.

When you use rule variables or automatic variables together with parameters or ruleset variables, make sure that they all have unique names so that there is no ambiguity. If the names are too similar, it might create confusion for the rule authors when they try to select the right term in the rule editor. They might end up with the wrong results at run time.

## Unit summary

- Describe rule languages
- Use the various rule editors to author rule artifacts
- Define the objects that rule artifacts manipulate

Authoring rules

© Copyright IBM Corporation 2019

*Figure 7-12. Unit summary*

## Review questions

1. True or False. Every rule must include a condition statement.
2. Apart from Business Action Language (BAL), which other language can you use to write rule artifacts?
  - A. Technical Rule Language (TRL)
  - B. BOM-to-XOM mapping (B2X)
  - C. ILOG Rule Language (IRL)
  - D. Guided Rule Language (GRL)
  - E. None of the above



Authoring rules

© Copyright IBM Corporation 2019

Figure 7-13. Review questions

Write your answers here:

- 1.
- 2.

## Review answers

1. True or False. Every rule must include a condition statement.  
False. Rules that do not include conditions always execute.
  
2. Apart from Business Action Language (BAL), which other language can you use to write rule artifacts?
  - A. Technical Rule Language (TRL)
  - B. BOM-to-XOM mapping (B2X)
  - C. ILOG Rule Language (IRL)
  - D. Guided Rule Language (GRL)
  - E. None of the above

The answer is C.



Figure 7-14. Review answers

## Exercise: Exploring action rules

Authoring rules

© Copyright IBM Corporation 2019

*Figure 7-15. Exercise: Exploring action rules*

## Exercise introduction

- Identify the parts of an action rule
- Explain the difference between using automatic variables or rule variables



Authoring rules

© Copyright IBM Corporation 2019

*Figure 7-16. Exercise introduction*

## Exercise: Authoring action rules

Authoring rules

© Copyright IBM Corporation 2019

*Figure 7-17. Exercise: Authoring action rules*

## Exercise introduction

- Use the Intellirule editor and Guided editor to author action rules
- Use rule variables, automatic variables, and parameters in rule statements



Authoring rules

© Copyright IBM Corporation 2019

*Figure 7-18. Exercise introduction*

## Exercise: Authoring decision tables

Authoring rules

© Copyright IBM Corporation 2019

Figure 7-19. Exercise: Authoring decision tables

## Exercise introduction

- Use the decision table editor to create a decision table



[Authoring rules](#)

© Copyright IBM Corporation 2019

*Figure 7-20. Exercise introduction*

---

# Unit 8. Customizing rule vocabulary with categories and domains

## Estimated time

01:00

## Overview

This unit teaches you how to work with categories and domains to customize rule vocabulary.

## How you will check your progress

- Review
- Exercises

## Unit objectives

- Simplify rule authoring by using categories
- Define domains

*Figure 8-1. Unit objectives*

## Topics

- Simplifying vocabulary with categories
- Defining domains
- Static domains
- Dynamic domains
- Updating dynamic domains in Decision Center

## 8.1. Simplifying vocabulary with categories

## Simplifying vocabulary with categories

Customizing rule vocabulary with categories and domains

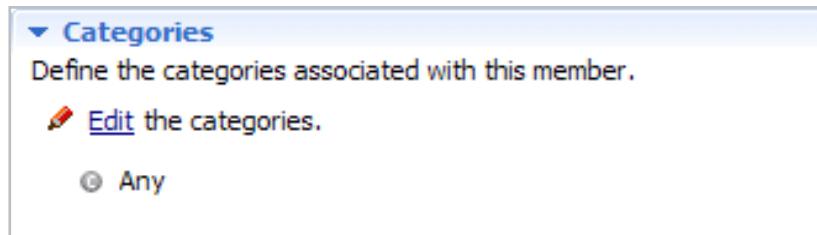
© Copyright IBM Corporation 2019

*Figure 8-3. Simplifying vocabulary with categories*



## Categories in the BOM editor

- Categories are identifiers that you apply to BOM elements to specify whether these BOM elements can be used in specific rules



Customizing rule vocabulary with categories and domains

© Copyright IBM Corporation 2019

Figure 8-4. Categories in the BOM editor

A **category** is an identifier that you apply to BOM classes and members to filter the BOM elements available in your business rules.

You tag BOM elements with categories to specify whether these BOM elements can be used in specific rules.

If a category is “hidden” for a specific rule, business elements that are tagged with that category are invisible in the rule editor.

When writing rules, if the category filter is in use, only BOM elements with matching categories show up in the vocabulary list for use in the rule editor.

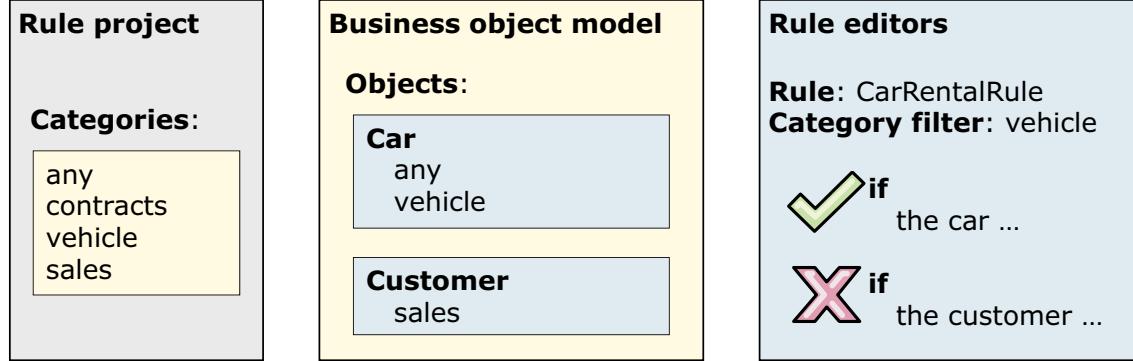
Use categories to simplify the list of vocabulary that is available in the rule editors. This simplifies rule authoring for the business users, and ensures that only the vocabulary that is relevant to the rule is visible in the vocabulary list.



## Categories

- Tags on the BOM elements that can be used as a filter in the rules
  - If a category is “hidden” for a specific rule, BOM elements that are tagged with that category are invisible in the rule editor
  - If the category filter is in use, only BOM elements with categories that match the rule category show up in the vocabulary list when that rule is edited
  - Simplifies rule authoring by removing vocabulary that is relevant to the rule from the list of choices in the rule editors

- Example:



Customizing rule vocabulary with categories and domains

© Copyright IBM Corporation 2019

Figure 8-5. Categories

Categories can be helpful when you have a large vocabulary. Categories simplify the authoring task by acting as filters. They reduce the number of elements available as vocabulary in the rule editors.

A category is a tag or identifier that can be applied to business classes and members and filtered in business rules. You set categories to specify whether a business element can be used in a specific rule. By default, the predefined category `any` is set on all business elements, meaning that all business elements can be used in all business rules that are linked to that BOM.

For example, as shown here, the `Car` class is assigned the category “vehicle” in the BOM. The rule project also has a business rule, `CarRentalRule`. `CarRentalRule` uses the `vehicle` category filter. From that business rule, you can see all the classes that are assigned the `vehicle` category, which in this case, is the `Car` class.

If you then define a category that is named “sales” and assign it to a `Customer` class, the `Customer` class is not visible in the rule editor vocabulary list for the `CarRentalRule` action rule. If you try to use the `Customer` class in the business rule anyway, a warning is raised.

## Category semantics

- Predefined category “any” is set on all business elements
- When no category is defined (including “any”), the element cannot be used in the rules

| If the category of BOM element is: | And the rule category filter is: | Then, in rule editor, the element is: |
|------------------------------------|----------------------------------|---------------------------------------|
| any                                | any (or another category)        | visible                               |
| any (or another category)          | any                              | visible                               |
| category1<br>category2             | category1                        | visible                               |
| category1<br>category2             | category3<br>category4           | hidden                                |
| Empty (no category)                | any (or another category)        | hidden                                |
| any (or another category)          | Empty (no category)              | hidden                                |

Customizing rule vocabulary with categories and domains

© Copyright IBM Corporation 2019

Figure 8-6. Category semantics

By default, the predefined category “any” is set on all business elements, meaning that all business elements can be used in all business rules that are linked to that BOM.

When you use categories, you must set them in three places:

- In the rule project
- On the BOM element
- In the rule

## 8.2. Defining domains

## Defining domains

Customizing rule vocabulary with categories and domains

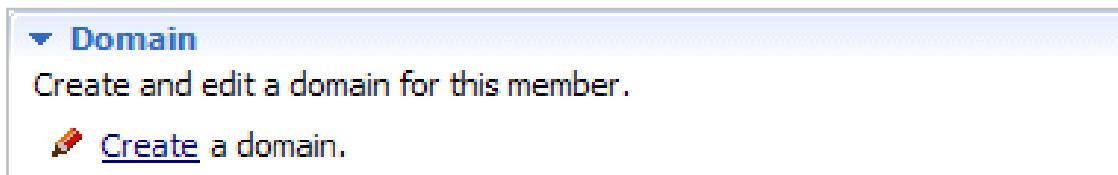
© Copyright IBM Corporation 2019

*Figure 8-7. Defining domains*



## Domains section

- BOM elements can be associated to domains
- A domain places a restriction on the values that BOM members can have
  - You can set a domain on classes, attribute types, method return types, and arguments



Customizing rule vocabulary with categories and domains

© Copyright IBM Corporation 2019

Figure 8-8. Domains section

Another feature that helps you customize the BOM vocabulary for rule authors is the use of *domains*. You define domains in the BOM editor.

Domains restrict the values of the BOM elements. Domains can be set on classes, attribute types, method return types, and arguments.

Domains can be static, dynamic, enumerated, or complex, and include these types:

- **Literals:** Specifies a list of values, for example, {1, 2, 3}
- **Static References:** Specifies a list of references to constants, for example: {static GroupA, static GroupB, static GroupC}
- **Bounded:** Specifies an interval between two bounding values, for example: ]0, 120]
- **Collection:** Specifies the cardinality and the type of collection elements, for example: 0, \* class Customer
- **Dynamic:** Specifies a list of values that some code execution sets dynamically
- **Other:** Domains that are defined with regular expressions syntax

Domains help business users as they edit rule artifacts. When they work in the rule editors, code completion in the editor can propose valid values. The editors also show errors and warnings if they do not use the correct values as specified in the domain.

## 8.3. Static domains

## Static domains

Customizing rule vocabulary with categories and domains

© Copyright IBM Corporation 2019

*Figure 8-9. Static domains*

## Domains: Literals

- A domain of type Literals is defined as an enumeration of literals
- Examples:
  - {1, 2, 3}: The possible value is any of the 1, 2, or 3 literals
  - {A, B, C, D}: The possible value is any of the A, B, C, or D literals
- You can define a domain of type Literals on any attribute of a primitive type

Figure 8-10. Domains: Literals

A literal domain specifies a list of values, such as {1, 2, 3}. You can use literal domains on any attribute of a primitive type.

## Domains: Static references

- A domain of static references is defined as an enumeration of references to constants
- Example:
  - {static GroupA, static GroupB, static GroupC}: The possible value is one of the three constants that are listed explicitly here
- You can define a domain of static references on an attribute of type A by using the static attributes of the class A

Figure 8-11. Domains: Static references

Static references specify a list of references to constants. For example, you can have a set of static groups: {static GroupA, static GroupB, static GroupC}

## Domains: Bounded

- A bounded domain is defined as an interval between two bounding values (included, or not)
- Example:
  - [0, 120]: The possible value is an integer value greater than or equal to 0, and less than or equal to 120
- If the domain has a missing bound, specify that missing bound with an asterisk (\*)
  - [0, \*]: The possible value is an integer value greater than or equal to 0
  - [\* , 0]: The possible value is an integer value less than or equal to 0
- Examples:
  - [0, \*]: The possible value is an integer value greater than or equal to 0
  - [\* , 0]: The possible value is an integer value less than or equal to 0

Figure 8-12. Domains: Bounded

Bounded domains specify an interval between two bounding values.

## Domains: Collection

- A collection domain is defined as a type of elements and a cardinality
- Examples:
  - 0,1 class Loan: The possible value is made of zero or one object of the BOM class Loan
  - 0,\* class Customer: The possible value is made of any number of (zero or more) objects of the BOM class Customer
- Similarly, to a bounded domain, use an asterisk (\*) to indicate that there is no upper bound to the number of elements

Figure 8-13. Domains: Collection

Collection domains specify *cardinality* and *type* of collection elements. As shown here, you can have zero or one of class Loan, or zero or many of class Customer.

## Domains: Other

- You can create domains of other types by using:
  - Regular expressions (patterns)
  - Enumerations of values
  - Intersections
  - A combination of these techniques
- Create a domain of type Other to support a complex domain that comes from the XML binding
- Examples:
  - "t\*g": The possible value is any string that starts with a "t" and ends with a "g"
  - For example, "tag", "training"

*Figure 8-14. Domains: Other*

In cases where the previous domain types do not meet your requirements, you can define a custom domain of type Other, according to the types of values that you need to include.

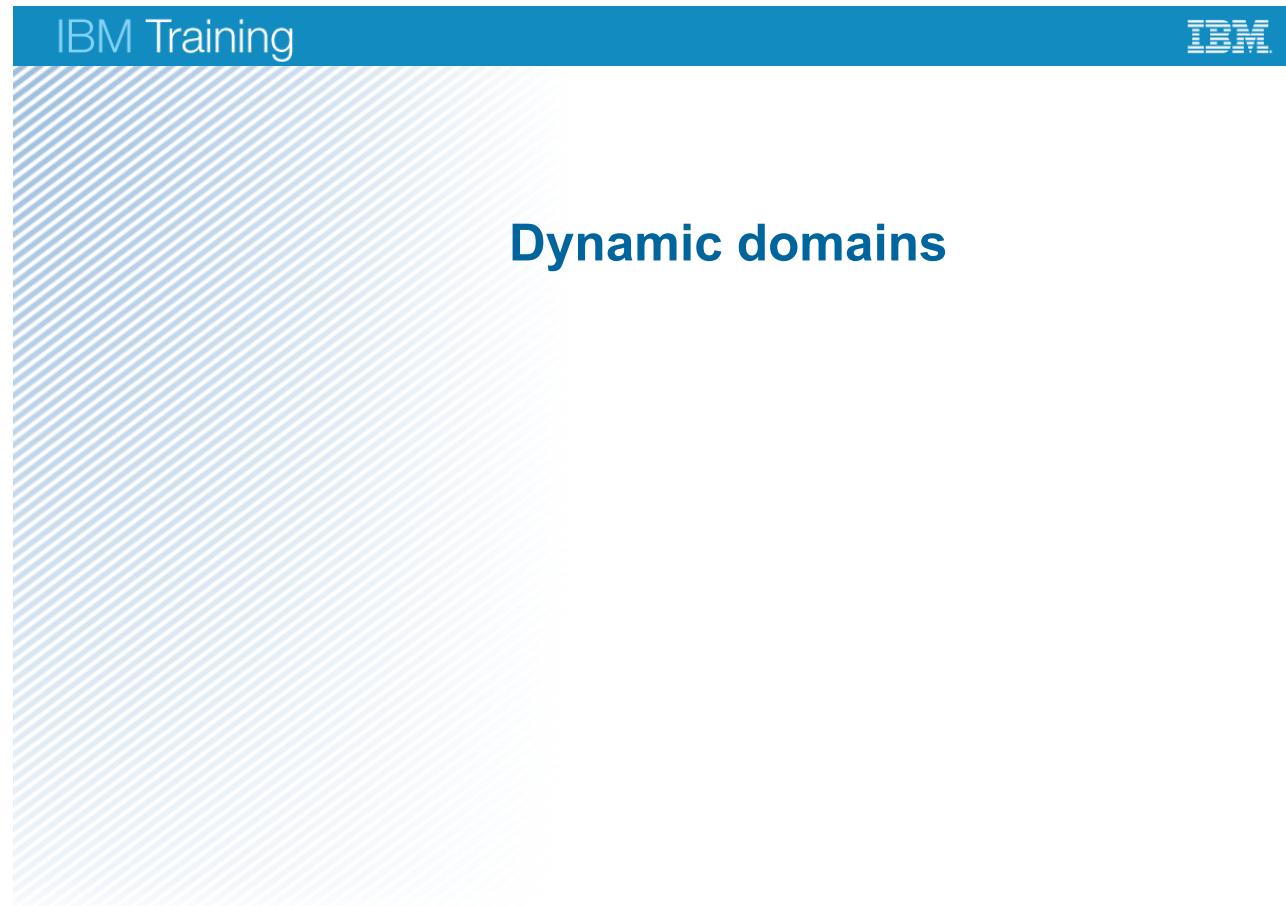
These can be regular expressions, enumerations of values, intersections, or a mix of these techniques. You can also use this type of domain to support complex domains that come from XML binding.

For example, consider this definition:

```
({1, 3, 5, 7, 9}, [0,6])
```

With such a domain definition, the possible value is any two-digit number where the first digit is 1, 3, 5, 7, or 9, and the second digit is in the range 0–6. Some examples would be 11, 35, and 73.

## 8.4. Dynamic domains



Customizing rule vocabulary with categories and domains

© Copyright IBM Corporation 2019

*Figure 8-15. Dynamic domains*

## Domains: Dynamic (1 of 2)

- A domain of type Dynamic is defined as an enumeration of values that the execution of some plug-in code dynamically sets
- As opposed to domains of type Dynamic, the domains of type Literals, Static References, Bounded, Collection, and Other are said to be *static*
  - The values for these domains are statically defined in the BOM editor

Customizing rule vocabulary with categories and domains

© Copyright IBM Corporation 2019

Figure 8-16. Domains: Dynamic (1 of 2)

As their name indicates, the values of dynamic domains are set as a result of code execution. So domain values can change during rule execution.

Similar to other types of domains, you create a domain of type `Dynamic` in the BOM editor.



### Note

#### IBM ODM on Cloud

ODM on Cloud supports dynamic domains with Microsoft Excel. It does not support custom data providers for dynamic domains.

## Domains: Dynamic (2 of 2)

- The general mechanism to set up a dynamic domain is based on a plug-in that reads the values of the dynamic values from the appropriate source
- To set up a dynamic domain:
  - Create a plug-in project
  - Implement the required API interface in the plug-in project
  - Create an extension point that is based on the plug-in
  - Deploy the extension point to Rule Designer and Decision Center
- This general mechanism is applicable to all types of source
- If the source is a Microsoft Excel file, the plug-in and the associated extension point are predefined and predeployed in Rule Designer and Decision Center

Figure 8-17. Domains: Dynamic (2 of 2)

Dynamic domains are based on a plug-in, and requires setting up a domain of type Dynamic. Operational Decision Manager provides a predefined plug-in and the associated extension points to support Microsoft Excel spreadsheet as the source for stored domain values.

During the exercise, you see how to work with dynamic domains that are based on Excel and how to ensure that business users can access those domain values from Decision Center.

If you do not uses Excel as the source for values, you must create the plug-in that reads from that source. You must also create the associated extension points for Rule Designer and Decision Center, and deploy the extension points to both environments.

## Dynamic domains: Microsoft Excel source (1 of 4)

- To define a dynamic domain with a Microsoft Excel spreadsheet as the source:
  1. Create the spreadsheet with the domain properties: values, labels, descriptions, and BOM-to-XOM mappings
  2. Add the spreadsheet to the `resources` folder of the project that defines the BOM
  3. Use the BOM editor to map the domain properties to the columns of the spreadsheet
  4. Publish the rule project that defines the dynamic domain to Decision Center

Figure 8-18. Dynamic domains: Microsoft Excel source (1 of 4)

You can use the **resources** folder of the rule project to hold additional files that must be shared between Rule Designer and Decision Center.

When you create the spreadsheet that defines the dynamic domain values, you save it to this **resources** folder.

Rule Designer has no specific menus to populate this folder with the Microsoft Excel spreadsheet that defines the dynamic domain. Instead, you can use any appropriate Windows functions (such as Copy, Cut, and Paste) or Eclipse menus (such as **New > File**, **New > Folder**, and **Import**).

When you publish the rule project to Decision Center, that folder and its contents are also published and become available to business users.

## Dynamic domains: Microsoft Excel source (2 of 4)

- Some properties must be defined on the BOM class to retrieve the information from the Excel file
- To map the properties correctly, the Excel file must have the following structure:
  - One row for each value of the dynamic domain
  - Three mandatory columns in each row
  - Optional columns in each row
  - No merged cells

Figure 8-19. Dynamic domains: Microsoft Excel source (2 of 4)

Some properties must be defined on the BOM class to retrieve the information from the Excel file. And to map the properties correctly, the Excel file must have the correct structure. You see how to do this during the exercise.

## Dynamic domains: Microsoft Excel source (3 of 4)

- Value column: Contains the values of the domain
- BOM to XOM column: Contains the BOM-to-XOM mapping of the value
- Label column: Contains the verbalization of the value
  - This label is the name that is displayed for the value when authoring rules
- Documentation column: *Optional*, contains a description of the value

|   | A              | B             | C               |
|---|----------------|---------------|-----------------|
| 1 | Value          | BOM to XOM    | Label           |
| 2 | Administrative | return "A01"; | Administrative  |
| 3 | Compensatory   | return "B34"; | Compensatory    |
| 4 | Educational    | return "B45"; | Educational     |
| 5 | FamilyPersonal | return "C56"; | Family/Personal |
| 6 | JuryDuty       | return "V78"; | Jury Duty       |
| 7 | Military       | return "B89"; | Military        |
| 8 | Overtime       | return "F23"; | Overtime        |
| 9 | Pregnancy      | return "V12"; | Pregnancy       |

Figure 8-20. Dynamic domains: Microsoft Excel source (3 of 4)

The domain spreadsheet must have the structure and columns that you see here.

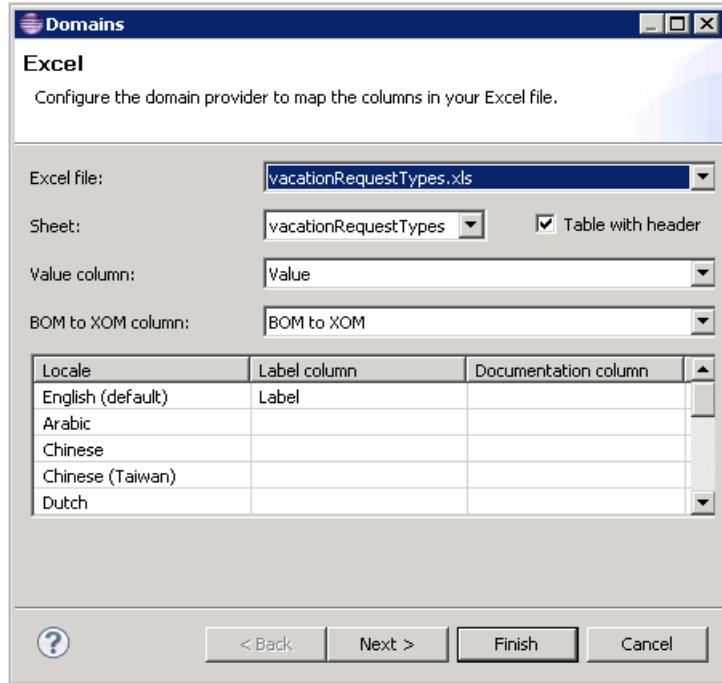
The Documentation column is optional for all locales, including the default one, which is the locale of your Eclipse application.

You can optionally add label and documentation columns for locales other than the default one. The values for the other locales are used when changing the locale of your Eclipse application.

You can also add columns other than the four predefined columns when you want to use custom properties for the values in your dynamic domain. The custom properties that you define through the Domains wizard apply to all the values in your dynamic domain.

## Dynamic domains: Microsoft Excel source (4 of 4)

- You create the dynamic domain in the Domain section of the BOM class
- With the Domains wizard, you must:
  1. Select the Microsoft Excel file
  2. Select the sheet that defines the domain
  3. Select the column for the values
  4. Select the column for the BOM-to-XOM mapping
  5. For the default locale, select the column for the Labels, and optionally the Documentation
  6. Optionally: Do the same for other locales



Customizing rule vocabulary with categories and domains

© Copyright IBM Corporation 2019

*Figure 8-21. Dynamic domains: Microsoft Excel source (4 of 4)*

After you create the dynamic domain, the values of the domain are visible in the BOM editor and also become available in the rule editors.

If you define labels or documentation for other locales, you must update the BOM for each locale. To do so, you must restart Rule Designer in the locale to update, and synchronize the BOM with the values in the Microsoft Excel file.

## Enumerated versus complex domains

- Domains of type Literals, Static References, and Dynamic are called *enumerated* domains
  - Enumerated domains are enforced in BAL-based rule artifacts
- Domains of type Bounded, Collection, and Other, not defined with an enumeration, are called *complex* domains
  - Complex domains are not enforced in BAL-based rule artifacts

Figure 8-22. *Enumerated versus complex domains*

As the adjective “enumerated” suggests, enumerated domains are domains that are defined with a (static or dynamic) enumeration of values.

One note about the difference between enumerated values and complex values in domains. If you try to write a rule that is based on an enumerated domain type, the rule does not compile if the value is outside of the domain. However, if you use complex domains, the values are not enforced at the rule level. You might receive warnings that the rule is not applicable, but the rule will compile. The semantic check that is done at the action rule level is primitive and does not detect complex patterns of incorrect usage that involve operators other than `is` or `is not`.

## Automatic creation of domains

- When you create the BOM from the XOM, Rule Designer automatically creates some domains
  - XOM `public`, `static`, and `final` attributes that are typed to the declaring class are considered as the elements of a BOM domain of type `Static References`
  - XOM members that are of generic types, like `Vector<C>` or `Array<C>`, are considered to be a BOM domain of type `Collection` of the `C` class

Customizing rule vocabulary with categories and domains

© Copyright IBM Corporation 2019

Figure 8-23. Automatic creation of domains

When you create the BOM from an existing XOM, some domains can be automatically created for you.

For example, if you have a XOM attribute that is defined as a vector or an array, the corresponding BOM attribute associate a domain of type `Collection` with that element. However, a BOM element that is generated from a XOM member of type `Java Collection` is not automatically generated as a BOM domain of type `Collection`. But you can set a domain of type `Collection` on these members so that they are automatically considered as a collection in the business rules.

You can create, add, and remove methods for the items in the `Collection` domain by using the BOM editor.

## 8.5. Updating dynamic domains in Decision Center

## Updating dynamic domains in Decision Center

Customizing rule vocabulary with categories and domains

© Copyright IBM Corporation 2019

*Figure 8-24. Updating dynamic domains in Decision Center*

## Dynamic domains in Decision Center

- Can modify values of dynamic domains in the Decision Center Business console
- Dynamic domain must be stored in an Excel file
  - Can edit the Excel file to change domain values and replace the file that is stored in Decision Center
  - The decision service BOM is updated from the values in the Excel file
- Changes to the BOM in Decision Center can be synchronized with the BOM in Rule Designer
- Work with dynamic domains through the **Decision Artifacts** tab and the **Model** tab
  - **Decision Artifacts** tab: Access, update, and delete dynamic domain Excel files as decision service resources
  - **Models** tab: View domain details and apply changes to the BOM after domain updates

[Customizing rule vocabulary with categories and domains](#)

© Copyright IBM Corporation 2019

Figure 8-25. Dynamic domains in Decision Center

You can work with dynamic domains in Decision Center.

When a rule project includes a dynamic domain that is defined in an Excel file, the file is stored as a project resource. You can change the domain values in the file, and then upload the updated file to Decision Center. You can then update the decision service BOM with the new values from the Excel file. You can also synchronize the changes to the Decision Center BOM with the BOM in Rule Designer.

When you work with a dynamic domain in Decision Center, you use the **Decision Artifacts** tab and the **Model** tab.

### Decision Artifacts tab

When you import a dynamic domain Excel file in a Rule Designer decision service project, it is stored in the **Resources** folder. When you publish the decision service to Decision Center, the dynamic domain Excel file is also located in the rule project **Resources** folder.

In Decision Center, you can access the dynamic domain Excel file from the **Decision Artifacts** tab. Make sure that you can view resource artifacts. The default view in the Business console **Decision Artifacts** tab is set to show only **Rules and Decision Tables**. To view resources, click the **Types** menu and either select **All Types** (to see all decision artifacts) or select **Resources** (to see resources in addition to business rules and decision tables).

To view rule project resources, use the navigator pane to expand the rule project that includes the dynamic domain Excel file, and click the **Resources** folder to load the project resources in the main pane. You can then download the Excel file, upload a file (or new version of a file), or delete a file.

### **Model** tab

The **Model** tab lists the dynamic domains that are used in the decision service. On this tab, you can view the domain values, and other details, such as the source Excel file and the project where the Excel file is stored.

If you update the domain by modifying and replacing the Excel file, you use the **Model** tab to apply the changes from the Excel file to the Decision Center BOM.



## Updating the dynamic domain (1 of 4)

- In the **Decision Artifacts** tab, find the Excel file that the decision service uses as a source for the dynamic domain
  - Click the **Resources** folder to load the list of project resources
  - Click the Excel file that you want to work with to open the file preview

| Name                            | Last Changed By | Last Changed On    |
|---------------------------------|-----------------|--------------------|
| vacationRequestTypes-1.xls      | rtsAdmin        | March 3, 2017 9... |
| <b>vacationRequestTypes.xls</b> | rtsAdmin        | March 3, 2017 1... |

Customizing rule vocabulary with categories and domains

© Copyright IBM Corporation 2019

Figure 8-26. Updating the dynamic domain (1 of 4)

To update a dynamic domain in Decision Center:

- Download the dynamic domain Excel file.
- Modify the file.
- Upload the modified Excel file.
- Apply the changes from the file to the BOM.

To view the dynamic domain Excel file, click the **Resources** folder. In the Resources pane, click the file that you want to access.



## Updating the dynamic domain (2 of 4)

- In the file preview, click the file name to download the file



- After you modify the file in Excel, replace the original Excel file in Decision Center with the updated file
- Two options for replacing the file:
  - In the Resources pane, delete the original file and upload the new version



- In the file preview pane, replace the original file by editing the file options and refreshing it with the new version



Customizing rule vocabulary with categories and domains

© Copyright IBM Corporation 2019

Figure 8-27. Updating the dynamic domain (2 of 4)

In the file preview that opens, click the file name to download the Excel file from Decision Center to your computer. You can now modify the domain values in the Excel file.

After you finish changing the domain values, you can replace the original Excel file with your updated file. You can replace the original file by using one of two methods:

- Go to the main Resource pane to delete the original file, then upload the new version.
- Go to the file preview pane and edit the file options. You can then refresh the file by selecting the updated file and saving it.

## Updating the dynamic domain (3 of 4)

- In the **Model** tab, you can review the domain and the changes to the domain

| Domain                     | Changes                    | Provider                 | Project              |
|----------------------------|----------------------------|--------------------------|----------------------|
| domains.VacationRequestTyp | domains.VacationRequestTyp | vacationRequestTypes.xls | trucksAndDrivers-bom |
| Administrative             | - Administrative           |                          |                      |
| Educational                | Educational                |                          |                      |
| Family/Personal            | Family/Personal            |                          |                      |
| Jury Duty                  | Jury Duty                  |                          |                      |
| Military                   | Military                   |                          |                      |
| Other Vacation             | Other Vacation             |                          |                      |
| Overtime                   | Overtime                   |                          |                      |
| Pregnancy                  | Pregnancy                  |                          |                      |
| Recognition                | Recognition                |                          |                      |
| Sick                       | Sick                       |                          |                      |
| Strike                     | Strike                     |                          |                      |
| -                          | + Training                 |                          |                      |
| Vacation                   | Vacation                   |                          |                      |
| Volunteer Fire and Rescue  | Volunteer Fire and Rescue  |                          |                      |
| Without Pay                | Without Pay                |                          |                      |

Customizing rule vocabulary with categories and domains

© Copyright IBM Corporation 2019

Figure 8-28. Updating the dynamic domain (3 of 4)

After you modify and replace the dynamic domain Excel file, you can see the changes to the domain values in the **Model** tab.

Expand the **Domain** field to see the original domain values, and expand the **Changes** field to see the updated values from the Excel file.

In this slide, you can see that the **Administrative** vacation request type value was removed, and the **Training** vacation request type was added.

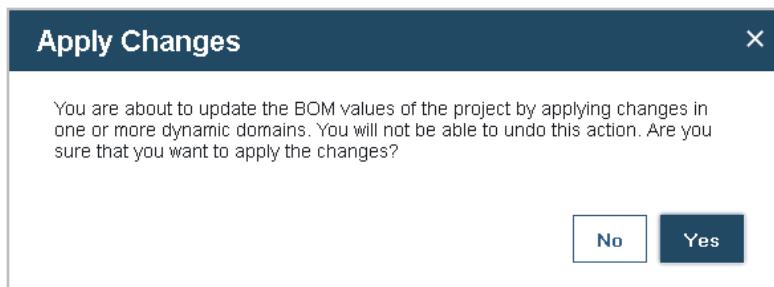


## Updating the dynamic domain (4 of 4)

- To apply the changes from the updated domain to the Decision Center BOM:
  - Select the domain that you want to update and click **Apply changes**

The screenshot shows a user interface for managing changes. At the top, there is a button labeled "Apply changes" with an orange border. Below it, there is a section titled "Changes" with a "Domain" checkbox checked. To the right, there are columns for "Provider" (showing "vacationRequestTypes.xls") and "Project" (showing "trucksAndDrivers-bom"). A tree view shows a checked node "domains.VacationRequestTyp" under "domains.VacationRequestTyp".

- In the Apply Changes window, click **Yes** to confirm the update to the BOM



Customizing rule vocabulary with categories and domains

© Copyright IBM Corporation 2019

Figure 8-29. Updating the dynamic domain (4 of 4)

To update the Decision Center BOM, select the **Domain** check box and click **Apply changes**. The Apply Changes windows opens and prompts you to confirm the updates to the BOM. Click **Yes** to confirm the changes.

After the update is complete, you can author rules with the updated domain values.

## Unit summary

- Simplify rule authoring by using categories
- Define domains

*Figure 8-30. Unit summary*

## Review questions

1. True or False: Categories simplify rule authoring by reducing the number of items in the rule editor vocabulary lists.
2. True or False: Domains are defined in project properties.



Figure 8-31. Review questions

Write your answers here:

1.

2.

## Review answers

1. True or False: Categories simplify rule authoring by reducing the number of items in the rule editor vocabulary lists.  
The answer is True.
  
2. True or False: Domains are defined in project properties.  
The answer is False. Domains are defined in the BOM editor.



Figure 8-32. Review answers

## Exercise: Working with static domains

Customizing rule vocabulary with categories and domains

© Copyright IBM Corporation 2019

*Figure 8-33. Exercise: Working with static domains*

## Exercise introduction

- Create various types of static domains
- Use domains in rules



Customizing rule vocabulary with categories and domains

© Copyright IBM Corporation 2019

*Figure 8-34. Exercise introduction*

## Exercise: Working with dynamic domains

Customizing rule vocabulary with categories and domains

© Copyright IBM Corporation 2019

*Figure 8-35. Exercise: Working with dynamic domains*

## Exercise introduction

- Create dynamic domains in Microsoft Excel spreadsheets
- Update and use dynamic domains in rules
- Access and update dynamic domains in Decision Center
- Synchronize dynamic domains between Rule Designer and Decision Center



Customizing rule vocabulary with categories and domains

© Copyright IBM Corporation 2019

*Figure 8-36. Exercise introduction*

# Unit 9. Working with queries

## Estimated time

00:45

## Overview

This unit explains how to use search and query tools with rule artifacts.

## How you will check your progress

- Review
- Exercise

## Unit objectives

- Use search features and queries to identify rules according to specific criteria
- Define semantic queries according to rule behavior
- Use queries to create ruleset extractors

## Topics

- Searching for rule artifacts
- Querying rules
- Extracting rulesets
- Queries in Decision Center

Working with queries

© Copyright IBM Corporation 2019

*Figure 9-2. Topics*

## 9.1. Searching for rule artifacts



Working with queries

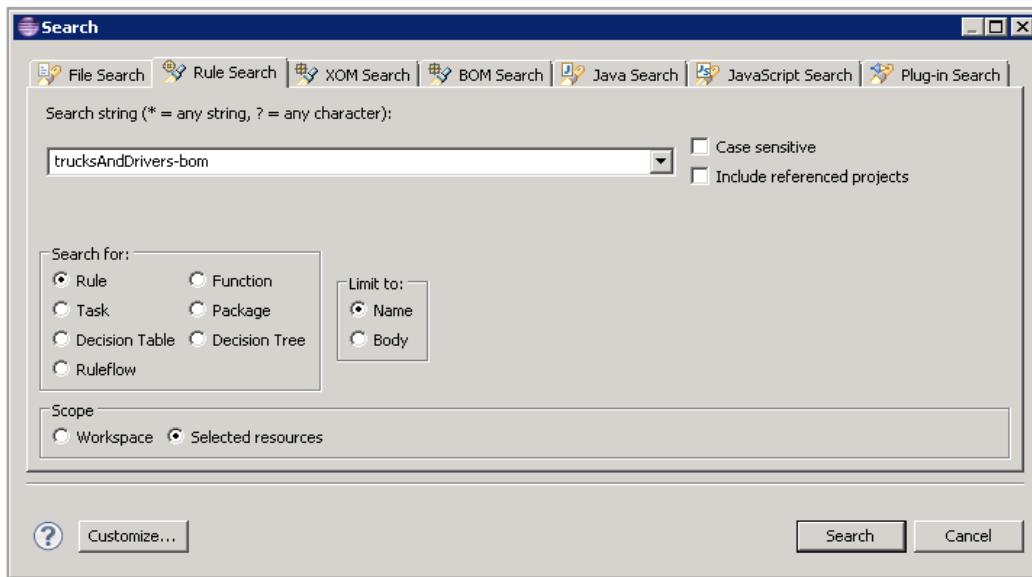
© Copyright IBM Corporation 2019

*Figure 9-3. Searching for rule artifacts*



## Search

- Search XOM and BOM classes by their name
- Search rule artifacts by their name or body content
- Integrated into Eclipse search



Working with queries

© Copyright IBM Corporation 2019

*Figure 9-4. Search*

With Rule Designer search tools, you can search packages, rule files, the BOM, and the XOM. You can also search for dependencies between rules, which is helpful to find which rules might impact other rules when they execute or which rules might be impacted as a result of other rules being executed.

## Search for rule dependencies

- Search for rules that the execution of other rules affects
- Search for rules that affect the execution of other rules

Working with queries

© Copyright IBM Corporation 2019

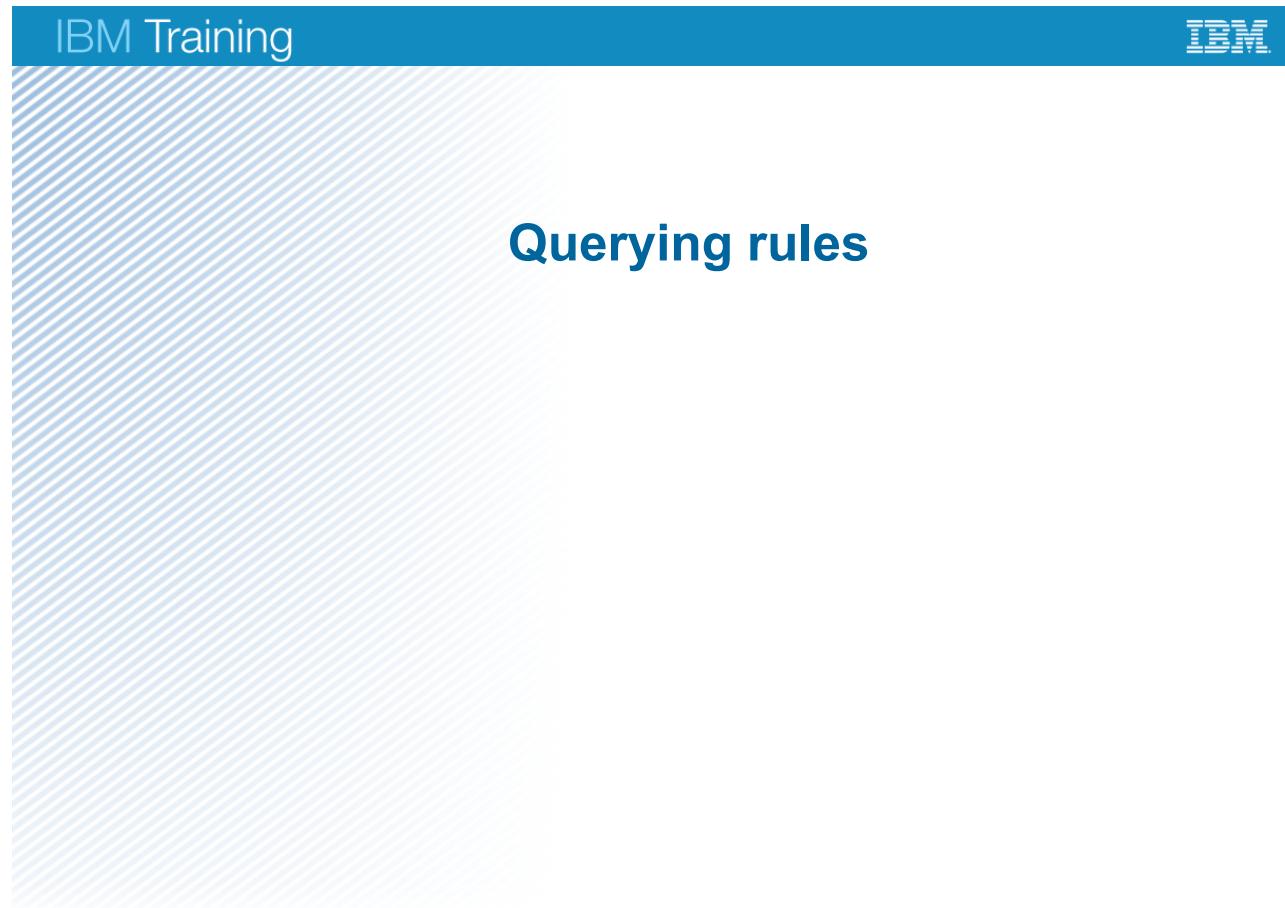
*Figure 9-5. Search for rule dependencies*

To search for rule dependencies, right-click the rule of interest, click **Find Rule Dependencies**, and select what you want to search for.

- **Rules which may enable or disable this rule**
- **Rules which may be enabled or disabled by this rule:**
- **Ruleflows that may select this rule**

When you do such a search, you run a predefined **query** on the behavior of the rules.

## 9.2. Querying rules



Working with queries

© Copyright IBM Corporation 2019

*Figure 9-6. Querying rules*

## Introducing queries

- In addition to basic searches based on static content or predefined queries, you can define your own queries
- You can create queries in Rule Designer and in Decision Center

Working with queries

© Copyright IBM Corporation 2019

*Figure 9-7. Introducing queries*

Both Rule Designer and Decision Center provide predefined queries. They also provide tools for you to build your own queries.

## When queries are useful (1 of 2)

- When multiple users collaborate on the rule repository to:
  - Find only rules that you wrote
  - Find all rules that someone added within a certain time period
  - Find all rules that contain some specific text in their documentation
- Evaluate impact of changes to the object model or changes to the rules
  - Example: Before adding a rule that modifies a certain object to a rule project, query for all rules that use that object to assess the impact of this rule
- Complete actions that are based on the results of a search
  - With the keyword `Do`, you can define an action that applies to the result of the search
  - Example:
    - Find all business rules such that the status of each business rule is new
    - Do set the status of each business rule to validated

Figure 9-8. When queries are useful (1 of 2)

Queries are particularly useful when you have multiple users who are collaborating on the same rule projects.

Semantic queries can identify rule behavior. They are also useful during rule analysis because they can evaluate the impact of change to the BOM or to the rules.

## When queries are useful (2 of 2)

- Complete actions that are based on the results of a search
  - With the keyword `Do`, you can define an action that applies to the result of the search
  - Example:  
Find all business rules such that the status of each business rule is new  
Do set the status of each business rule to validated
- Ruleset extraction
  - Use specific criteria to determine the rules that you require to create the ruleset
- Debugging
  - Find rule artifacts that are listed in trace logs or error messages

*Figure 9-9. When queries are useful (2 of 2)*

You can also run queries that apply actions to the query results. You build these types of queries with the keyword `Do`.

In the example here, the query finds all rules with the status value of `new`. After you run the query, you can check through the list of rules that are returned, and then apply the `Do` statement. All the rules will have their status changed to `validated`.

This is a useful feature when you need to make the same change to many rules.

## Business Query Language (BQL)

- You create queries in the Query editor by using the Business Query Language (BQL)
- BQL is derived from BAL
- BQL is tailored for querying rules
- Like BAL, BQL uses a natural language syntax

Figure 9-10. Business Query Language (BQL)

Queries are defined in a query-specific language that is called BQL. It is based on BAL so you use most of the same constructs in the queries that you use in rules.

## Query conditions

- To write a query condition, select the project element type that you want to query, and refine with filters
- Queried elements types can be:
  - Action rules
  - Decision tables
  - Decision trees
  - Rule package
  - And others
- Filters are introduced with “such that”
- Filters can be on:
  - Properties
  - Definition
  - Behavior

Working with queries

© Copyright IBM Corporation 2019

*Figure 9-11. Query conditions*

When you write a query condition, you are prompted to select the type of artifact that you are looking for.

You see in the query editor that you can search for rules, decision tables, rule packages, technical rules, ruleflows, templates, functions, variables, or variable sets.

By default, queries are run on all rules.

## Example of rule query conditions

- Find all ruleflows such that each ruleflow is in package "accounts"
- Find all business rules such that the effective date of each business rule is after 31/12/2010

Working with queries

© Copyright IBM Corporation 2019

Figure 9-12. Example of rule query conditions

After you select the type of artifact that you want to search, you can add filters by using the `such that` statement. The `such that` phrase adds additional conditions and limits the results that are returned by the query.

### Examples:

Find all ruleflows such that each ruleflow is in package "accounts"

Find all business rules such that the effective date of each business rule is after 31/12/2010

With the `such that` statement, you can filter on:

- Properties
- Definitions
- Behavior

## Filter on properties

- You can query rules with filters on any rule property, including:
  - User-defined properties
  - Classes or data members that are referenced in rules
  - The full text of the rule (by using a text string)
- Example:  
Find all business rules such that the status of each business rule is new

Figure 9-13. Filter on properties

You can run queries that are based on any rule property, including user-defined properties, classes, data members, and methods that are referenced in rules, or actual text strings that are used in the rule statement.

## Filter on definitions

- You can query rules with filters on project element definitions to find rules that use or modify the value of a member or call a method
- Use the following predicates (non-exhaustive list):
  - uses the value of
  - uses the phrase
  - uses the phrase ... where
  - modifies the value of
  - is using BOM class
  - is using BOM member

### • Example

Find all action rules

such that each action rule uses the phrase [set the credit score of 'a borrower' to 'a number', where 'a number' equals 100]

Figure 9-14. Filter on definitions

You can run queries to find rules that read or modify the value of a BOM member, or that call a method.

You set such filters by using query predicates, such as the ones that are listed on this slide. You can also use negations of these predicates.

## Filter on behavior

- You can query rules by filtering on their behavior, its “meaning”, or the result of what it does
- Query on rule conditions with the following predicates:
  - may apply when
  - may become applicable when
  - may lead to a state where
- Query on rule dependencies to identify the links and dependencies between them with the following predicates:
  - may select
  - may enable
  - may disable
  - may be enabled by
  - may be disabled by

*Figure 9-15. Filter on behavior*

Semantic queries filter rules according to their behavior.

To find rules that become applicable when a certain expression is true, or rules that can lead to a certain condition upon execution, use these query predicates:

- may apply when
- may become applicable when
- may lead to a state where

To identify the links and dependencies between rules, or to verify how the execution of a rule affects the applicability or execution of other rules, use these query predicates:

- may select
- may enable
- may disable
- may be enabled by
- may be disabled by

## Query actions

- You can add actions that would apply to the result of the query
- Actions are written after the query conditions
- You introduce actions with the keyword **Do**
- By default, actions are not run when you run the query
- To run the actions on the query results, you must select **Run query actions** after you run the query
- **Example:**

Find all business rules such that the status of each business rule is new

Do set the status of each business rule to validated

Figure 9-16. Query actions

In your queries, you can add actions that are applied to the result of the query.

To specify the actions to complete on the result, add them after the conditions in your query, and introduce them with the following keyword: **Do**

Even if you create a query that has actions, you are not obliged to run these actions on the result of the query. You can do a first run to test the query. After you test your query, you can decide whether you want to apply the actions on the result, or not.

To run the actions on the result of the query, select the **Run query actions** check box when you run the query.

## Queries and ruleset extractor

- Queries are also used to create ruleset extractors
- You apply a ruleset extractor that is based on a query to create a ruleset archive that contains only the rules that the query finds

*Figure 9-17. Queries and ruleset extractor*

Queries can also be used to create a ruleset extractor.

To use a ruleset extractor, you first create a query and then create the ruleset extractor from this query. You apply the ruleset extractor to create your ruleset archive so that the extracted ruleset contains only a restricted number of rule artifacts.

## 9.3. Extracting rulesets



Working with queries

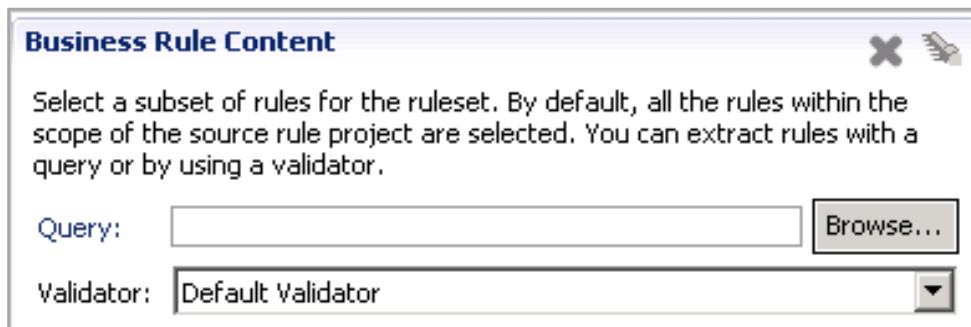
© Copyright IBM Corporation 2019

*Figure 9-18. Extracting rulesets*



## Ruleset archive

- To pass only a subset of rules from a decision operation to the rule engine:
  - Use a query to extract the rules
  - Write a validator class to select the required rules
- Specify the extractor in the decision operation on the **Overview** tab in the **Business Rule Content** section.



Working with queries

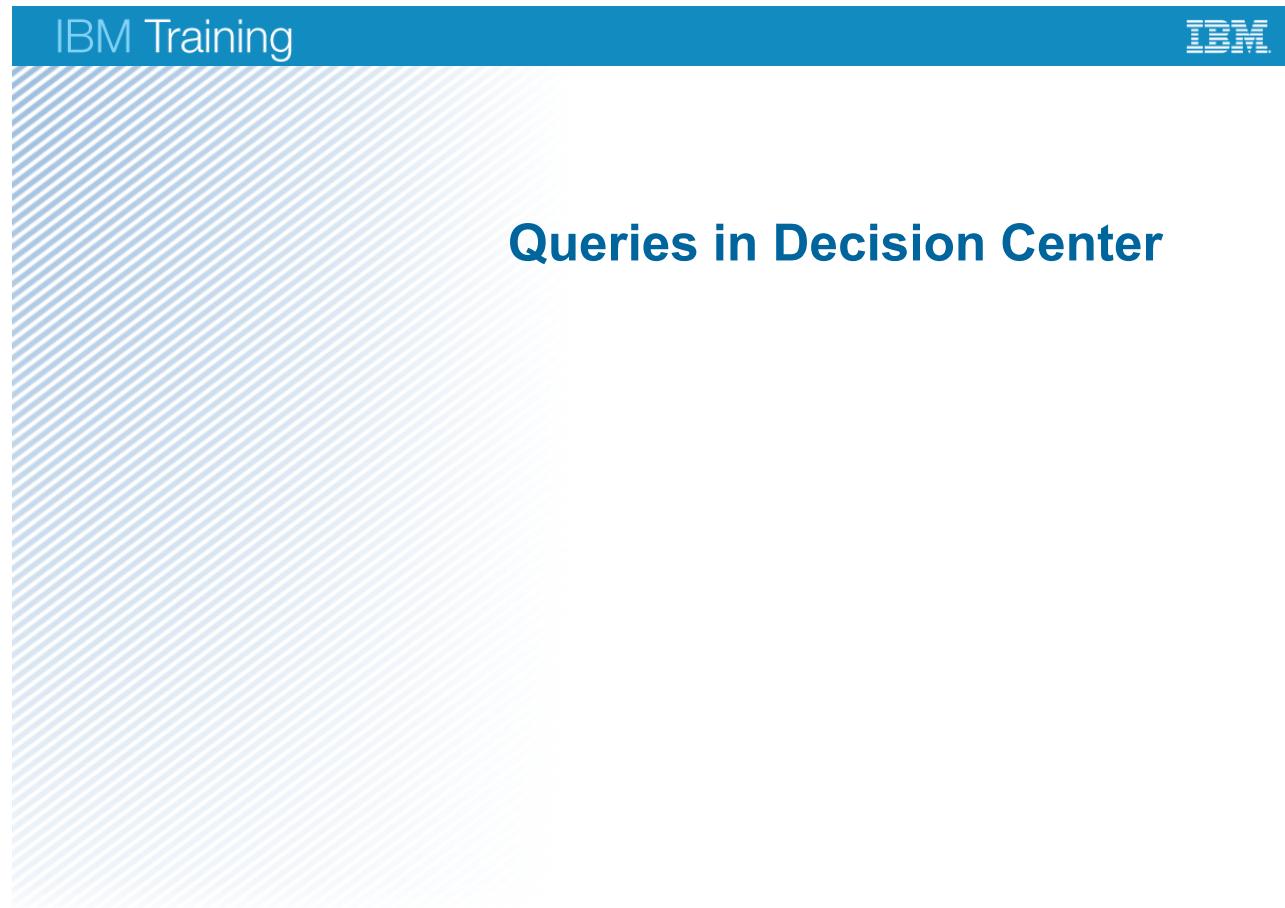
© Copyright IBM Corporation 2019

*Figure 9-19. Ruleset archive*

Rules relating to a specific decision are organized for execution and stored in a ruleset, which you must package into a *ruleset archive* to pass it to the rule engine.

The ruleset archive contains a technical language version of the rules and rule artifacts, and all the supporting data necessary to execute your ruleset, including functions, ruleflows, the BOM, and the BOM-to-XOM mapping.

The default behavior is to include all the rules in all the referenced rule projects. The ruleset extractor restricts the artifacts that are included in the ruleset.



Working with queries

© Copyright IBM Corporation 2019

*Figure 9-20. Queries in Decision Center*

## Queries in Decision Center

- You can run queries in the Decision Center Business console
  - Use **Queries** tab
  - Search for elements in decision services; filter business rules or other project elements
  - Apply actions to query results
  - Queries are run against the current project or other linked projects
- Can synchronize queries between Rule Designer and Decision Center
- Some query constructs are specific to Rule Designer
  - {this} is editable
  - {rule artifact} is using BOM class
  - the parent package of {a rule package}
- Some query constructs are specific to Rule Designer
  - the creator of {this}
  - the last modification date of {this}
  - the group of
- Make sure that queries use appropriate constructs before publishing to Decision Center or synchronizing the decision service

[Working with queries](#)

© Copyright IBM Corporation 2019

*Figure 9-21. Queries in Decision Center*

As in Rule Designer, you can use queries in the Business console to find artifacts in decision services and apply actions to the results.

Queries synchronize between Rule Designer and Decision Center. When you synchronize a decision service project between Rule Designer and Decision Center, the queries in both modules are also synchronized. For example, if you edit a query in Rule Designer and synchronize the decision service project with Decision Center, the query is updated in Decision Center.

Note that certain query constructs are module-specific. That is, some query constructs can be used only in Decision Center or in Rule Designer. If you write queries with module-specific constructs, these queries are synchronized between both Rule Designer and Decision Center. However, you cannot run these queries in the other module. For example, if you write a query that uses a construct specific to Rule Designer, the query appears in the list of available queries, but you cannot run it in Decision Center.

To make sure that a query that is written in one module can be run in the other module, avoid using module-specific query constructs.

For more information about module-specific queries, see the product documentation.

The screenshot shows the 'IBM Training' interface with the 'IBM' logo in the top right. Below it, the title 'Working with queries: Queries tab overview' is displayed in blue. A bulleted list item states: 'In the Decision Center Business console, use the **Queries** tab'. The main area shows a navigation bar with tabs: 'Decision Artifacts', 'Queries' (which is selected and highlighted in blue), 'Tests', 'Simulations', and 'Deployments'. Under the 'Queries' tab, there are two sections: 'Queries List' and 'Query Results'. The 'Queries List' section contains a tree view with a node 'loan-rules' expanded, showing five sub-options: 'may become applicable', 'may lead to a state', 'the priority of the rules', 'the status of a rule', and 'using BOM member'. There are also navigation arrows at the bottom of this list.

Working with queries

© Copyright IBM Corporation 2019

Figure 9-22. Working with queries: Queries tab overview

After you open a decision service in the Decision Center Business console, you can access the **Queries** tab. You can use the **Queries** tab to create and run a query. From that tab, you can also save your query, view the results of a query search, and apply actions to the query results.

Select existing queries

New Query Save Run

Working with queries

© Copyright IBM Corporation 2019

Figure 9-23. Working with queries: Queries List

In Business console, the **Queries** tab has two subtabs: **Queries List** and **Query Results**

The Queries List tab opens by default, with the query editor to write and run queries. You can also select existing queries from the navigation pane.

To create a query, in the main pane, click **New Query**. Type a name for the query in the **Name** field, and type your query in the **Query Expression** field. You can optionally add information in the **Description** field. If you want to save this query, click **Save**. The query is added to the list of queries under the project.

To run a query, go to the main pane and click **Run**.



## Working with queries: Query Results

The screenshot shows the 'Decision Artifacts' interface with the 'Queries' tab selected. The 'Query Results' subtab is active. A toolbar at the top has buttons for 'Apply query actions' and 'Query results'. Below the toolbar is a search bar labeled 'Filter results' with a 'Filter:' input field. The main area displays a table titled 'Query Results: 17 results' with columns: Name, Project, Folder, Last Changed By, and Last Changed On. The table lists three items: 'checkAge' (Project: loan-rules, Folder: validation / borrower, Last Changed By: rtsAdmin, Last Changed On: March 28, 2017), 'checkAmount' (Project: loan-rules, Folder: validation / loan, Last Changed By: rtsAdmin, Last Changed On: March 28, 2017), and 'checkCreditScore' (Project: loan-rules, Folder: eligibility, Last Changed By: rtsAdmin, Last Changed On: March 28, 2017). The 'checkAge' row is currently selected, indicated by a blue border around its cells.

| Name             | Project    | Folder                | Last Changed By | Last Changed On     |
|------------------|------------|-----------------------|-----------------|---------------------|
| checkAge         | loan-rules | validation / borrower | rtsAdmin        | March 28, 2017 3... |
| checkAmount      | loan-rules | validation / loan     | rtsAdmin        | March 28, 2017 3... |
| checkCreditScore | loan-rules | eligibility           | rtsAdmin        | March 28, 2017 3... |

Working with queries

© Copyright IBM Corporation 2019

Figure 9-24. Working with queries: Query Results

When you run a query, the **Query Results** subtab opens, and the query searches through the project for the elements that you specified. When the query finishes running, the **Query Results** subtab shows the results of the query.

You can filter the query results to show only specific elements that match your filter.

If the query has an action part, you can apply those actions. You can always review the results before you decide to apply the actions.

In the toolbar above your query results, click the **Apply the query actions** to perform the actions from the query. By default, all the elements that are retrieved by the query are selected, so the action is performed on all of them. If you select certain project elements among the results, the action is performed only on the selected project elements.

## Unit summary

- Use search features and queries to identify rules according to specific criteria
- Define semantic queries according to rule behavior
- Use queries to create ruleset extractors

Working with queries

© Copyright IBM Corporation 2019

*Figure 9-25. Unit summary*

## Review questions

1. Which of the following features can you use to create queries in the Query Editor to search in packages, rules, the BOM, and the XOM?
  - A. Business Query Language (BQL)
  - B. Rule Query Language (RQL)
  - C. Intellirule Query Language (IQL)
  - D. Both a and b
2. True or False: A query can be used to extract a subset of rules for a decision operation.



Working with queries

© Copyright IBM Corporation 2019

Figure 9-26. Review questions

Write your answer here:

1.

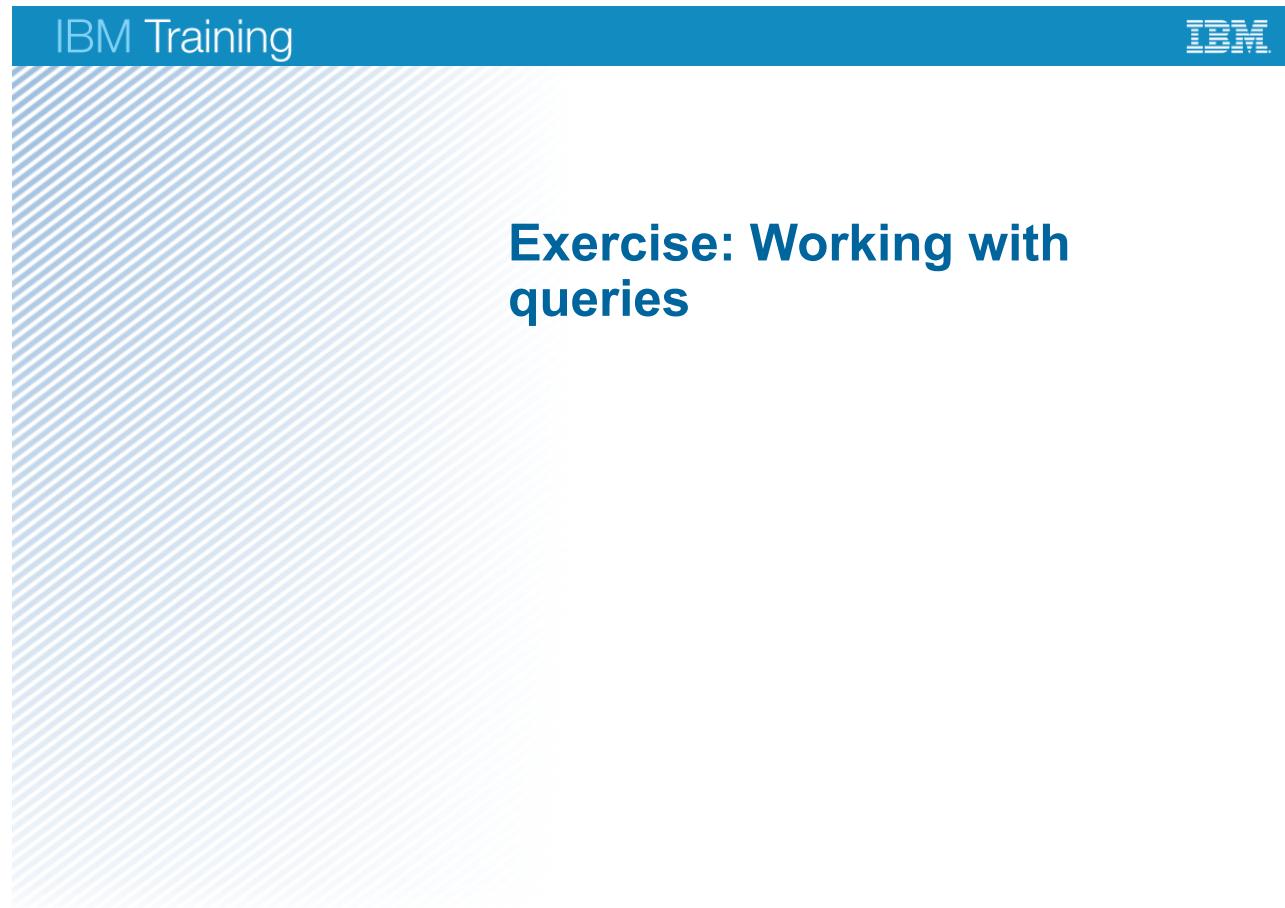
2.

## Review answers

1. Which of the following features can you use to create queries in the Query Editor to search in packages, rules, the BOM, and the XOM?
  - A. [Business Query Language \(BQL\)](#)
  - B. Rule Query Language (RQL)
  - C. Intellirule Query Language (IQL)
  - D. Both a and b

The answer is [A](#).
2. [True](#) or False: A query can be used to extract a subset of rules for a decision operation.  
The answer is [True](#).





Working with queries

© Copyright IBM Corporation 2019

*Figure 9-28. Exercise: Working with queries*

## Exercise introduction

- Search for rule artifacts and find rules according to their dependencies
- Define and run queries and apply actions on query results
- Synchronize queries between Rule Designer and Decision Center



Working with queries

© Copyright IBM Corporation 2019

*Figure 9-29. Exercise introduction*

# Unit 10. Running and debugging

## Estimated time

00:45

## Overview

In this unit, you learn how to verify that the implemented business logic is free of errors.

## How you will check your progress

- Review
- Exercises

## Unit objectives

- Use launch configurations to execute and debug rulesets
- Work with automatic exception handling
- Work with Rule Designer debugging tools

Running and debugging

© Copyright IBM Corporation 2019

*Figure 10-1. Unit objectives*

## Topics

- Working with launch configurations
- Automatic exception handling
- Using Rule Designer debugging tools

Running and debugging

© Copyright IBM Corporation 2019

*Figure 10-2. Topics*

## 10.1. Working with launch configurations

## Working with launch configurations

Running and debugging

© Copyright IBM Corporation 2019

*Figure 10-3. Working with launch configurations*

## Running and debugging decision services

- Before executing rules on your application, you can run and debug rules in Rule Designer to make sure that the rules behave as expected
- You can launch the rulesets in standard mode or in debugging mode
  - Standard: The rules execute, but you cannot suspend or examine the execution
  - Debug: You can suspend, then resume execution, inspect variables, and evaluate expressions
- To run or debug a decision operation, the rule engine needs values for all the `IN` and `IN_OUT` parameters for your ruleset

[Running and debugging](#)

© Copyright IBM Corporation 2019

Figure 10-4. Running and debugging decision services

Ruleset execution involves instantiating a new rule engine (or connecting to an existing one), sending the ruleset to the engine, providing application objects to the rule engine, and then executing the ruleset.

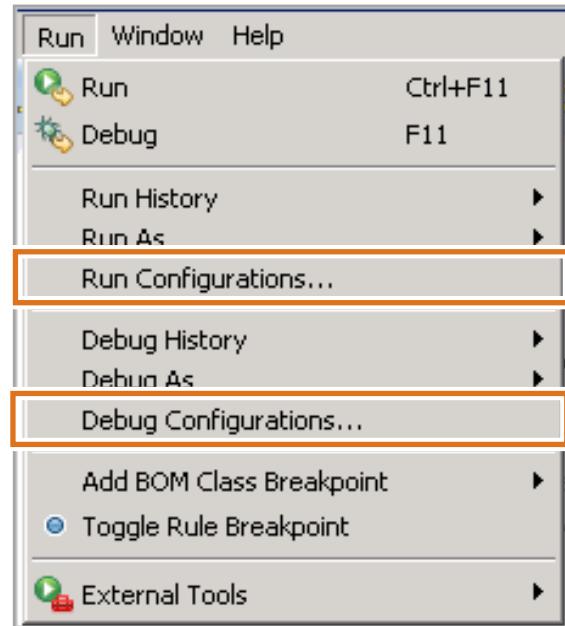
Depending on whether you run the ruleset with or without the debugger, the following terms apply:

- *Run*: When you want to run the rules without debugging and check the results
- *Debug*: When you want to suspend execution and use the debug tools

When you launch rulesets in debug mode, you establish a connection between the debugger client and the rulesets that you are launching.

## Defining launch configuration (1 of 2)

- From the **Run** menu in Rule Designer, create launch configurations to run or debug decision operations
  - No code required
  - Set input parameters manually



Running and debugging

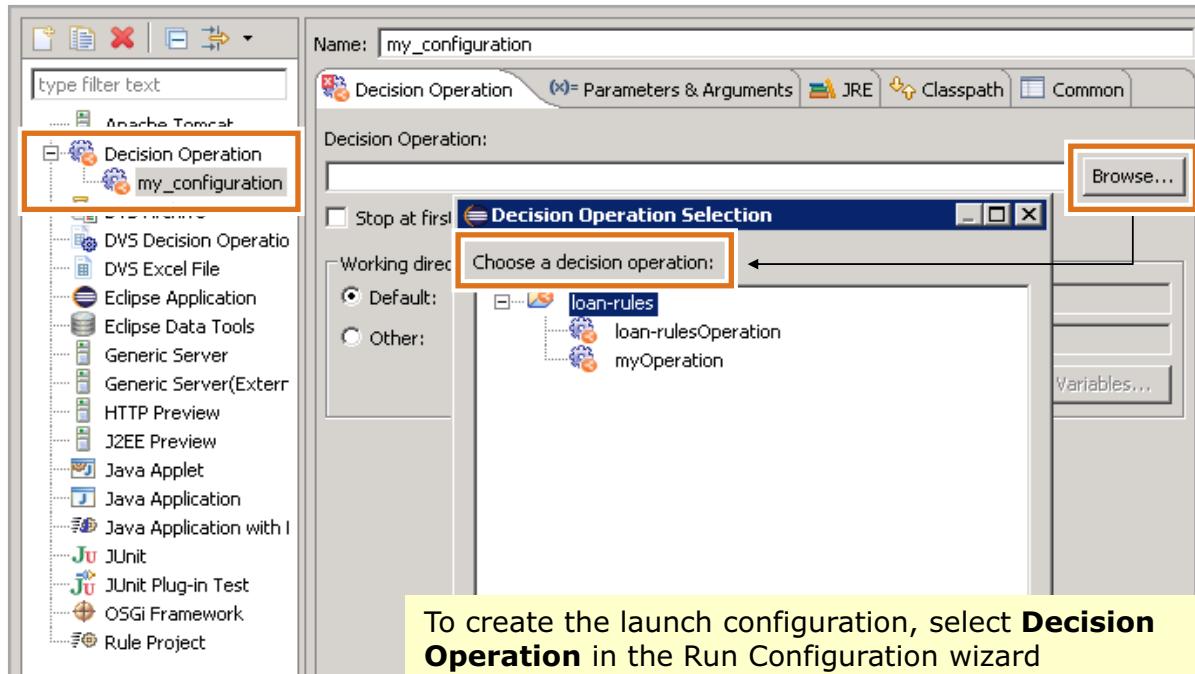
© Copyright IBM Corporation 2019

Figure 10-5. Defining launch configuration (1 of 2)

With launch configurations, you can predefine execution properties, such as parameter values and which decision operation to use when you execute the ruleset locally.



## Defining launch configuration (2 of 2)

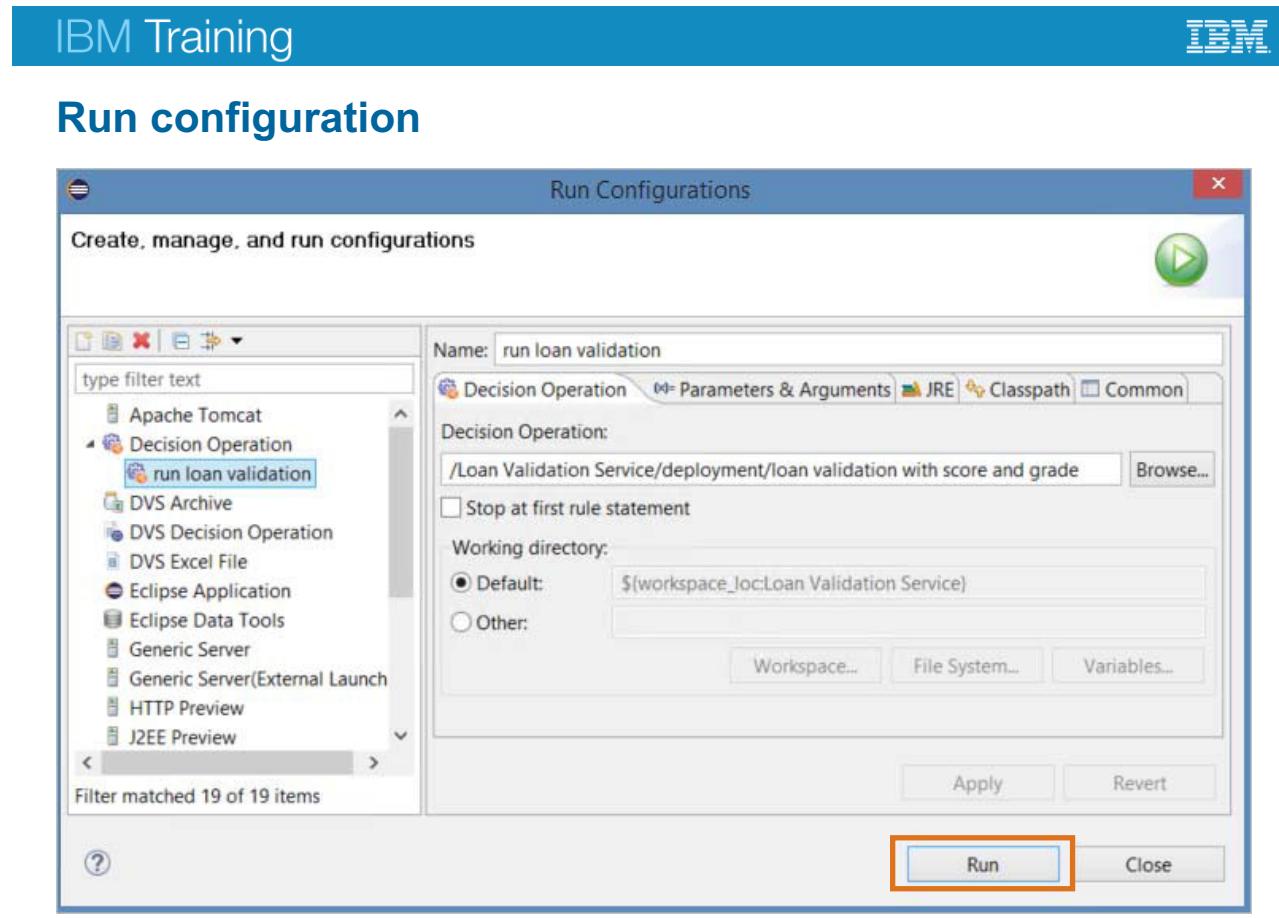


Running and debugging

© Copyright IBM Corporation 2019

Figure 10-6. Defining launch configuration (2 of 2)

For testing and debugging purposes, decision service rule projects are associated with a decision operation (.dop file). When you create the launch configuration, you select a launch configuration of the type **Decision Operation** and choose which decision operation to run.

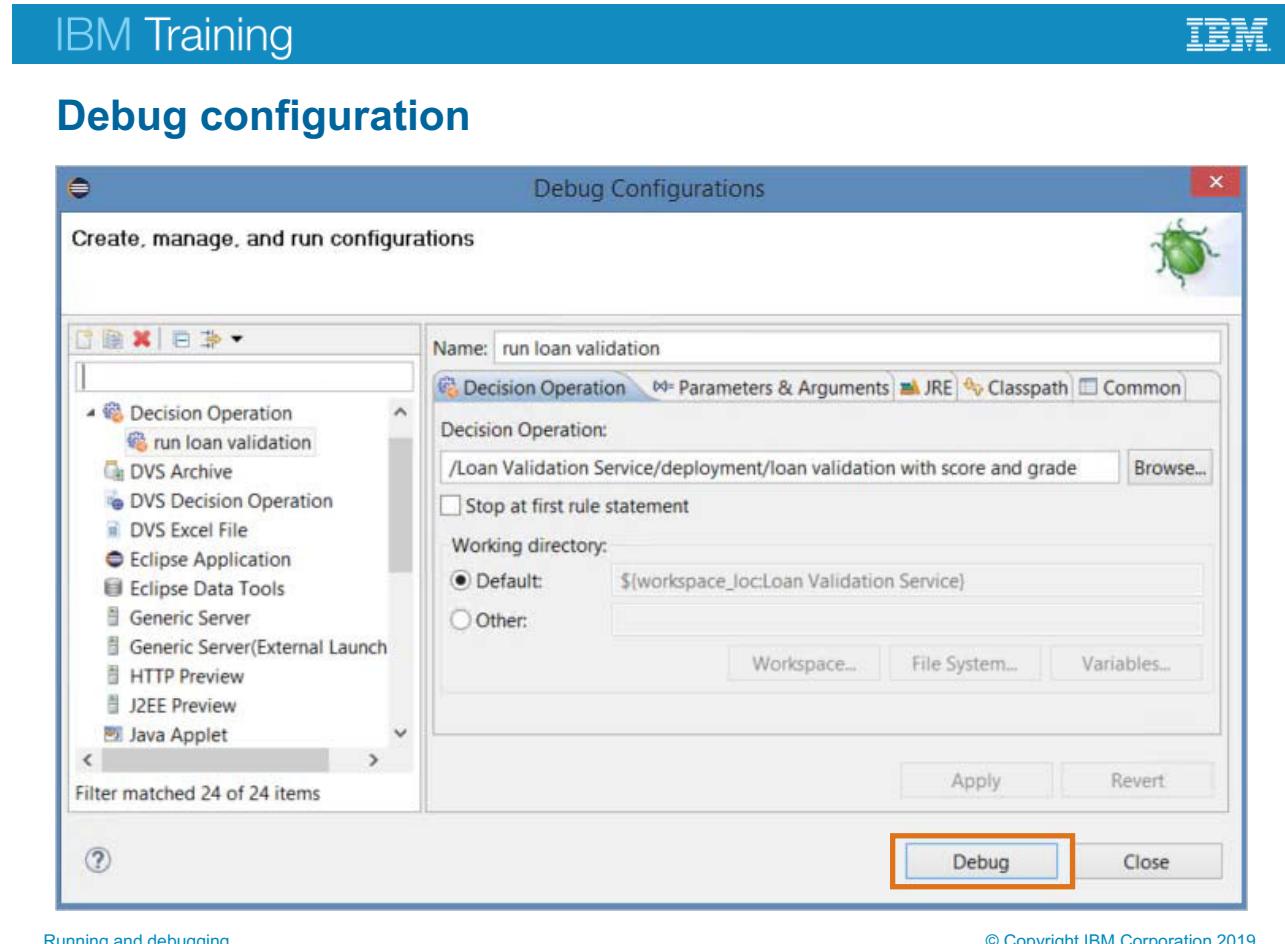


Running and debugging

© Copyright IBM Corporation 2019

Figure 10-7. Run configuration

When you choose to run with a run configuration, the ruleset executes normally. The run configuration dialog box has a **Run** button.



Running and debugging

© Copyright IBM Corporation 2019

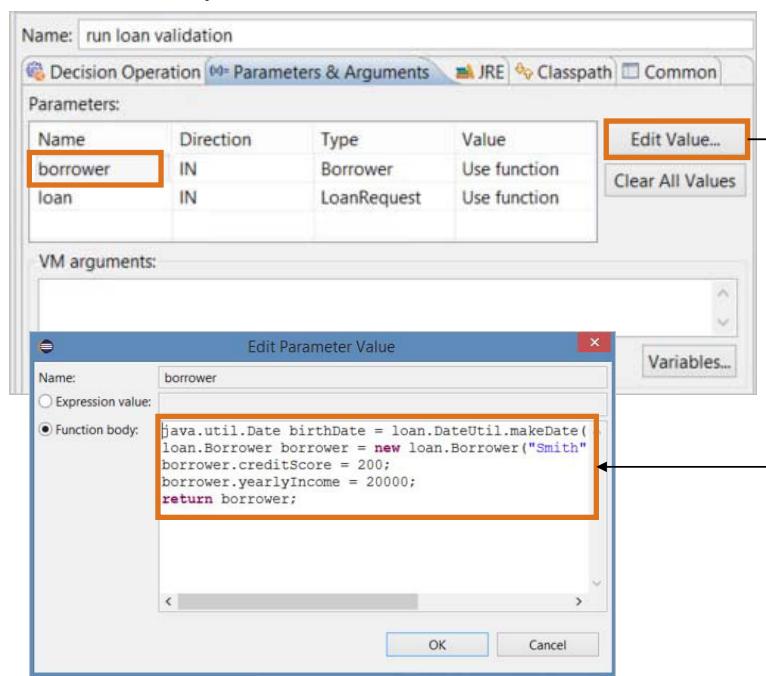
Figure 10-8. Debug configuration

When you choose to run with a debug configuration, ruleset execution switches to debug mode. The debug configuration dialog box has a **Debug** button.



## Parameters and arguments

- Manually set parameter values to test execution
  - Example: `borrower` parameter



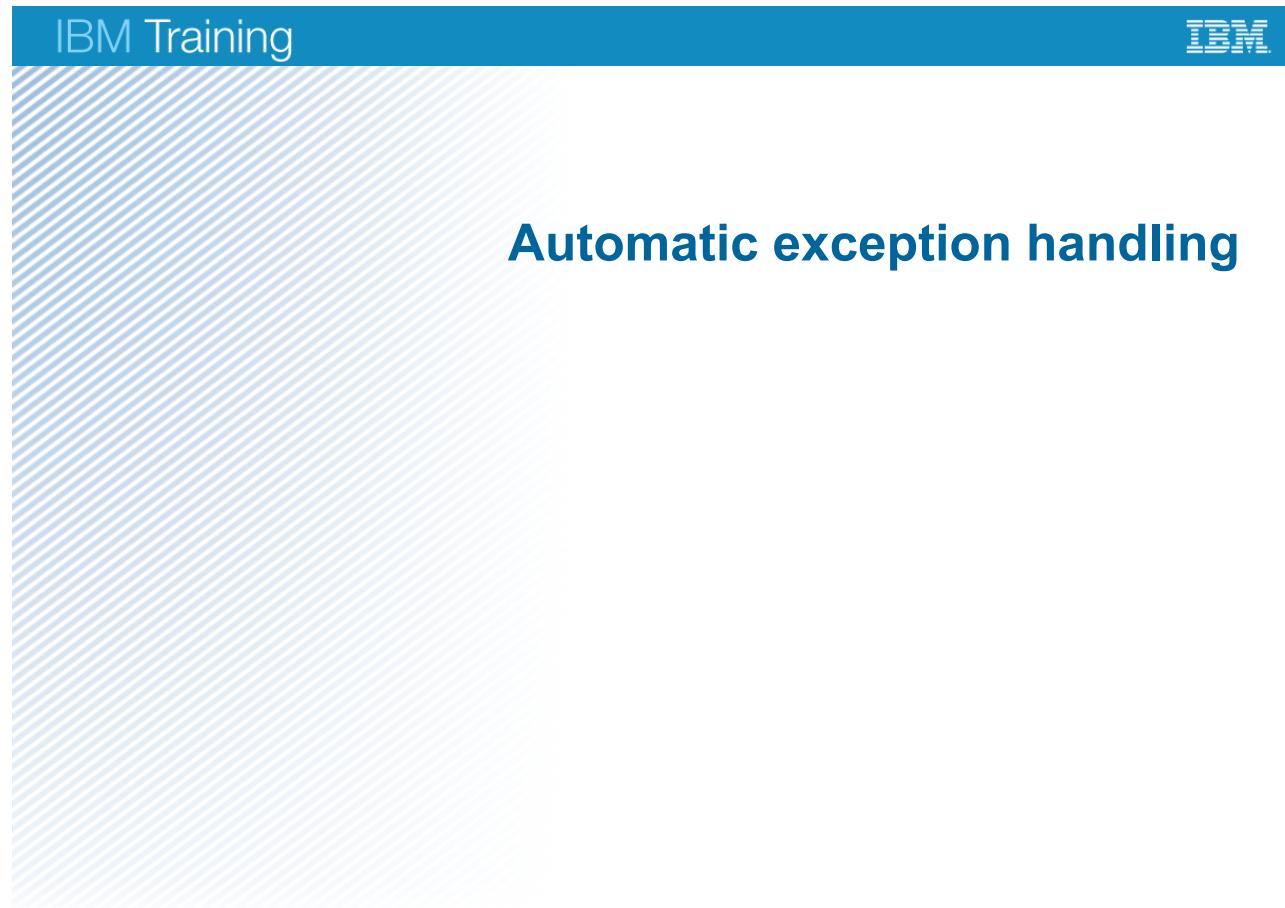
Running and debugging

© Copyright IBM Corporation 2019

*Figure 10-9. Parameters and arguments*

To execute your ruleset with a launch configuration, you set the parameters in the configuration, run it, and check the result. You do not have to create any other artifact, such as a Java project.

## 10.2. Automatic exception handling



Running and debugging

© Copyright IBM Corporation 2019

*Figure 10-10. Automatic exception handling*

## Handling exceptions

- Exceptions in rule conditions prevent rules from firing
- To handle exceptions, you can either:
  - Automatically process exceptions that prevent rules from firing
  - Customize responses to specific exceptions
- Automatic exception handling is used to deal with exceptions in rule conditions
  - Enabled at the decision service project level on the rule engine
- Custom exception handling can deal with exceptions in rule actions

Running and debugging

© Copyright IBM Corporation 2019

*Figure 10-11. Handling exceptions*

During rule execution, the rule engine evaluates the condition part of the rule against the data to determine whether to proceed to the action part of the rule. When data is missing or incomplete, a null pointer exception is thrown, and the engine sees this as an error and stops processing so that there is no output.

To handle these types of exceptions, rule authors needed to add checks in their conditions to test that an object is not null before testing other conditions on this object. However, you can enable automatic exception handling so that the engine sees the missing data as an “unknown value” instead of an error. Automatic exception handling allows the engine to ignore exceptions in conditions and continue processing rule actions.

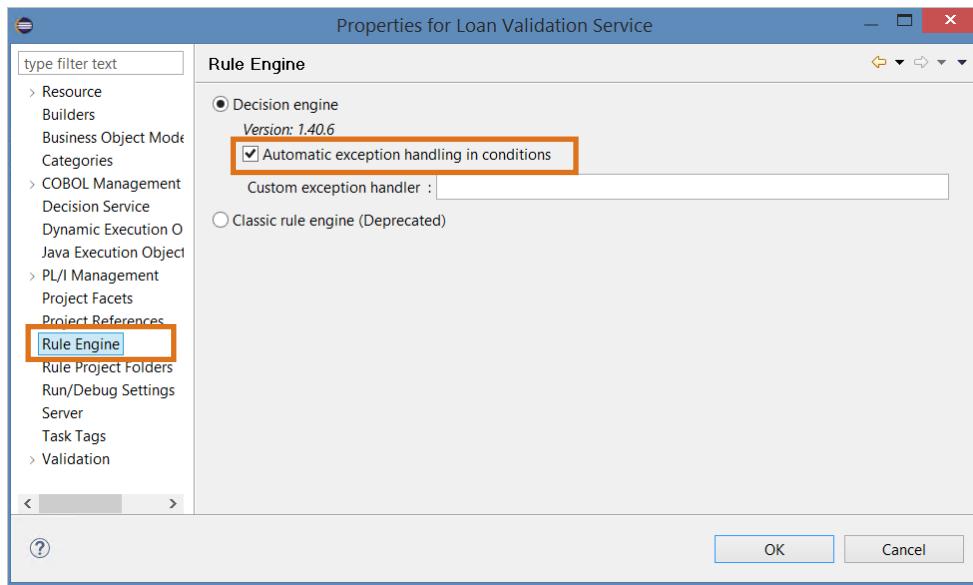
To handle exceptions in rule actions, you can use custom exception handling.



IBM

## Enabling automatic exception handling

- Enable automatic exception handling at the decision service project level
  - In the Properties dialog box, select Rule Engine
  - Select **Automatic exception handling**



Running and debugging

© Copyright IBM Corporation 2019

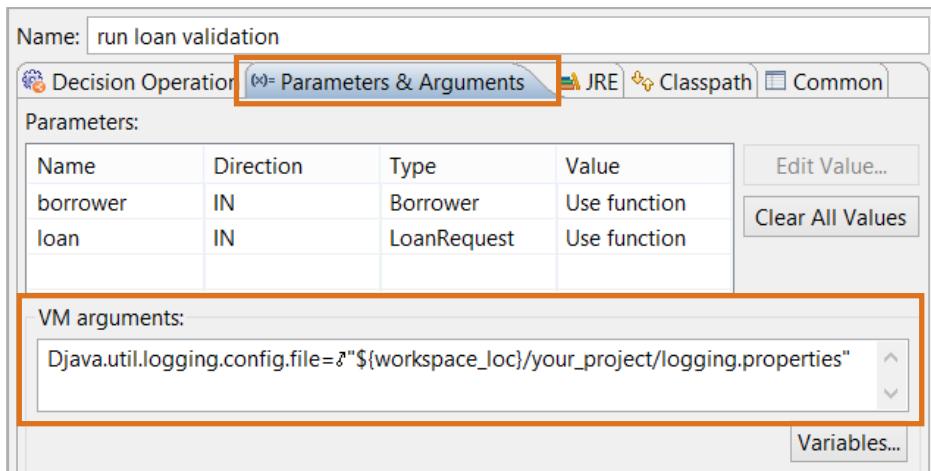
*Figure 10-12. Enabling automatic exception handling*

You enable automatic exception handling at the decision service project level. In the decision service project properties for Rule Engine, you select **Automatic Exception Handling**.



## Capturing trace logs in Rule Designer

- To determine where an exception takes place, add logging by defining a `logging.properties` configuration file
  - This logging configuration file must be available to the JVM that is used in the rule execution run or to the debug configuration
- In the debug configuration, use a Java VM parameter to point the `logging.properties` file



Running and debugging

© Copyright IBM Corporation 2019

*Figure 10-13. Capturing trace logs in Rule Designer*

To gather more information about the exceptions, you can define a logging configuration file, and make that file available to your JVM or to the debug configuration.

In the debug configuration, you set a Java VM parameter to declare the path to the logging configuration file. You can provide the path to the file in your file storage system or point to a file in your workspace, by using one of these arguments:

- `Djava.util.logging.config.file=file_path/logging.properties`
- `Djava.util.logging.config.file="${workspace_loc}/your_project/logging.properties"`

When you run the debug configuration, the Console view displays the log for automatic exception handling. The log is prefaced with: `AEH_LOG`

## 10.3. Using Rule Designer debugging tools

## Using Rule Designer debugging tools

Running and debugging

© Copyright IBM Corporation 2019

*Figure 10-14. Using Rule Designer debugging tools*

## Rule Designer debugging tools

- In Rule Designer, you can debug both the rule execution and the Java code
- With the Debugging tools, you can:
  - Inspect the execution of the rules
  - Inspect the state of the engine
  - Set breakpoints on classes, objects, rules, decision tables and trees, and ruleflows
- You can use automatic exception handling and logging to see where execution fails or where exceptions occur
  - Based on results, you can determine which ruleflow tasks or rules to set breakpoints on and track

[Running and debugging](#)

© Copyright IBM Corporation 2019

*Figure 10-15. Rule Designer debugging tools*

If your ruleset is not executing properly, you can use the Rule Designer debug tools. You can set breakpoints, step through the execution, or use expression evaluation to debug the ruleset. You can also use automatic exception handling and logging to see where execution fails or where exceptions occur.

In Rule Designer, you can debug both rule execution and Java code. Rule Designer provides a set of tools for you to:

- Inspect rule execution
- Inspect the state of the engine
- Set breakpoints on classes, objects, rules, decision tables, and ruleflows

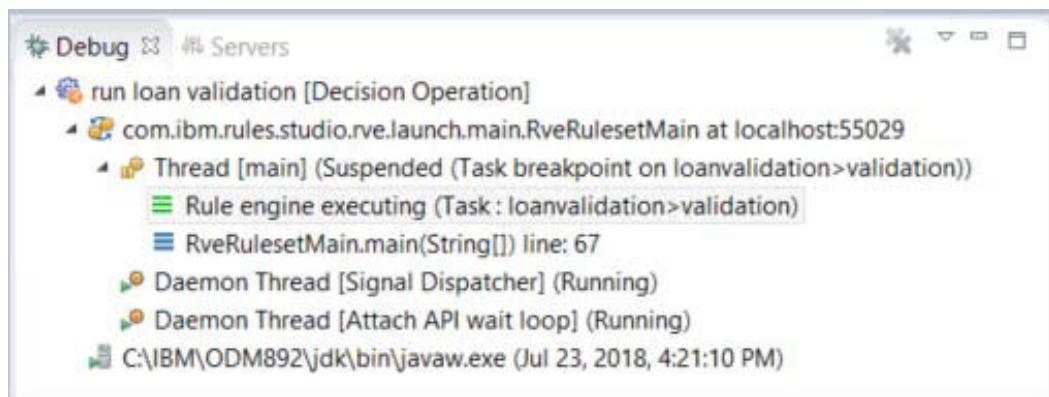
When started, the debugger does the following actions:

- Checks the ruleset
- Connects to a rule engine
- Sends the ruleset to the engine
- Resets the engine
- Fires all the rules



## Debug perspective: Debug view

- The Debug perspective includes several views to help you monitor ruleset execution
- Debug view
  - Manage the debugging of a rule project or any Java program in the workbench
  - Displays the stack frame for the suspended threads for each target you are debugging



Running and debugging

© Copyright IBM Corporation 2019

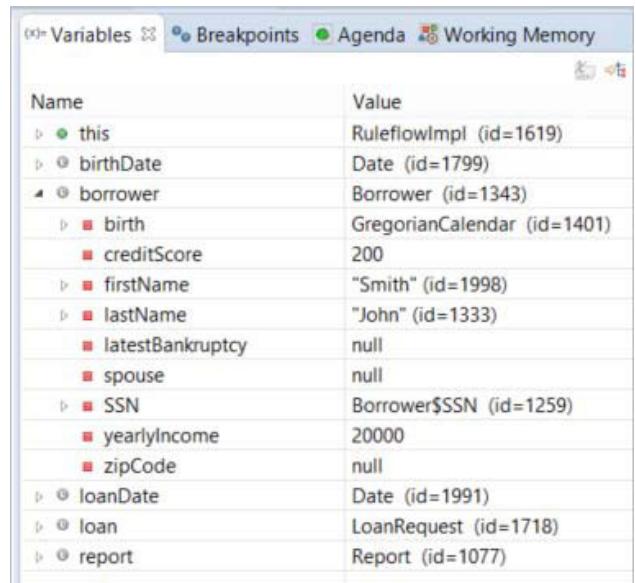
*Figure 10-16. Debug perspective: Debug view*

In the Debug perspective, the Debug view displays the stack frame for the suspended threads for each target you are debugging. Each thread is displayed as a node in the tree.

You can get the stack trace of the execution history. The stack trace is useful, particularly when an exception was thrown in the code or when a method is called from various places within your code and you want to learn from where it is called when a particular problem occurs.

## Debug perspective: Variables view

- Variables view shows the names of the variables that are bound to a business rule and parameters visible in the engine
- Values in the Variables view are updated after the evaluation of each expression
- Example: borrower
  - The borrower parameter values are listed in the Value column
  - Some values show null value, which might be the cause of a null pointer exception



| Name             | Value                       |
|------------------|-----------------------------|
| this             | RuleflowImpl (id=1619)      |
| birthDate        | Date (id=1799)              |
| borrower         | Borrower (id=1343)          |
| birth            | GregorianCalendar (id=1401) |
| creditScore      | 200                         |
| firstName        | "Smith" (id=1998)           |
| lastName         | "John" (id=1333)            |
| latestBankruptcy | null                        |
| spouse           | null                        |
| SSN              | Borrower\$SSN (id=1259)     |
| yearlyIncome     | 20000                       |
| zipCode          | null                        |
| loanDate         | Date (id=1991)              |
| loan             | LoanRequest (id=1718)       |
| report           | Report (id=1077)            |

Running and debugging

© Copyright IBM Corporation 2019

Figure 10-17. Debug perspective: Variables view

The Variables view shows the names of the variables that are bound to a business rule and parameters visible in the engine. The values in the Variables view are updated after the evaluation of each expression. As you step through execution, you can see whether variable values match expectations and are being updated correctly as rule expressions are evaluated.



## Debug perspective: Breakpoints view

- From the Breakpoints view, you can view which breakpoints are set on Java code, rules, and working memory instances
- Use breakpoints to stop the execution of rules at any point to examine the state of variables, the agenda, and working memory
- In the Breakpoints view, you can remove all breakpoints after your debugging session is done

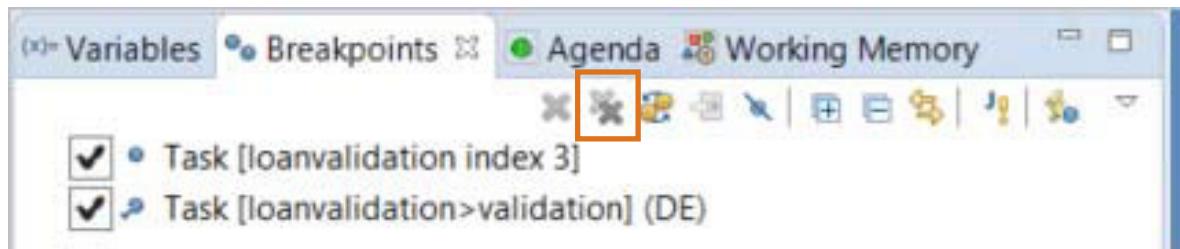


Figure 10-18. Debug perspective: Breakpoints view

## Debug perspective: Working memory and Agenda views

- When executing rules in RetePlus mode
  - The Working Memory view displays the objects in working memory
  - The Agenda view displays any rule instances that are scheduled for execution
- Working memory
  - Display the value and type of the various fields of the object
  - Dynamically inspect the objects as they affect the rules
- Agenda
  - Shows the current state of the agenda
  - Displays any business rule instances that are scheduled for execution
- Use the Agenda and Working memory views together to see how rules and objects affect each other

[Running and debugging](#)

© Copyright IBM Corporation 2019

*Figure 10-19. Debug perspective: Working memory and Agenda views*

- **Working Memory**

The Working Memory view displays the list of all the objects in working memory. This view applies to the RetePlus execution mode activities. This view displays the value and type of the attributes in an object, and allows direct inspection of the objects that it uses, which is important in determining whether a rule is behaving correctly. For example, rules that are currently in the agenda and that use a specific value can easily be checked to see whether they have a problem with their behavior. You can dynamically inspect objects as they affect rules.

- **Agenda**

The Agenda view shows the current state of the agenda. It displays any business rule instances that are scheduled for execution. This view applies to the rule tasks. The view can display the priority of a rule in the agenda and the objects that it uses.

## Unit summary

- Use launch configurations to execute and debug rulesets
- Work with automatic exception handling
- Work with Rule Designer debugging tools

Running and debugging

© Copyright IBM Corporation 2019

*Figure 10-20. Unit summary*

## Review questions

1. Which term denotes the type of session under which you execute a rule project in Rule Designer?
  - A. Decision Warehouse
  - B. Launch configuration
  - C. BOM-to-XOM mapping
  - D. All of the above
2. True or False: With automatic exception handling, the rule engine can finish processing the rules and return the output.
3. True or False: Debugging is a development task that is done iteratively in Rule Designer, while working with your rule artifacts.



Running and debugging

© Copyright IBM Corporation 2019

Figure 10-21. Review questions

Write your answers here:

- 1.
- 2.
- 3.

## Review answers (1 of 2)



1. Which term denotes the type of session under which you execute a rule project in Rule Designer?
  - A. Decision Warehouse
  - B. Launch configuration
  - C. BOM-to-XOM mapping
  - D. All of the above

The answer is B.
2. True or False: With automatic exception handling, the rule engine can finish processing the rules and return the output.  
The answer is True.
3. True or False: Debugging is a development task that is done iteratively in Rule Designer, while working with your rule artifacts.  
The answer is True. Debugging is a step that you undertake in Rule Designer, iteratively, while you develop your business rule artifacts.

Running and debugging

© Copyright IBM Corporation 2019

Figure 10-22. Review answers (1 of 2)

## Exercise: Executing rules locally

Running and debugging

© Copyright IBM Corporation 2019

*Figure 10-23. Review answers (2 of 2)*

## Exercise introduction

- Create launch configurations to run rulesets locally



Running and debugging

© Copyright IBM Corporation 2019

*Figure 10-24. Exercise: Executing rules locally*

## Exercise: Debugging a ruleset

Running and debugging

© Copyright IBM Corporation 2019

*Figure 10-25. Exercise introduction*

## Exercise introduction

- Use automatic exception handling
- Set breakpoints in rules, decision tables, and ruleflows
- Run a debugging session



Running and debugging

© Copyright IBM Corporation 2019

*Figure 10-26. Exercise: Debugging a ruleset*

# Unit 11. Enabling tests and simulations

## Estimated time

01:15

## Overview

This unit teaches you how to enable business users to run tests and simulations.

## How you will check your progress

- Review
- Exercise

## Unit objectives

- Describe the basic features of testing and simulation
- Collaborate with business users to set up testing and simulation

*Figure 11-1. Unit objectives*

## Topics

- Overview of testing and simulation
- Setting up testing and simulation
- Working with scenarios

Enabling tests and simulations

© Copyright IBM Corporation 2019

*Figure 11-2. Topics*

## 11.1. Overview of testing and simulation

## Overview of testing and simulation

Enabling tests and simulations

© Copyright IBM Corporation 2019

*Figure 11-3. Overview of testing and simulation*

## What is testing and simulation?

- Testing
  - Validate the accuracy of a ruleset by comparing expected results with the actual results obtained execution
- Simulations
  - Evaluate the potential impact of changes to rules (“what-if” analysis)
  - Results are shown in key performance indicators (KPIs)

Figure 11-4. What is testing and simulation?

Use the testing and simulation features to validate rules in development and business user environments.

*Testing* verifies that a ruleset is correctly designed and written. You test by comparing the expected results, that is, how you expect the rules to behave, and the actual results that are obtained when applying rules with the scenarios that you defined.

*Simulations* enable you to see the effects of changes to rules and data through key performance indicators (KPIs). You use simulations to do what-if analysis against realistic data to evaluate the potential impact of changes you might want to make to your rules. This data corresponds to your usage scenarios, and often comes from historical databases that contain real customer information.

## What are scenarios?

- Scenarios
  - Provide values for input parameters for ruleset execution and expected result values
  - Use real or fictitious use cases
  - Generated in Microsoft Excel format
- Excel format supports
  - Projects with relatively simple and small object models
  - Testing that uses thousands of scenarios
- Scenario files must be generated in Rule Designer or Decision Center from valid decision operations

Enabling tests and simulations

© Copyright IBM Corporation 2019

Figure 11-5. What are scenarios?

*Scenarios* represent fictional or actual business data that you use as input for both testing and simulation.

The scenarios contain all the necessary information that is required to validate behavior of rules.

The data can be specific use case scenarios, or extracted from historical databases that contain real customer information.

You can use Microsoft Excel spreadsheets to store your scenarios, where:

- Rows represent a scenario
- Columns indicate what data is included for each scenario

Microsoft Excel scenarios can use flat or tab-based formats.

## Example scenario: Loan application

- Input from BOM:
  - Borrower
  - Loan
- Column headings use the verbalized names for BOM classes and attributes
- Rows correspond to two scenarios:
  - What happens when John Smith asks for a large loan (500,000)
  - What happens when John Smith asks for a small loan (25,000)

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

Figure 11-6. Example scenario: Loan application

For example, the following Microsoft Excel sheet contains these scenarios:

- A scenario that is named Big Loan to see how your rules work with an important loan request
- A scenario that is named Small Loan to see how your rules work with a smaller loan request

The Microsoft Excel format is appropriate to test projects with relatively simple and small object models or testing that uses thousands of scenarios. Business users can generate scenario file templates in Decision Center Business console but might require your help for more complex formats for their scenarios. You can support business users by providing scenario files that are prepopulated with values.

## Decision Runner

- Test and simulation engines
- Input for tests and simulations:
  - Scenarios and rules are submitted to the runtime service
  - For tests, you also submit expected results
  - Decision runner sends the rules and scenarios to Rule Execution Server, which executes the rules on the scenarios

Enabling tests and simulations

© Copyright IBM Corporation 2019

*Figure 11-7. Decision Runner*

To run tests and simulations, you use a test and simulation engine called Decision Runner.

This runtime service is a web application that is hosted on an application server with Rule Execution Server installed.

During both testing and simulation, you submit the scenarios and rules to the runtime service. For tests, you also submit expected results. In turn, the runtime service sends the rules and scenarios to Rule Execution Server, which executes the rules against the scenarios.

## Reports

- The information that is generated during execution is returned in a detailed report
- Reports contain
  - Summary section with percentage of scenarios that are executed successfully
  - Results section with the results for each test on each scenario
- Test results:
  - Expected results are compared to actual results obtained during execution
  - Returns a report that details whether the scenario passed or failed
- Simulation results:
  - Returns a report with an interpretation of the execution results based on KPIs to compute business metrics

*Figure 11-8. Reports*

Running a test suite or simulation generates a large amount of information. A report summarizes the generated information, detailing whether each scenario passed or failed. Test results can be one of:

- Successful: Expected results match actual results
- Failure: Expected results do not match actual results
- Error: Scenarios failed to execute

For simulations, the report includes an interpretation of the results that are based on key performance indicators (KPI). To facilitate interpretation of simulation results for business users, you can customize the KPIs and the appearance of reports.

You can use ODM APIs to run Decision Runner so that you can integrate test execution into your own build systems.

## Testing and simulation in the decision lifecycle

- During development and rule validation phases:
  - Test newly authored rules
  - Verify rules from previous releases
- During rule analysis and authoring phases:
  - Simulate the effects of rule changes
- Business users can do various types of tests
  - Ruleset testing: Tests the local unit of logic, which in this case is the set of rules to ensure that they work
  - Boundary testing: Determines the behavior when boundary values are used and when values outside the boundaries are used
  - Regression testing: Ensures that previous rules continue to work properly
  - Impact testing: Determines the effects of changing rules

Figure 11-9. Testing and simulation in the decision lifecycle

Understanding when to test and the variety of tests that business users run can help you to support them when you generate the scenario files and populate them with data.

## 11.2. Setting up testing and simulation

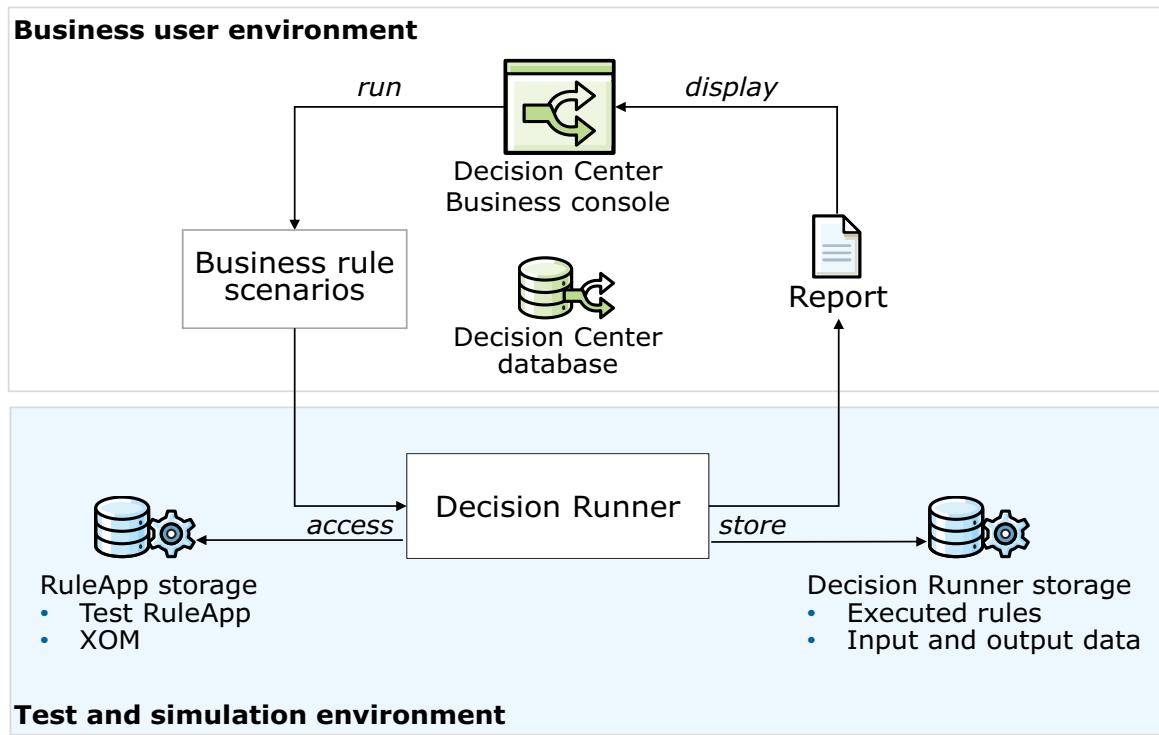
## Setting up testing and simulation

Enabling tests and simulations

© Copyright IBM Corporation 2019

*Figure 11-10. Setting up testing and simulation*

## Enabling remote testing in Business console



Enabling tests and simulations

© Copyright IBM Corporation 2019

Figure 11-11. Enabling remote testing in Business console

In this graphic, you can see that business users run tests and simulations in the Business console. They can generate scenario files and run them with Decision Runner. The Java XOM must be accessible to Decision Runner for scenario execution. After execution, the Decision Runner returns a report with detailed results of the test or simulation.

Test execution results are stored in Decision Runner storage.

After you set up the test server, and make it accessible from Decision Center, business users can create scenario file templates and run tests and simulations directly from the Business console. Scenarios are stored in the Decision Center database, and they are queried and version-controlled like other rule artifacts.

## Supporting business users for testing and simulation

- Varying levels of collaboration with business users might be required for some tasks
- Validate the project
  - Define the rule project with the correct ruleset parameters, which determine the scenario values to test
  - Validate the BOM and generate complex scenario file templates
- Make the Java XOM accessible
- Publish the decision service to Decision Center
- Provide access to an instance of the Decision Runner
- Create custom data providers

*Figure 11-12. Supporting business users for testing and simulation*

As developers, you support business users by preparing the test and simulation environment so that business users can run tests and simulations directly from Decision Center with minimal dependence on you. Varying levels of collaboration with business users might be required for some tasks. Business users can generate and populate scenario file templates in Business console, but they might need your help for more complex scenario files. In Business console, business users can also define KPIs and report formats.

Setting up the test environment for business users involves these tasks:

- Validate the project
  - To enable business users to test a ruleset, you first validate the BOM and ensure that scenario file templates can be generated properly. You collaborate with rule authors to identify what data to test, and define the tests to include in the Expected Results sheet.
  - A valid project means that you can create the correct columns when generating the scenario file template.
  - You can validate the project in Rule Designer.
- Make the Java XOM accessible
  - If the decision service uses a Java XOM, that XOM must be available to the Decision Runner so that tests and simulations can be run.

- You can deploy the Java XOM from Rule Designer.
- Publish the rule projects from Rule Designer
  - If the BOM is modified when validating the project, you must make sure that those changes are published to Decision Center so that business users can generate the correct scenario template files.
- Provide access to an instance of the Decision Runner
- Creating custom data providers
  - After you generate a scenario file template, it can be populated with test data that is taken from a database or another source.
  - You can obtain a custom data provider for tests or simulations in the Business console by creating an XML descriptor file.

## 11.3. Working with scenarios

## Working with scenarios

Enabling tests and simulations

© Copyright IBM Corporation 2019

Figure 11-13. Working with scenarios

## Scenario files and the BOM

- Scenario files mirror your BOM
  - When you define the scenario file template, you use the BOM class and attribute names
  - When the template file is generated, column headings use the verbalization
- If you are working in Rule Designer to prepare template files:
  - Be familiar with both the BOM and its verbalization
  - Understand the relationship between the scenario files and classes in the BOM
- You must understand your object model and the main objects you are sending as scenario data

Figure 11-14. Scenario files and the BOM

Scenario files mirror your BOM. To recognize the relationship between the Excel files and the BOM, you must understand your object model and be familiar with the BOM members **and** their verbalization.

As you see during the exercises, when you create the scenario file template, you select which BOM members to include. The template is generated with the verbalized names of the BOM members. The structure of the scenario file reflects which objects you send as input.

Before generating the scenario file templates, you must validate that the BOM is properly configured so that columns in the **Scenarios** and **Expected Results** sheets generate correctly.

To help business users recognize the relationship between the Microsoft Excel files and the BOM, they must understand the object model, and be familiar with the BOM members and their verbalization. You must also collaborate with business users to identify the main objects that are used as scenario data.

## Relationships in the BOM (1 of 2)

- Relationships in your object model are translated differently in Decision Center than in tests
- In Decision Center, the “of the” constructions in the rule statements indicate relationships between objects

the property type of the property **of the** borrower

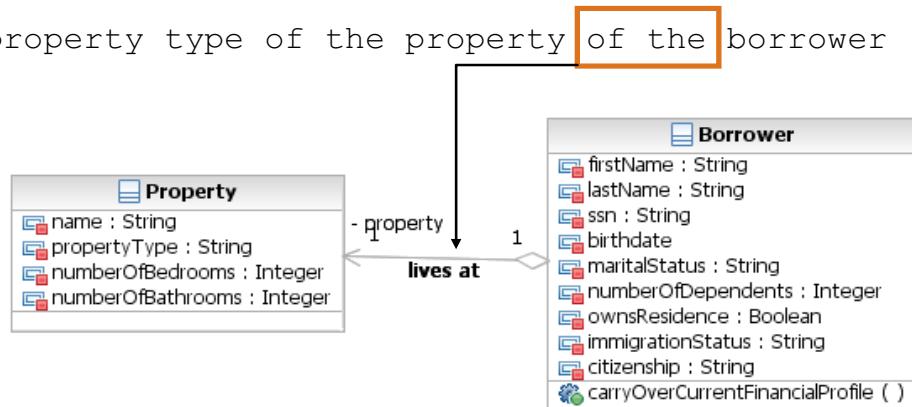


Figure 11-15. Relationships in the BOM (1 of 2)

When testing objects with relationships, you should be aware of how these relationships are translated in the scenario file.

For example, the relationship between the property and the borrower is shown here in the object model. In Decision Center, this relationship is translated with the “of the” construct in the rule syntax, as shown here:

the property type of the property of the borrower



## Relationships in the BOM (2 of 2)

- Relationships in the BOM are translated as multiple tabs in scenario files

- The **Scenarios** tab defines the Borrower (parent class)
- The **PropertyInfo** tab defines the Property (child class)

testsuite.xls [Compatibility Mode] - Microsoft Excel

| Scenario ID | description | first name | last name | birth date | SSN | credit score | property info | yearly income | zip code | start date | number of monthly payments | amount |
|-------------|-------------|------------|-----------|------------|-----|--------------|---------------|---------------|----------|------------|----------------------------|--------|
| Scenario 1  |             |            |           |            |     |              |               |               |          |            |                            |        |

Scenarios    PropertyInfo    Expected Results    HELP

Enabling tests and simulations

© Copyright IBM Corporation 2019

Figure 11-16. Relationships in the BOM (2 of 2)

In the scenario files, relationships are translated as multiple tabs in the scenario file. The top sheet shows the main input classes. Related classes are on tabs.

The **PropertyInfo** class is related to the **Borrower** class.



## Structure of the scenario files (1 of 3)

- The top-level sheet is always called **Scenarios**
- The main objects that are provided as input to rule execution are shown on the Scenarios sheet
  - Example shows the **Borrower** and **Loan** classes and their attributes as input to rule execution

| Scenario ID | description | first name   | last name | birth date | SSN | credit score | property info | yearly income | zip code | start date | number of monthly payments | amount |
|-------------|-------------|--------------|-----------|------------|-----|--------------|---------------|---------------|----------|------------|----------------------------|--------|
| Scenario 1  |             | the borrower |           |            |     |              |               |               |          | the loan   |                            |        |

Enabling tests and simulations

© Copyright IBM Corporation 2019

Figure 11-17. Structure of the scenario files (1 of 3)

Scenario file templates can include these sheets:

- Scenarios** sheet: Used to enter the input data, that is, the scenarios for your test suites or simulations. All scenario files have this sheet.
- Data entry** sheet: Used to regroup information that is used in other sheets.

The name of this data entry sheet contains the type of data that is expected in the column of the other sheet where its entries are used.

- Expected Results** sheet: Used to enter the expected results when running tests.
- Expected Execution Details** sheet: Used to enter the expected execution details when running tests.

The main objects that are provided as input to rule execution are shown on the **Scenarios** sheet, which is the top-level sheet in the scenario files. As shown here, the borrower and the loan are the input parameters for this ruleset, and columns are generated corresponding to the class constructor arguments.



## Structure of the scenario files in Microsoft Excel (2 of 3)

- Column headings use the verbalization of the attribute
  - The column heading **property info** is the verbalization of the `PropertyInfo` attribute
- The cell that refers to the specific class uses the class identification string
  - For Scenario 1, the **property info** values **PropertyInfo1**

|    |             | the borrower |            |           |            |           |              |               |              |          |            | the loan         |  |  |
|----|-------------|--------------|------------|-----------|------------|-----------|--------------|---------------|--------------|----------|------------|------------------|--|--|
|    | Scenario ID | description  | first name | last name | birth date | SSN       | credit score | property info | early income | zip code | start date | number of months |  |  |
| +  | Scenario 1  |              | Joe        | Smith     | 4/12/1983  | 123456789 | 60           | PropertyInfo1 | 52,500       | 12345    | 4/1/2010   |                  |  |  |
| 10 |             |              |            |           |            |           |              |               |              |          |            |                  |  |  |
| 11 |             |              |            |           |            |           |              |               |              |          |            |                  |  |  |
| 12 |             |              |            |           |            |           |              |               |              |          |            |                  |  |  |
| 13 |             |              |            |           |            |           |              |               |              |          |            |                  |  |  |
| 14 |             |              |            |           |            |           |              |               |              |          |            |                  |  |  |

Enabling tests and simulations © Copyright IBM Corporation 2019

Figure 11-18. Structure of the scenario files in Microsoft Excel (2 of 3)

The first column in the **Scenarios** sheet is the **Scenario ID**, which uniquely identifies each scenario.

The **Scenarios** sheet includes:

- Required columns: These columns provide the data that is required to execute your rulesets. The name of a required column is shown in bold in the Microsoft Excel sheet.  
Required columns are defined through a constructor of the BOM class. You must define a constructor. When you have more than one constructor, choose the constructor with arguments that you want included in the template. You might need to modify the names of the constructor arguments to make them meaningful to business users.
- Optional columns: Optional columns correspond to attributes of the BOM class. If the attribute is verbalized, the column heading uses the verbalization of the attribute.

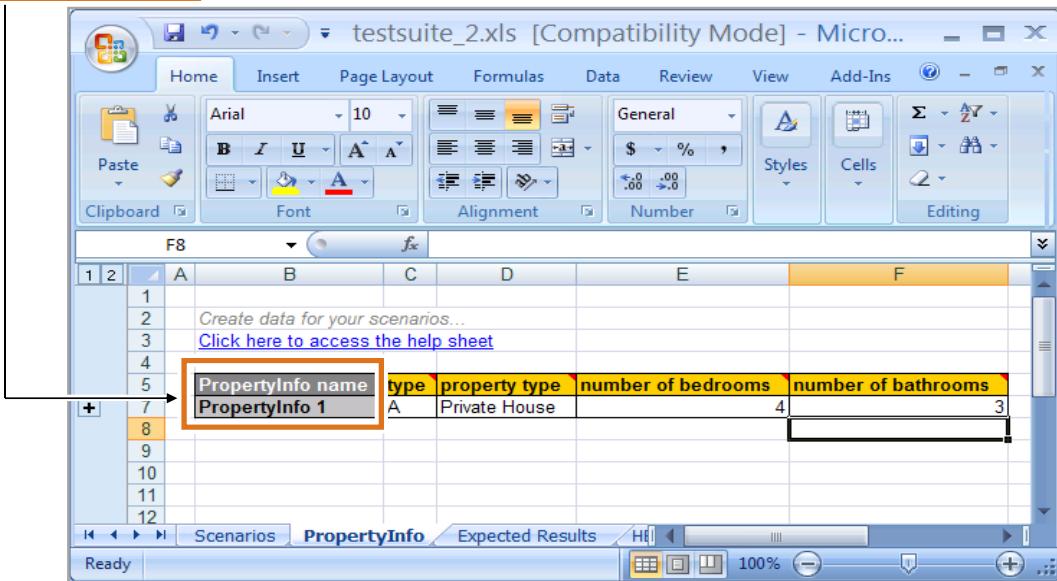


### Information

Only attributes that are defined as “**Read/Write**” or “**Write Only**” can be used to create optional columns in the **Scenarios** sheet.

## Structure of the scenario files in Microsoft Excel (3 of 3)

- The spreadsheet tab for the child class uses the class ID to show which scenario the values are related to
  - PropertyInfo1** identifies the attribute values for **property info** in **Scenario 1**



The screenshot shows a Microsoft Excel window titled "testsuite\_2.xls [Compatibility Mode] - Micro...". The ribbon tabs at the top are Home, Insert, Page Layout, Formulas, Data, Review, View, Add-Ins, and Help. The "Home" tab is selected. The main area shows a table with the following data:

| PropertyInfo name | type | property type | number of bedrooms | number of bathrooms |
|-------------------|------|---------------|--------------------|---------------------|
| PropertyInfo 1    | A    | Private House | 4                  | 3                   |

Enabling tests and simulations

© Copyright IBM Corporation 2019

Figure 11-19. Structure of the scenario files in Microsoft Excel (3 of 3)

When you open the  **PropertyInfo**  tab, the  **PropertyInfo1**  child class is listed, along with its attributes that are included for testing.



## Expected results (1 of 2)

- Define expected results according to how you expect the rules to behave when executed with your scenarios
  - Compare expected results to the actual results obtained from execution to see whether they match
- In the **Expected Results** sheet, the ruleset output parameters and the BOM classes define the columns
- Example: The three columns represent the expected results for three tests

| 5  | Scenario ID | the yearly repayment equals | the loan report is approved equals | the message equals           |
|----|-------------|-----------------------------|------------------------------------|------------------------------|
| 9  | Big Loan    | 39597                       | FALSE                              | Too big Debt-To-Income ratio |
| 10 | Small Loan  | 1979                        | TRUE                               | Loan approved                |
| 11 |             |                             |                                    |                              |

Scenarios    Expected Results    HELP   

Figure 11-20. *Expected results (1 of 2)*

To evaluate the results after running tests, you define **expected results** for each scenario according to how you expect the rules to behave.

After test execution, a report is returned. The report compares the expected values to the actual results to see whether they match.

When the expected values do not match the actual results, you can investigate whether the rule is incorrect or whether the scenario values must change.

**Expected Results** are on the last tabbed sheet of the scenario file template.

The columns in the **Expected Results** sheet are generated according to the BOM classes that you use as ruleset output parameters. You select which attributes you want returned for testing the decision results.

To skip a test for a particular scenario, leave the column empty.

## Expected results (2 of 2)

- Developers can provide a populated scenario file
  - Requires more programming time from the developers
  - Collaborate with developers to identify default values
- If the results after an initial run are correct, you can use them to replace the prepopulated results
- To populate the expected results, use:
  - Default values
  - True expected values
  - Actual results from previous rule execution

Figure 11-21. Expected results (2 of 2)

Business users might need you to provide a prepopulated scenario file for them. If you are replacing an existing system, you might be able to populate the expected results with data from your old system. However, if the behavior of the old system changes with the new implementation, the output data might not match the previous results. Be prepared to analyze the implementation logic to make sure that the implementation produces the expected behavior.

Business users can manually edit a prepopulated scenario file. Some experimentation with the values might be required until both the rules and the results are correct. If the business users populate the expected results themselves, they can use:

- Default values
- True expected values
- Actual results that are obtained during the previous rule execution

## Adding and removing columns in the Microsoft Excel file

- By default, the template includes a column for each constructor argument and for each non-static and writable attribute
- Configure the BOM when you must add or remove columns in the **Scenarios** and **Expected Results** sheets to generate the scenario file template

Figure 11-22. Adding and removing columns in the Microsoft Excel file

You can configure the BOM to add or remove columns in the **Scenarios** and **Expected Results** sheets when you generate the scenario file templates.

By default, the template includes a column for each constructor argument and for each attribute (providing it is non-static and writable).

To add or remove columns that are generated for the constructor arguments, you must either modify the list of arguments of the constructor, or choose a different constructor.

## Adding columns with virtual attributes

- You can add new columns to the **Scenarios** and **Expected Results** sheets by creating virtual attributes in the BOM
- To include a virtual attribute in:
  - The **Scenarios** sheet: Create writable attributes
  - The **Expected Results** sheet: Create readable attributes
- You must define the BOM-to-XOM mapping for all virtual attributes

Enabling tests and simulations

© Copyright IBM Corporation 2019

Figure 11-23. Adding columns with virtual attributes

To add new columns to the **Scenarios** and **Expected Results** sheets, create *virtual* attributes in the BOM.

With virtual attributes, you can test attributes of a complex type, such as collections of complex objects, or maps.

You must use:

- Writable attributes (“**Read/Write**” or “**Write Only**”) for optional columns in the **Scenarios** sheet
- Readable attributes (“**Read/Write**” or “**Read Only**”) for columns in the **Expected Results** sheet

## Removing columns in the scenario file

- Columns in the scenario file are generated for each attribute of the BOM class that you are testing
- If you do not want an attribute to be included as a column in the **Scenarios** sheet, specify in the BOM that this member is ignored for testing purposes

Figure 11-24. Removing columns in the scenario file

If you do not want to include all the attributes as columns in the **Scenarios** sheet, you must specify in the BOM which members to ignore for testing purposes.

For example, do not include as input any attributes that are based on calculated values. When these attributes are included as input, the columns remain empty because they do not have input values. These empty columns can be a source of confusion for business users. Instead, you might use these attributes in the **Expected Results** sheet to test that their values are calculated correctly as a result of rule execution.

## Unit summary

- Describe the basic features of testing and simulation
- Collaborate with business users to set up testing and simulation

Enabling tests and simulations

© Copyright IBM Corporation 2019

*Figure 11-25. Unit summary*

## Review questions

1. True or False: Testing allows business users to validate the behavior of a ruleset by comparing the expected results with KPIs.
2. True or False: Scenarios represent fictitious or actual business data that you use as input for both testing and simulation.
3. True or False: Business users can run tests and simulations in Business console with minimal dependence on the development team.
4. True or False: You must always generate and define scenario template files in Rule Designer, and then publish them to Decision Center for business users.



Enabling tests and simulations

© Copyright IBM Corporation 2019

Figure 11-26. Review questions

Write your answers here:

- 1.
- 2.
- 3.
- 4.

## Review answers (1 of 2)

1. True or False: Testing allows business users to validate the behavior of a ruleset by comparing the expected results with KPIs.

The answer is False. With testing, you validate the behavior of a ruleset by comparing the results that you expect with the results that are obtained during execution.



2. True or False: Scenarios represent fictitious or actual business data that you use as input for both testing and simulation.

The answer is True.

3. True or False: Business users can run tests and simulations in Business console with minimal dependence on the development team.

The answer is True.

Figure 11-27. Review answers (1 of 2)

## Review answers (2 of 2)

4. True or False: You must always generate and define scenario template files in Rule Designer, and then publish them to Decision Center for business users.  
The answer is False. Business users can generate and define scenario files directly in Business console.



Figure 11-28. Review answers (2 of 2)

## Exercise: Enabling rule validation

Enabling tests and simulations

© Copyright IBM Corporation 2019

*Figure 11-29. Exercise: Enabling rule validation*

## Exercise introduction

- Validate the BOM and generate scenario file templates
- Customize scenario file templates
- Validate remote testing conditions for business users in the Business console



Enabling tests and simulations

© Copyright IBM Corporation 2019

*Figure 11-30. Exercise introduction*

---

# Unit 12. Managing deployment

## Estimated time

01:00

## Overview

This unit teaches you how to deploy and manage rule artifacts for execution in Rule Execution Server. It also covers how to use Ant tasks and the Build Command Maven plug-in for RuleApp management.

## How you will check your progress

- Review
- Exercises

## Unit objectives

- Describe the principles for managing RuleApp and XOM deployment
- Prepare deployment configurations
- Build and deploy RuleApps outside of Rule Designer

Figure 12-1. Unit objectives

## Topics

- Introducing managed execution
- Deploying decision services
- Managing XOMs
- Managing resources with Ant tasks
- Managing resources through the REST API
- Building RuleApps with the Build Command Maven plug-in

Figure 12-2. Topics

## 12.1. Introducing managed execution

## Introducing managed execution

Managing deployment

© Copyright IBM Corporation 2019

*Figure 12-3. Introducing managed execution*

## Introduction to Rule Execution Server

- Managed environment for rule execution
- Set of independent and cooperating software modules that interact with the rule engine
  - Provide management, performance, security, and logging capabilities
- Flexible modular architecture that can service different server clients and integration with enterprise infrastructure
- Can handle rule execution for multiple applications

Managing deployment

© Copyright IBM Corporation 2019

*Figure 12-4. Introduction to Rule Execution Server*

Rule Execution Server is designed to simplify rule execution for you by providing a management and monitored environment. You can use it for simultaneous execution of multiple rulesets, ruleset updates, and transaction management. It also handles the creation, pooling, and management of ruleset instances. The flexible architecture can support various enterprise integrations, including the Java Standard and Enterprise Editions.

## Key functions of Rule Execution Server

- Execution scalability by using resource pooling
- Management through a web-based interface and JMX tools
- Full integration in Java EE and Java SE
- Automation with Ant tasks
- Logging, monitoring, and debugging integration
- Integration into the development process

Managing deployment

© Copyright IBM Corporation 2019

*Figure 12-5. Key functions of Rule Execution Server*

More specifically, Rule Execution Server incorporates the following key features that provide enterprise-level ruleset execution and management facilities:

- Execution scalability by using resource pooling
- Management through a web-based interface and JMX tools
- Full integration in Java EE and Java SE
- Automation with Ant tasks
- Logging, monitoring, and debugging integration
- Integration into the development process

Rule Execution Server supports clustering and addresses many of the demands of 24x7 production applications. It can handle business rule execution for multiple business rule applications, and is designed for scalability.

You learn more about Rule Execution Server in Unit 13, "Executing rules with Rule Execution Server".

## Setting up a Rule Execution Server

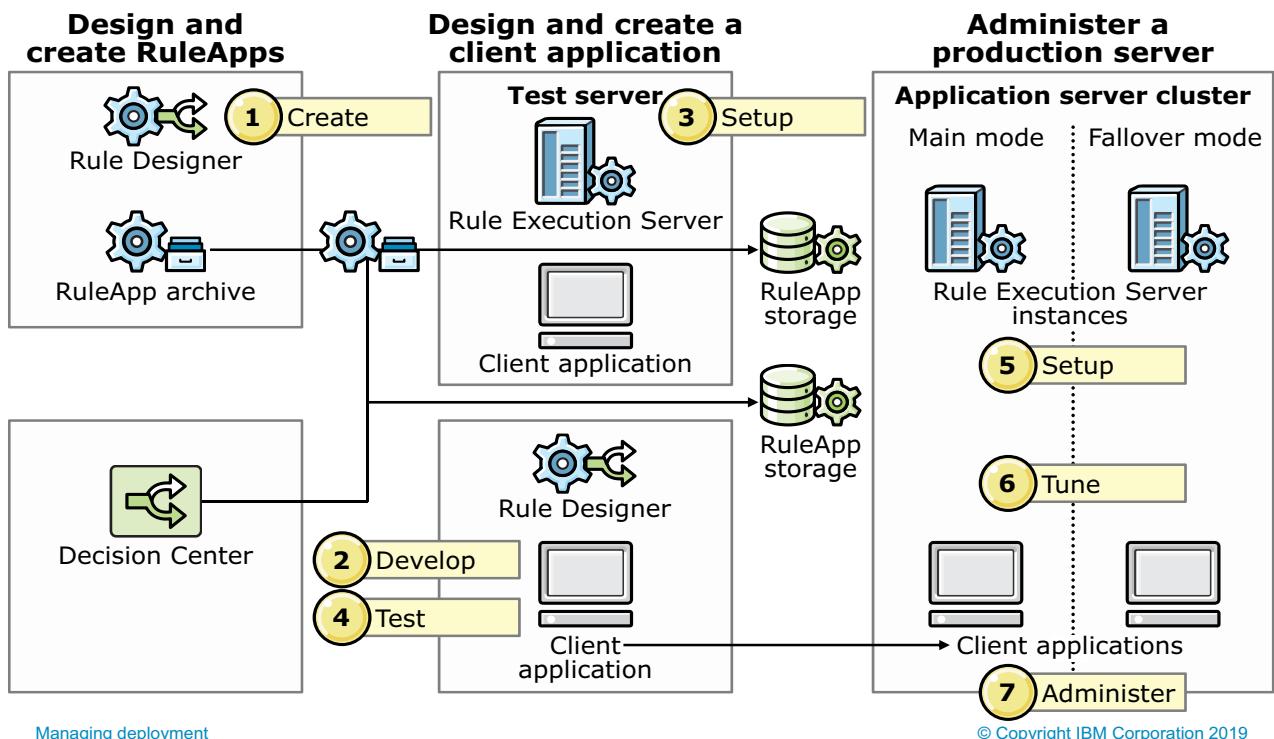


Figure 12-6. Setting up a Rule Execution Server

Architects, administrators, developers, and testers work together to create, deploy, tune, and administer RuleApps and client applications that run on Rule Execution Server instances on a production enterprise cell.

To use Rule Execution Server, follow the task flow to complete the steps that are applicable to you. The order in which tasks are done depends on your environment.

The diagram shows the task flow to set up a Rule Execution Server, and the infrastructure that is used at each step. The highlighted steps are administrative tasks that are explained in more detail on the next slide.

### 1. Create

Developers design and create a RuleApp from a rule project. A RuleApp is a deployment and management unit for Rule Execution Server. A RuleApp contains one or more rulesets. RuleApps are deployed to the application server to make the ruleset available to a client application.

### 2. Set up

Administrators set up test servers and production servers. In this course, Rule Execution Server is installed on the Sample Server.

### 3. Deploy

Developers or administrators can deploy the RuleApp to one or more Rule Execution Servers.

In your enterprise environment, you might have several Rule Execution Servers running on different application servers, such as:

- Test servers.
- Production servers. Your production server can involve two or more application servers in a cluster that are installed on two or more computers.

#### 4. Develop

Developers also design and create a client application that uses the deployed rules. The client application must contain execution code that calls the deployed ruleset that is contained in a RuleApp on Rule Execution Server.

#### 5. Test

To test ruleset execution, you must run the client application, which calls the ruleset.

Before deploying RuleApps and client applications to a production server, developers and testers must test with sufficient data that the applications are stable and that the results are correct. You can also run tests to verify that your RuleApps return valid results and set up Decision Warehouse to store execution traces.

Client applications are standard Java applications that you test and debug with Rule Designer. Use the JUnit framework for formalized testing and performance monitoring.

#### 6. Tune

The goal of performance tuning is to decrease the amount of time and resources that your application server requires to process requests.

Administrators can tune Rule Execution Server and server settings in a production cluster to achieve the best performance. For example:

- Log
- Trace level
- Thread pool
- Garbage collection

Developers can also look at the rulesets that are executed as decision services. Performance can be improved by reducing the ruleset size and by using mutually exclusive rule tasks.

#### 7. Administer

Administrators can work in Rule Execution Server console to monitor decision services to provide stability.

## 12.2. Deploying decision services

## Deploying decision services

Managing deployment

© Copyright IBM Corporation 2019

*Figure 12-7. Deploying decision services*

## Deployment principles

- Prerequisites:
  - Rulesets are defined with decision operations
  - Target architecture is known
- Deployment phases:
  - Create a deployment configuration
  - Define the decision operation to deploy
  - Define the target server in the deployment configuration
  - Deploy to the identified Rule Execution Server

Figure 12-8. Deployment principles

The next steps after business rule development are to package and deploy the rules to your execution environment. For managed execution of rules with Rule Execution Server, rulesets are packaged in *RuleApps* before you deploy them to Rule Execution Server.

## RuleApp

- Deployable management unit that contains one or more rulesets
- A deployed ruleset contains an archive, which supports its execution
- For classic rule projects, RuleApps are handled inside RuleApp projects
  - Contain the RuleApp descriptor and information such as the Rule project dependencies and the ruleset archive paths
- For decision services, the deployment configuration generates the RuleApp archive that contains the rulesets during the deployment process

Figure 12-9. RuleApp

A RuleApp is a deployable management unit that contains one or more rulesets. A deployed ruleset contains an archive, which supports its execution. A RuleApp is described in an XML file that is named a RuleApp descriptor. All executable resources are defined as elements of the RuleApp descriptor.

When you deploy a decision service with a deployment configuration, Rule Designer generates a RuleApp archive that contains the rulesets. These rulesets are defined in the decision operations in the decision service.

## Ruleset path

- Each ruleset in a RuleApp can be identified by using a ruleset path:  
`/ {RuleApp name} [/ {version}]/{ruleset name} [/ {version}]`
- The ruleset path acts as the entry point for clients to access the business logic that is encapsulated in the RuleApp
- In a client application, the ruleset path identifies the ruleset to execute
- You can omit the version fields in the ruleset path to refer to the latest deployed ruleset

Figure 12-10. Ruleset path

Each ruleset in a RuleApp is identified with a unique ruleset path:

`/ {RuleApp name} [/ {RuleApp version}]/{ruleset name} [/ {ruleset version}]`

The ruleset path acts as the entry point for client applications to access the business logic that is encapsulated in the RuleApp. In a client application, the ruleset path identifies the ruleset to execute. You can omit the version fields in the ruleset path so that Rule Execution Server executes the latest deployed ruleset.

## Deployed RuleApp and ruleset names

- In names of RuleApps and rulesets that are deployed to Rule Execution Server, you can use only:
  - Letters (a–z, A–Z)
  - Digits (0–9)
  - The underscore character (\_)
- Because of this difference, the names of a deployed RuleApp and of its rulesets in Rule Execution Server might differ from these names in Rule Designer or Decision Center Console
- The ruleset path is based on the RuleApp name and on the ruleset name as they exist in Rule Execution Server

*Figure 12-11. Deployed RuleApp and ruleset names*

Names of RuleApps and rulesets that are deployed to Rule Execution Server can contain only letters (a–z, A–Z), digits (0–9), and the underscore character (\_).

When a RuleApp is deployed to Rule Execution Server, the RuleApp name and the name of its rulesets in Rule Execution Server might differ from what you initially had in Rule Designer or Decision Center. Characters that are not authorized are removed or modified.

When you use ruleset paths, for example in a client application, you must base them on the RuleApp names and on the ruleset names as *they exist in Rule Execution Server*.

### 1+1=2 Example

Consider a `loan-deployRuleApp` RuleApp packaging a `loan-rules` ruleset. After you deploy `loan-deployRuleApp` to Rule Execution Server, you see the following elements in the Rule Execution Server console:

- `loan_deployRuleApp`
- `loan_rulesRuleset`

The ruleset path for the ruleset is equal to:

loan\_deployRuleApp/<RuleApp version>/loan\_rulesRuleset/<ruleset version>

---

## RuleApp archive

- **Reminder:** You create a ruleset archive to pass a ruleset to the rule engine for its execution
- Similarly, you can create a RuleApp archive from a RuleApp, and then deploy this RuleApp archive to create the RuleApp on Rule Execution Server
- Rule Designer generates the RuleApp archive when you deploy the RuleApp to Rule Execution Server

Figure 12-12. RuleApp archive

You must create a ruleset archive to pass your rules to the rule engine for their execution.

Similarly, you create a RuleApp archive from your RuleApp, and deploy this RuleApp archive to create the RuleApp on Rule Execution Server.

You can export a RuleApp archive by opening the `archive.xml` file in your RuleApp project and by selecting **Export a RuleApp archive** on the **Overview** tab.

In Rule Designer, when you ask to deploy the RuleApp to Rule Execution Server, Rule Designer creates the RuleApp archive for you.

## RuleApp properties and ruleset properties

- Several RuleApp properties and ruleset properties are predefined to handle the most common requirements in terms of configuration of the rule engine behavior at run time
- Examples:
  - `rulesetSEQUENTIALTRACEenabled`
  - `ruleappINTERCEPTORclassname`
- You can set properties to a RuleApp, applicable to all rulesets in this RuleApp
- You can set properties to a ruleset, applicable to this ruleset only

*Figure 12-13. RuleApp properties and ruleset properties*

Several RuleApp and ruleset properties are predefined to handle the most common requirements in terms of configuration of the rule engine behavior at run time, including the following ones:

- `rulesetSEQUENTIALTRACEenabled` is a predefined ruleset property that you use to enable, or disable, the trace mode of the rule engine for business rules that are executed with the sequential execution mode or the Fastpath execution mode.
- `ruleappINTERCEPTORclassname` is a predefined RuleApp property that you use to define the ruleset interceptor that applies to the rulesets in your RuleApp.

A ruleset interceptor is a piece of code in your client application that you use to select rulesets at run time, and to add services to execution components transparently. For example, with a ruleset interceptor, you can check your ruleset paths, complement them as required, and modify the input parameters. Ruleset interceptors can also work on predefined or user-defined ruleset properties. Ruleset interceptors are not explained in further detail in this course.

You can set properties on your RuleApp. These properties apply to all the rulesets in this RuleApp. To set RuleApp properties, open the `archive.xml` file in your RuleApp project, and use the **RuleApp Properties** grid on the **Overview** tab to add, edit, or remove RuleApp properties.

You can also set properties on your rulesets in your RuleApp. These properties apply to your ruleset only. To set ruleset properties, you open the `archive.xml` file in your RuleApp project. On

the **Ruleset Archives** tab, select the ruleset where to set properties. You can then add, edit, or remove ruleset properties in the **Ruleset Properties** grid.

**Note**

You can also set RuleApp properties and ruleset properties by using the Rule Execution Server console.

## Deployment configurations (1 of 4)

- From Rule Designer and Decision Center Business console, you can deploy RuleApps by using a deployment configuration
- Decision operations
  - Configurations can reference one or more decision operations
- Target servers
  - Configurations can reference one or more target servers
  - Target servers can be defined for nonproduction or production
- Create more than one deployment configuration for a decision service
  - Each deployment configuration can serve a different purpose
  - Example, one can be used for testing and another for deploying to a production server

Figure 12-14. Deployment configurations (1 of 4)

To deploy your RuleApps from Rule Designer or from Decision Center Business console, you create a *deployment configuration* to tell Rule Designer which Rule Execution Server instance is your target server.

You indicate whether a deployment configuration is for a nonproduction or production environment. The nonproduction setting limits the target servers that can be used with the configuration. It also adds some restrictions on deployment from the Business console within the governance framework. When a release of a decision service is complete in the governance framework, you can deploy it to a production server.

# IBM Training

## Deployment configurations (2 of 4)

**General**

Name: loan-deploy

Type:  Production  Nonproduction Deploy the XOM:  Yes  No

Description:

**Decision Operations**

Define decision operations for RuleApp deployment:  
loan-rulesOperation

**RuleApp**

Configure the RuleApp to be created upon deployment.

|                       |             |
|-----------------------|-------------|
| RuleApp Name:         | loan_deploy |
| RuleApp Base Version: | 1.0         |

RuleApp Properties:

| Name | Value |
|------|-------|
|      |       |

**Target Servers**

Define target servers for RuleApp deployment:  
Sample

**Deployment**

[Proceed to RuleApp deployment...](#)

Overview | Decision Operations | Target Servers | Source

Managing deployment

© Copyright IBM Corporation 2019

Figure 12-15. Deployment configurations (2 of 4)

A deployment configuration includes the following information:

- A configuration type: production or nonproduction. This option can be used to limit deployment to certain servers.
- The RuleApp name that is used by the client application to request a decision.
- The decision operations with the rules to be deployed.
- The target servers.
- The ruleset version base number and version policy.



## Deployment configurations (3 of 4)

**Deployment, Decision Operations - loan-deploy**

**Configured Decision Operations**  
Select and configure decision operations for RuleApp deployment:

**loan-rules**  
loan-rulesOperation

**Ruleset General Behavior**

Base Version:  
 Enabled     Debug     Trace

**Ruleset Properties**  
Add predefined or custom properties to the ruleset to specify extra information.

| Name | Value |
|------|-------|
|      |       |

**Ruleset Version Policy**  
Select the version policy that applies upon deployment:

Increment minor version numbers  
Updates the minor version for each ruleset. Makes the new version available but retains previous versions.

Use the base version numbers  
Uses the numbers provided in the deployment configuration. Replaces the latest version of each ruleset with this release. Used for hot fixes or

The user can define the version numbers

[Overview](#)
[Decision Operations](#)
[Target Servers](#)
[Source](#)

Managing deployment

© Copyright IBM Corporation 2019

Figure 12-16. Deployment configurations (3 of 4)

Define the decision operations, ruleset properties, and version policy on the **Decision Operations** tab.



## Deployment configurations (4 of 4)

- One or more target servers are defined on the **Target Servers** tab
  - Server definitions are saved as part of the workspace properties

## Managing deployment

© Copyright IBM Corporation 2019

*Figure 12-17. Deployment configurations (4 of 4)*

## Deployment from Decision Center

- Business users with administrator rights can create or edit deployment configurations
  - When working within the governance framework, a deployment configuration can be created or edited only within a change activity
  - The deployment configuration also contains the groups that can use that configuration to deploy, and options for managing snapshots
- Release state and permissions in the Business console determine which target servers can be used
  - Business users are limited to test environments
  - Administrators can deploy to production



Managing deployment

© Copyright IBM Corporation 2019

Figure 12-18. Deployment from Decision Center

Business users with administrator rights can deploy RuleApps from Decision Center to Rule Execution Server. They can create or edit deployment configurations.



### Important

From Rule Designer, you define the decision operations. Decision operations cannot be edited in the Business console. Business users can edit which decision operations are referenced by a deployment configuration, but they cannot create or edit the decision operation.

If your decision service uses a managed Java XOM, deploy the Java XOM to the appropriate Rule Execution Server instance first. Then, publish your decision service or rule project from Rule Designer to Decision Center, and deploy from there.



## Deployment from Rule Execution Server console

- You can create a RuleApp container and load the existing rulesets to Rule Execution Server through the console

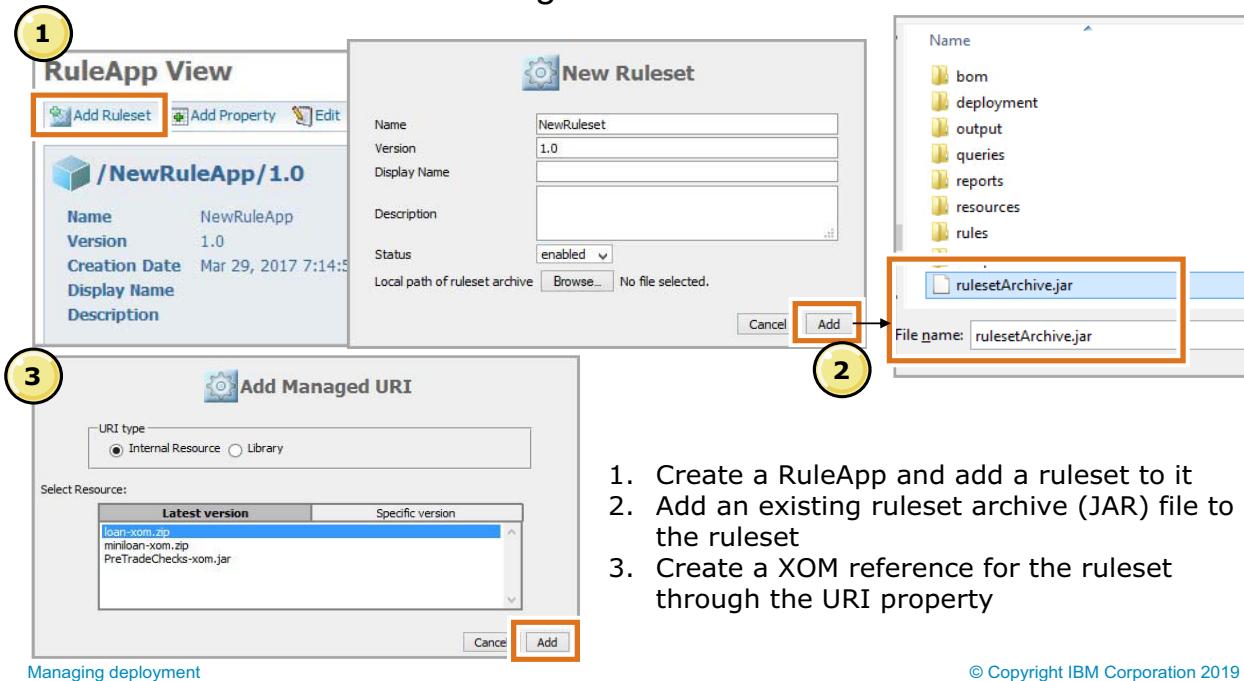


Figure 12-19. Deployment from Rule Execution Server console

Administrators and developers use the Rule Execution Server console to manage and monitor rule execution in Rule Execution Server. Less technical business users do not use Rule Execution Server console.

You can create, load, and manage RuleApps, rulesets, and XOM resources directly through the console.

## 12.3. Managing XOMs

## Managing XOMs

Managing deployment

© Copyright IBM Corporation 2019

*Figure 12-20. Managing XOMs*

In this topic, you learn how to manage XOMs with Rule Execution Server.

## XOM deployment with decision services

- XOM management ensures that any RuleApp that is deployed to Rule Execution Server references the correct XOM
- With XOM management, when you deploy a RuleApp to Rule Execution Server, Rule Designer follows this process:
  1. Builds the XOM binary from the source files that are referenced by the decision service
  2. Deploys the XOM (if it is not already present in Rule Execution Server) and retrieves the XOM URI
  3. Copies this URI to each ruleset in the RuleApp

Figure 12-21. XOM deployment with decision services

Rule Designer copies the URI to a property of the ruleset called `ruleset.managedxom.uris`. With this mechanism, each ruleset in Rule Execution Server references the correct XOM.

## Modifying the XOM

- You modify the XOM source files in Rule Designer
  - Rule Designer always builds the XOM binary from the source files
- When you import a decision service from Decision Center into an empty workspace, you do not obtain the XOM source files
  - You must obtain these source files in the usual way, for example through source code control
- The XOM binary obtained from Decision Center is copied to the resources/xom-libraries folder
  - The files in this folder are meant to be synchronized only and they should not be used in the XOM path

Figure 12-22. Modifying the XOM

## Managed Java XOM elements

- You can manage the Java XOM *resources* and *libraries* that are attached to a ruleset
- A Java XOM *resource* is a `.jar` file or a `.zip` file
  - You identify such a resource with a unique URI based on the `resuri` protocol
  - Example: `resuri://common-classes.jar/1.0`
- A Java XOM *library* is a set of resources or libraries, or both
  - You identify such a library with a unique URI based on the `reslib` protocol
  - Example: `reslib://loan-xom/1.4`

*Figure 12-23. Managed Java XOM elements*

The Java XOM resources are either:

- JAR files: Java archive files, with the `.jar` extension
- ZIP files: Compressed files that contain Java classes and the files that they use

The `resuri` URI is created with the name of the file and a version, which is incremented each time the file checksum changes.

## Link from a ruleset to a managed Java XOM element

- The `ruleset.managedxom.uris` ruleset property defines the link between a specific ruleset and a specific managed Java XOM element
- The value of this ruleset property is a list of URIs to .jar files, .zip files, or libraries
- A specific ruleset is linked to a specific managed Java XOM element if:
  - This ruleset defines the `ruleset.managedxom.uris` ruleset property
  - The value of this ruleset property contains the URI to this Java XOM element

Figure 12-24. Link from a ruleset to a managed Java XOM element

The value of the `ruleset.managedxom.uris` ruleset property must contain only URIs that follow the `resuri` protocol or the `reslib` protocol. In the value, the URIs are comma-separated.



## Java XOM deployment

- By default, the deployment configuration for a decision service includes deployment of the XOM



- The deployed Java XOM is stored in the Rule Execution Server persistence layer and can be managed in the same way as a RuleApp
- When you deploy a RuleApp along with its Java XOM, Rule Designer sets the `ruleset.managedxom.uris` ruleset property in the ruleset that is packaged in the deployed RuleApp

[Managing deployment](#)

© Copyright IBM Corporation 2019

Figure 12-25. Java XOM deployment

You can deploy a Java XOM with the RuleApp in the deployment configuration. Or, you can deploy the XOM separately in Rule Designer by right-clicking the main rule project and clicking **Rule Execution Server > Deploy XOM**.

When you deploy the Java XOM with the RuleApp, Rule Designer defines and sets the `ruleset.managedxom.uris` ruleset property.

## XOM deployment from Decision Center

- Decision services that are published to Decision Center include XOM libraries
  - When Rule Designer builds the XOM from the source files, it places the XOM binary under `resources/xom-libraries`
  - In Rule Designer, on the Properties page for the main decision service project, select **Enable XOM management in Decision Center**
  - The `resources` folder is synchronized between Rule Designer and Decision Center

Figure 12-26. XOM deployment from Decision Center

## Managed XOM and Decision Center

- When you deploy from Decision Center, the `ruleset.managedxom.uris` property is added to the ruleset in the deployed RuleApp
- When you modify the XOM:
  - Redeploy it to the Rule Execution Server to update the `ruleset.managedxom.uris` property
  - Synchronize the decision service between Rule Designer and Decision Center

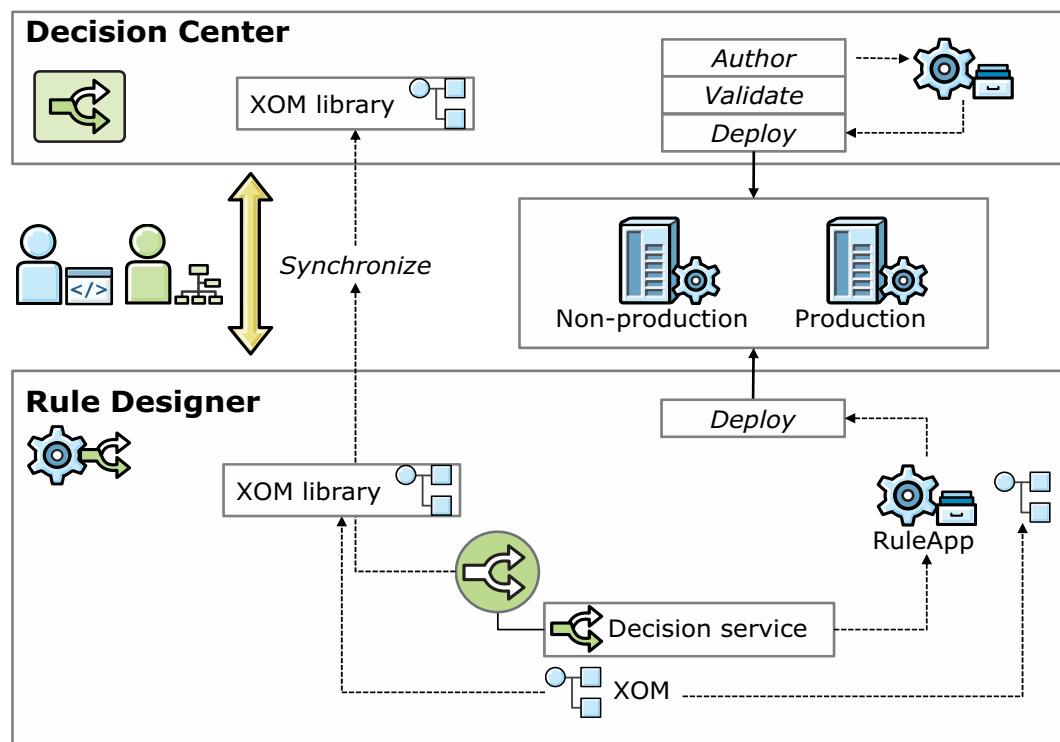
Figure 12-27. Managed XOM and Decision Center

To make sure that a decision service in Decision Center can execute when it is deployed to Rule Execution Server, the XOM must also be deployed to that Rule Execution Server instance.

If you modify the XOM after publishing a decision service to Decision Center, make sure you redeploy the XOM to Rule Execution Server to ensure that the managed XOM URI is updated. Then, synchronize the decision service between Rule Designer and Decision Center.



## Managed XOMs in Rule Designer and Decision Center



Managing deployment

© Copyright IBM Corporation 2019

Figure 12-28. Managed XOMs in Rule Designer and Decision Center

This diagram recaps the actions that developers or business analysts complete in Rule Designer, the actions that business users complete in Decision Center regarding the managed XOM, and the data that these actions work on.

1. Programming with rules starts in Rule Designer, where:
  - a. Developers and business analysts design the XOM based on data model.
  - b. Developers or business analysts create the decision services with the BOM based on the XOM, the associated vocabulary, and some rule artifacts.
2. Developers or business analysts synchronize the decision service between Rule Designer and Decision Center.
3. After synchronization, business users can author the required rule artifacts in Decision Center.
4. By default, the XOM is deployed with the decision service to the Rule Execution Servers.
5. If the XOM does not evolve, then it is possible to synchronize Decision Center and Rule Designer environments at any time, thus ensuring that all actors of the rule application are using the same artifacts.
6. If the XOM evolves in Rule Designer, then the XOM must be redeployed from Rule Designer to the test or production execution environment. The project must also be synchronized to ensure that the latest version of the XOM is also known in Decision Center.

As guidelines:

- Deploy the XOM each time it is changed.
- Between two XOM changes, the BOM and rule artifacts (the rule project) can be synchronized between the environments as required.

## Embedded managed Java XOMs

- Managed Java XOM resources (.jar) and libraries that are included in a RuleApp archive
- Embed managed Java XOMs in a RuleApp archive to maintain the version of the managed Java XOM and the version of the corresponding ruleset together
- Use embedded managed Java XOM feature from
  - Rule Execution Server console
  - REST APIs
  - Ant tasks
  - Build Command

Figure 12-29. *Embedded managed Java XOMs*

Embedded managed Java XOMs are managed Java XOM resources (.jar) and libraries that are included in a RuleApp archive. By embedding managed Java XOMs in a RuleApp archive, you can easily maintain the version of the managed Java XOM and the version of the corresponding ruleset together.

## Managed XOM storage

- You can store the managed Java XOM in the persistence layer of Rule Execution Server independently of the ruleset
  - For example, in a test environment, you can store ruleset archives locally if they are too large to work with from a database

*Figure 12-30. Managed XOM storage*

You can store the managed Java XOM independently of the ruleset, in the persistence layer of Rule Execution Server.

## 12.4. Managing resources with Ant tasks

## Managing resources with Ant tasks

Managing deployment

© Copyright IBM Corporation 2019

*Figure 12-31. Managing resources with Ant tasks*

In this topic, you learn how to manage XOMs with Rule Execution Server.

## Deployment and management with Ant

- To automate RuleApp management, use Ant scripts
  - Create a `build.xml` file as an Ant script
- Example: To deploy a RuleApp archive, you run this command from the container directory

```
ant res-deploy -Dserver.port=9090
```

- Ant commands include:

| Ant commands | Task                       |
|--------------|----------------------------|
| res-jar      | Create RuleApps            |
| res-deploy   | Deploy RuleApp archives    |
| res-fetch    | Download a RuleApp archive |
| res-undeploy | Remove a RuleApp archive   |

Figure 12-32. Deployment and management with Ant

You can use Ant to automate your RuleApp management, in particular RuleApp deployment.

## Setting up the Ant environment

- Ant is packaged with ODM in the following directory:

*InstallDir/shared/tools/ant*

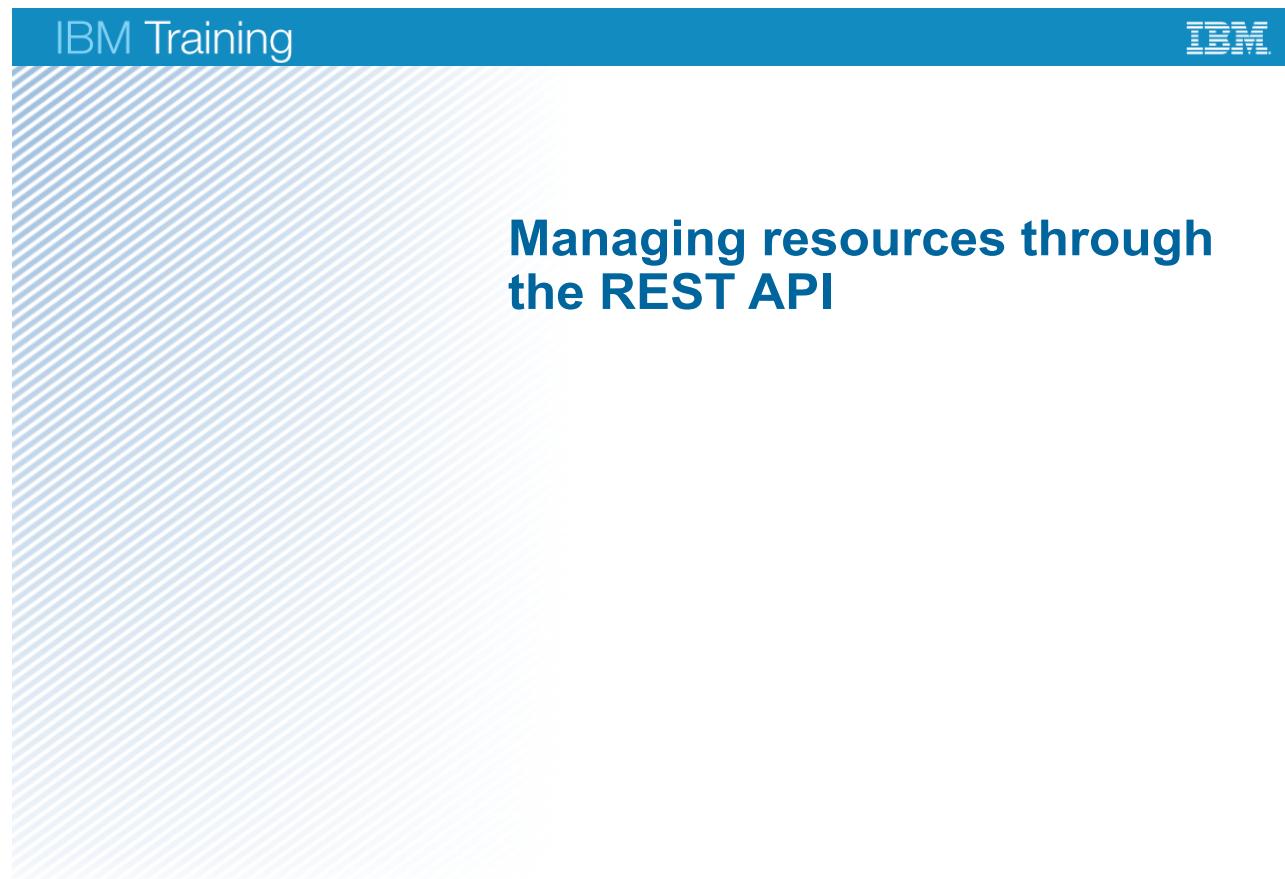
- Set the **ANT\_HOME** environment variable to:

*InstallDir/shared/tools/ant*

- Set the Path environment variable to: ***ANT\_HOME/bin***

Figure 12-33. Setting up the Ant environment

## 12.5. Managing resources through the REST API



Managing deployment

© Copyright IBM Corporation 2019

*Figure 12-34. Managing resources through the REST API*



## REST API tool in Rule Execution Server console

- Use the **REST API** tab in the Rule Execution Server console to manage REST resources through HTTP methods

REST API WADL file: /res/apiauth/v1/DecisionServer.wadl

**/ruleapps**

**GET** /ruleapps  
getRuleApps Returns all the RuleApps contained in the repository.

**GET** /ruleapps?count=true  
getCountOfRuleApps Counts the number of elements in this list.

**POST** /ruleapps  
deployRuleAppArchive Deploys a RuleApp archive in the repository, based on the merging and versioning policies passed as parameters. The RuleApp archive is request body.

**POST** /ruleapps  
addRuleApp Adds a new RuleApp in the repository. The RuleApp representation is passed in the request body in JSON or XML format. If the RuleApp already exists response body contains a specific error description and the HTTP status 202 is returned.

Managing deployment

© Copyright IBM Corporation 2019

Figure 12-35. REST API tool in Rule Execution Server console

## 12.6. Building RuleApps with the Build Command Maven plug-in



Figure 12-36. Building RuleApps with the Build Command Maven plug-in

## Building RuleApps with Build Command

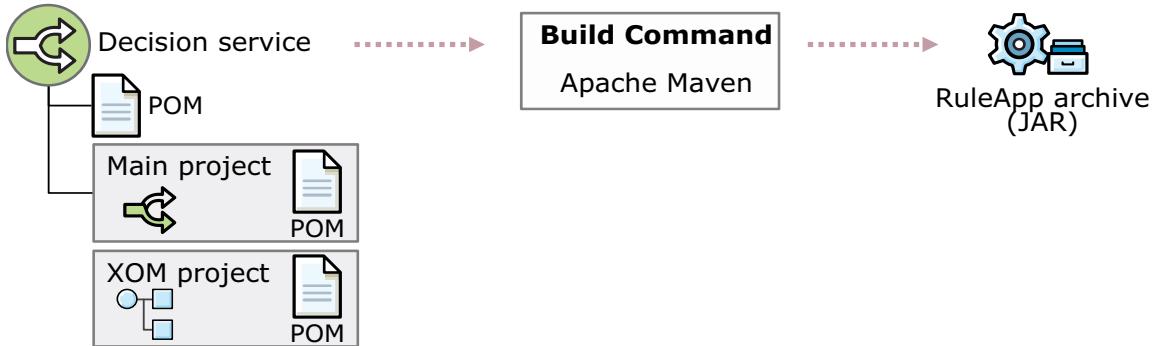
- Build rule and Java projects as rule applications outside of Eclipse with Build Command Maven plug-in
- Requirements:
  - The Build Command Maven plug-in requires Apache Maven and Java
  - The Path environment variable must include %JAVA\_HOME%/bin
- Maven repository
  - You can configure the Maven repository (local or remote) in the configuration settings file
  - Default repository: \${user.home}/.m2/repository

Figure 12-37. Building RuleApps with Build Command

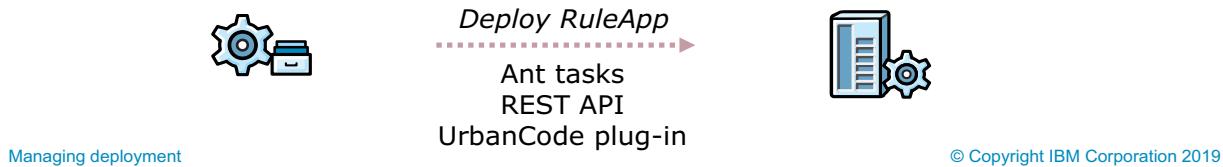
You can build your rule and Java projects as rule applications outside of Eclipse by using the Maven plug-in of Build Command.

## Build and deploy

- Build Command generates one or more RuleApp files (JAR) for the project



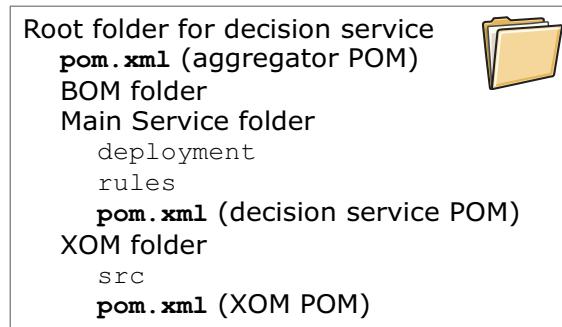
- JAR files can be deployed to Rule Execution Server through Ant tasks, REST API or UrbanCode



*Figure 12-38. Build and deploy*

## Project Object Model (POM) files

- Project Object Model (POM) files
  - Decision service POM for the deployment configuration in the decision service
  - XML POM for the XOM
  - Aggregator XOM
- The aggregator POM and XOM POM files are optional depending on your project structure and how you want to manage projects
- Project structure
  - For decision services, the POM file belongs in the main decision service project folder



Managing deployment

© Copyright IBM Corporation 2019

*Figure 12-39. Project Object Model (POM) files*

The Project Object Model (POM) files are required for the plug-in to know the relationship between projects and how to build them. You must write one decision service POM file for each decision service project.

If you are using Maven in a continuous integration pipeline, then your XOM project is probably already a Maven project, with its own POM file, and you don't need to write a new one. Also, if your XOM project is a Maven project and you want to manage its lifecycle separately from the decision service project, you do not need the aggregator POM.

For more information, see the product documentation:

[www.ibm.com/support/knowledgecenter/en/SSQP76\\_8.10.x/com.ibm.odm.dserver.rules.designer.run/build\\_topics/con\\_buildcmd\\_setup\\_proj.html](http://www.ibm.com/support/knowledgecenter/en/SSQP76_8.10.x/com.ibm.odm.dserver.rules.designer.run/build_topics/con_buildcmd_setup_proj.html)

## Example decision service POM file (1 of 2)

```

<parent>
    <groupId>loanservice</groupId>
    <artifactId>parent</artifactId>
    <version>1.0.0</version>
    <relativePath>..</relativePath>
</parent>

<artifactId>loan-rules</artifactId>
<packaging>decisionservice</packaging>

<build>
<plugins>
    <plugin>
        <groupId>com.ibm.rules.buildcommand</groupId>
        <artifactId>rules-compiler-maven-plugin</artifactId>
        <configuration>
        <deployments>
            <deployment>
                <name>loan-deploy</name>
            </deployment>
        </deployments>
    ...

```

Managing deployment

© Copyright IBM Corporation 2019

*Figure 12-40. Example decision service POM file (1 of 2)*

You must define a `groupId` value to identify the set of related decision service projects. This `groupId` is also used in the aggregator POM and the XOM POM files to identify the relationship.

You also provide an `artifactID` to identify the decision service. In the `<deployments>` section of the POM file, you include the name of the deployment configuration as defined in your decision operation.

## Example decision service POM file (2 of 2)

```

...
<resolvers>
    <resolver>
        <kind>JAVA_PROJECT</kind>
<url>platform:/loan-xom</url>
    <artifactKey>${project.groupId}:loan-xom</artifactKey>
    </resolver>
</resolvers>
</configuration>
</plugin>
</plugins>
</build>
<dependencies>
    <dependency>
        <groupId>${project.groupId}</groupId>
        <artifactId>loan-xom</artifactId>
        <type>jar</type>
        <version>${project.version}</version>
    </dependency>
</dependencies>

```

*Figure 12-41. Example decision service POM file (2 of 2)*

The `<resolvers>` section identifies the XOM to use. This information must match the XOM information in the `.ruleapp` file for your decision service.

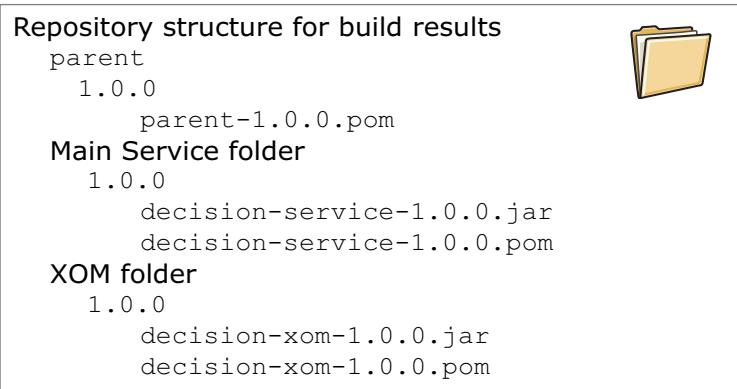
In the `<dependencies>` section, you identify the XOM project that is referenced by your decision service.

## Build results

- From the root directory of the decision service project, run the build command

```
mvn clean install
```

- The Build Command generates a target directory that contains one `.jar` file per deployment configuration in the directory of each project that you build
  - Each `.jar` file is a RuleApp archive
  - The Maven repository



*Figure 12-42. Build results*

If you specified an embedded managed Java XOM in a deployment parameter, the corresponding RuleApp archive also contains the managed Java XOM files.

## Unit summary

- Describe the principles for managing RuleApp and XOM deployment
- Prepare deployment configurations
- Build and deploy RuleApps outside of Rule Designer

Figure 12-43. Unit summary

## Review questions

1. True or False: Rulesets must be packaged as RuleApps for deployment.
2. True or False: Deployment is an administrative task that cannot be performed in Decision Center.
3. What tools can you use to deploy rules to Rule Execution Server?
  - A. Rule Designer
  - B. Decision Center Business console
  - C. Rule Execution Server console
  - D. Ant tasks
  - E. All of the above
4. True or False: You can use the Build Command Maven plugin to package RuleApps for deployment.

Managing deployment

© Copyright IBM Corporation 2019

Figure 12-44. Review questions

Write your answers here:

- 1.
- 2.
- 3.
- 4.

## Review answers

1. True or False: Rulesets must be packaged as RuleApps for deployment.  
**The answer is True.**
2. True or False: Deployment is an administrative task that cannot be performed in Decision Center.  
**The answer is False. Business users can deploy from Decision Center.**
3. What tools can you use to deploy rules to Rule Execution Server?
  - A. Rule Designer
  - B. Decision Center Business console
  - C. Rule Execution Server console
  - D. Ant tasks
  - E. All of the above**The answer is E.**
4. True or False: You can use the Build Command Maven plug-in to package RuleApps for deployment.  
**The answer is True.**

Managing deployment

© Copyright IBM Corporation 2019

Figure 12-45. Review answers



## Exercise: Managing deployment

Managing deployment

© Copyright IBM Corporation 2019

*Figure 12-46. Exercise:*

## Exercise introduction

- Define a RuleApp and ruleset properties
- Use deployment configurations to deploy decision services
- Deploy the XOM for its management in Rule Execution Server



Figure 12-47. Exercise introduction

## Exercise: Using Build Command to build RuleApps

Managing deployment

© Copyright IBM Corporation 2019

*Figure 12-48. Exercise:*

## Exercise introduction

- Define POM files
- Build a RuleApp archive from a set of projects



*Figure 12-49. Exercise introduction*

---

# Unit 13. Executing rules with Rule Execution Server

## Estimated time

02:15

## Overview

This unit explains how to create client applications that request the managed execution of business rules with Rule Execution Server. It also covers the various enterprise environments in which Rule Execution Server can run.

## How you will check your progress

- Review
- Exercise

## Unit objectives

- Describe the Rule Execution Server architecture
- Describe the platforms in which Rule Execution Server can be deployed
- Explain the APIs that are used to create client applications that request ruleset execution with Rule Execution Server

Figure 13-1. Unit objectives

## Topics

- Managed execution with Rule Execution Server
- Rule Execution Server modular architecture
- Managed execution in action
- Platforms for Rule Execution Server
- Introducing the Rule Execution Server API
- Executing rules in Java SE
- Executing rules in Java EE
- Executing rules as transparent decision services

Figure 13-2. Topics

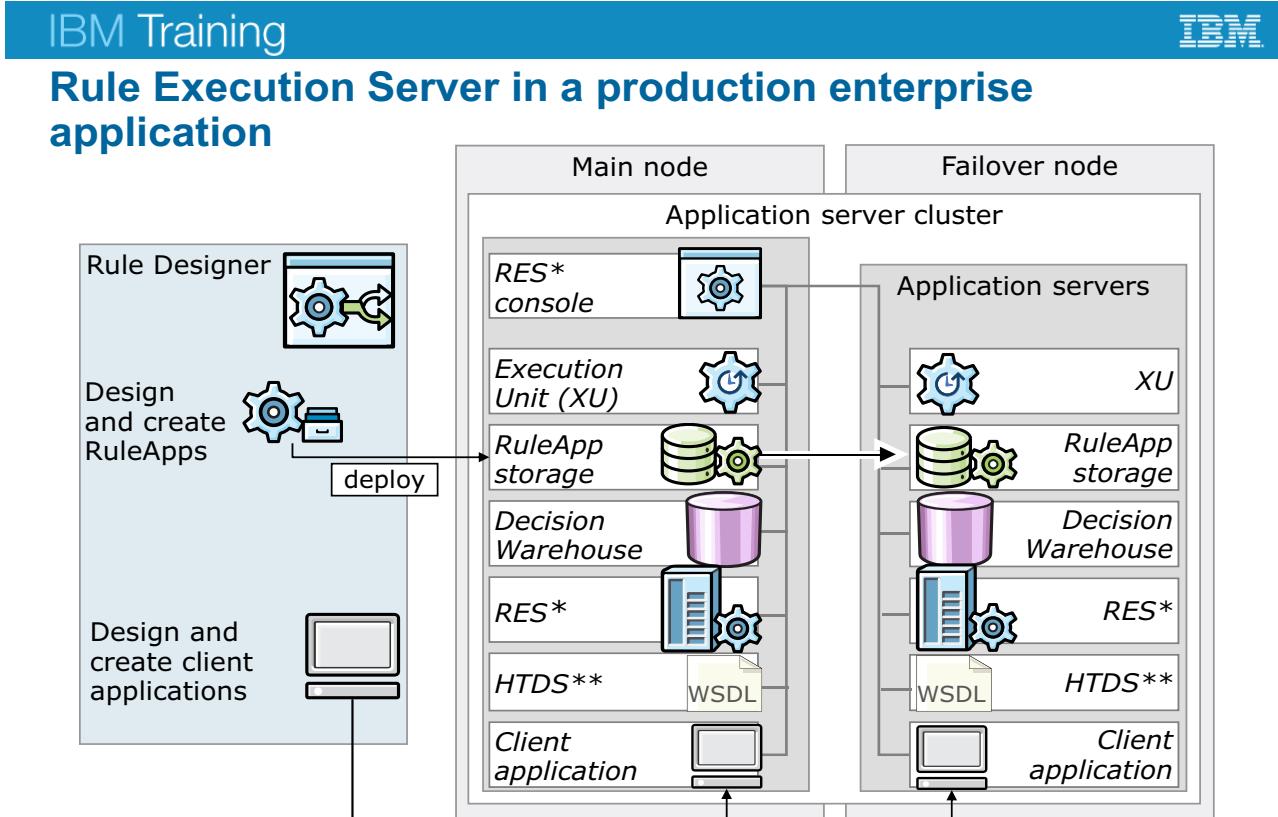
## 13.1. Managed execution with Rule Execution Server

## Managed execution with Rule Execution Server

Executing rules with Rule Execution Server

© Copyright IBM Corporation 2019

*Figure 13-3. Managed execution with Rule Execution Server*



\* RES = Rule Execution Server

\*\* HTDS = Hosted transparent decision service

Executing rules with Rule Execution Server

© Copyright IBM Corporation 2019

Figure 13-4. Rule Execution Server in a production enterprise application

Rule Execution Server architecture is based on a number of independent, but cooperating, software modules, as you see depicted in this diagram.

During this unit:

- You explore the Rule Execution Server modular architecture that supports flexibility.
- You learn how the Rule Execution Server modules can be deployed to match your requirements.
- You are introduced to the required API to write Java client applications.
- Finally, you put it all together (API, Rule Execution Server components, and deployment).
- You also see how to execute rules within the Java SE environment, the Java EE environment, and as a hosted transparent decision service.

## Introducing managed execution with Rule Execution Server

- Deploy RuleApp once, execute in multiple environments
- Create the client application according to your enterprise context
  - The modular architecture of Rule Execution Server supports flexibility
  - Rule Execution Server components are deployable on various platforms
- Use rule engine API and Rule Execution Server API to write the client application according to the chosen platform
  - Java SE
  - Java EE
  - Web services for service-oriented architectures (SOA)
  - **Note:** Rule Execution Server API is recommended for decision engine

Figure 13-5. Introducing managed execution with Rule Execution Server

## Example of possible platforms

- Within a Java SE application
  - Rule Execution Server is used as an execution service that is embedded in Java SE together with the rule application
- Within a Java EE application
  - You host Rule Execution Server in an application server
- Through web services
  - Use hosted or monitored transparent decision services to access to Rule Execution Server through web services
  - Suitable for SOA

*Figure 13-6. Example of possible platforms*

Typically, you execute your business rules with Rule Execution Server:

- Within a Java SE application: Rule Execution Server that is used as an execution service, which is embedded in Java SE, together with the rule application.
- Within a Java EE application: Rule Execution Server is hosted on an application server.
- Through web services: Use hosted or monitored transparent decision services to access to Rule Execution Server through web services, which are suitable for a service-oriented architecture (SOA).
- By using a SCA component, in either Java SE or Java EE: Your client code directly accesses RuleApps and rulesets in Rule Execution Server. This configuration is supported on some servers only, and is not covered in this course. For more information, see the product documentation.

## 13.2. Rule Execution Server modular architecture

## Rule Execution Server modular architecture

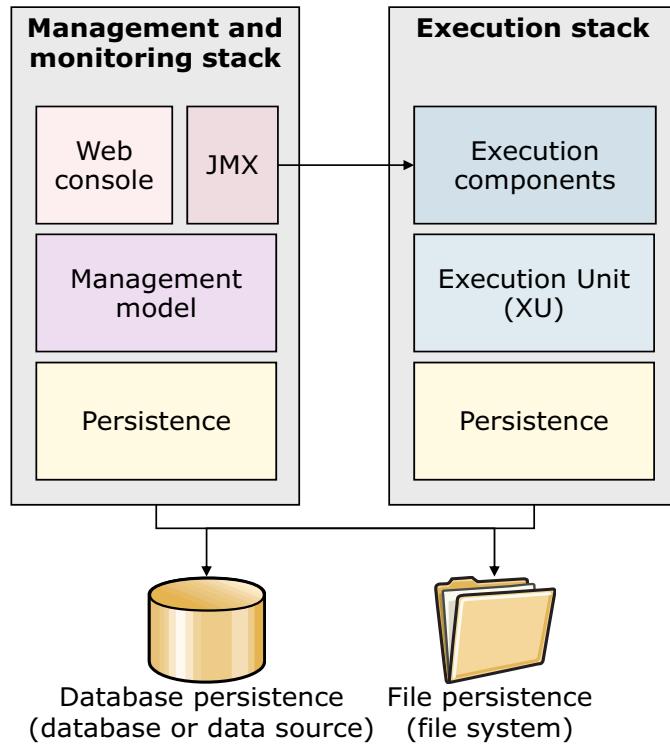
Executing rules with Rule Execution Server

© Copyright IBM Corporation 2019

*Figure 13-7. Rule Execution Server modular architecture*

## Rule Execution Server architecture

- Modular architecture
  - Management and monitoring stack
  - Execution stack
  - Persistence layer
- Set of independent, cooperating components that interact with the rule engine
  - Accommodates various enterprise environments
- Choose what and how to integrate within your enterprise infrastructure



[Executing rules with Rule Execution Server](#)

© Copyright IBM Corporation 2019

Figure 13-8. Rule Execution Server architecture

Rule Execution Server has a flexible modular architecture that can service different server clients and integration with enterprise infrastructure.

Rule Execution Server components are gathered within an execution stack, a management and monitoring stack, and a persistence layer.

With this modular architecture, you can select the approach for your enterprise integration that best matches your requirements. You then deploy the Rule Execution Server components and create your client application correspondingly.

As a developer, you must understand this architecture of Rule Execution Server to be able to:

- Build client applications that call externalized business rules (as you learn in this unit)
- Write custom Java SE or Java EE applications or components (an advanced feature, not covered in this course)

Application architects must also understand this architecture, and have a deeper knowledge to complete their tasks, listed here, and not further covered in this course:

- Package and deploy Rule Execution Server and user components on the application server.

Although how to do this deployment is not covered in this course, you can find some information about this topic later in this unit.

- Manage enterprise applications and control access to:
  - Enterprise data stores
  - Runtime client application code that is executing business policy
  - System-level monitoring tools to assess quality of service
  - Reporting tools that capture changes to business policy
  - Analytical tools that monitor business policy execution

In this course for developers, the following sections give only an *overview* of the various Rule Execution Server components of the Rule Execution Server architecture. If you must act as an architect or as an administrator, you can find more details about this architecture in the product documentation.

## Execution stack: Execution components

- Execution components (Java SE and Java EE) authorize the execution of a ruleset by the Execution Unit (XU)
- Java SE execution components:
  - Stateless rule sessions
  - Stateful rule sessions
  - Decision traces
  - Ruleset execution interceptors
- Java EE execution components include the same components as Java SE, plus:
  - Message-driven (asynchronous) rule beans
  - Remote invocation (RMI)

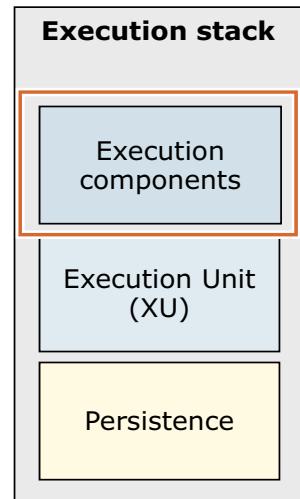


Figure 13-9. Execution stack: Execution components

The Execution stack comprises:

- Java execution components (Java SE and Java EE)
- JMX Execution Model components
- Execution Units (XU, see next slide)

Java SE execution components and Java EE execution components authorize the execution of a ruleset by the *Execution Unit* (XU). Java SE execution components comprise:

- Stateless ruleset sessions (further detailed next)
- Stateful ruleset sessions (further detailed next)
- Decision traces
- Ruleset execution interceptors

Java EE execution components comprise the same as Java SE execution components, plus:

- Message driven (asynchronous) rule beans (further detailed next)
- Remote invocation (RMI)

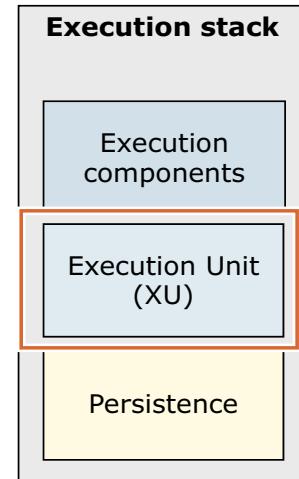


## Information

The execution stack is in essence the same for both Java SE and Java EE. This similarity also exists for the management stack.

## Execution Unit

- A resource adapter for Java EE Connector Architecture (JCA)
  - Handles the low-level aspects of ruleset execution
  - Collaborates with the server to provide several connector system-level contracts as a service provider interface
- Runs independently of the management model
  - Makes configuration and runtime data available to the management model
  - Implements the JCA contracts between the application server and the rule engine
- Benefits:
  - Scalability
  - High-performance execution of rulesets
  - An XU container to create and pool connections of the XU (JCA)
  - Logging, statistics, and debugging
  - Notification that a RuleApp was modified



[Executing rules with Rule Execution Server](#)

© Copyright IBM Corporation 2019

Figure 13-10. Execution Unit

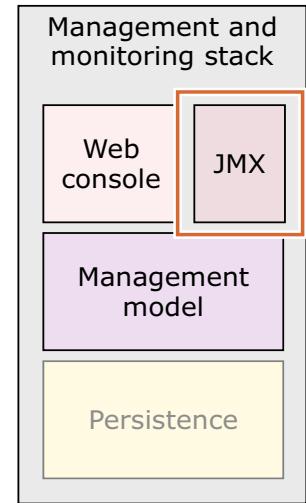
In Rule Execution Server, an Execution Unit (XU) manages rule engines. The XU is a resource adapter for Java EE Connector Architecture (JCA), which handles the low-level details of ruleset execution and provides access to manage its resources. You can also access configuration and runtime data either through a JMX MBean or by using TCP/IP management.

The XU manages rule engines, so the application server or application client uses the XU to connect to the rule engine. The XU retrieves and loads rulesets from persistence layer, passes data between the application and the rule engine, and manages the rule engines that are attached to loaded rulesets. The XU can create several rule engine instances. It also manages hot deployment. For hot deployment, the XU first receives notification from the JMX layer when new versions of rulesets are available, and then it responds.

The XU is a stand-alone deployable unit that you install on the application server. You use the XU RAR file, which contains the XU and the persistence layer, to install the XU.

## JMX management and execution model

- JMX API underlies Rule Execution Server architecture
  - MBeans model system administration functions
  - Access MBeans through Rule Execution Server console
- Management model
  - Provides access to runtime JMX MBeans for the Rule Execution Server model
  - Responsible for hot update and deployment
- Execution model
  - Provides access to runtime JMX MBeans for the XU to notify of changes and retrieve statistics
  - Exposes execution statistics as MBeans that can then be monitored by using JMX tools (like Tivoli)
  - XU instances run a local JMX MBean server



*Figure 13-11. JMX management and execution model*

The management and monitoring stack comprises:

- Management console
- JMX Management Model components

Rule Execution Server uses the Java Management Extension (JMX) API as its underlying architecture.

The JMX API uses Java objects that are called MBeans to model system administration functions. Each MBean contains a set of attributes that define parameters for various management functions and operations that define administrative actions.

The management and monitoring model components constitute an interface that depends on your application server and that manages business logic, including remote updating and browsing. They are based on JMX, a part of Java SE.

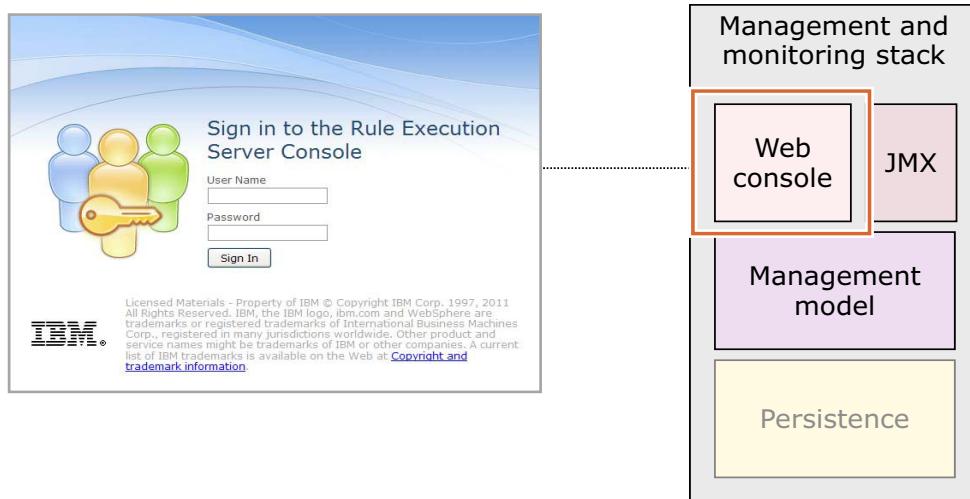
The JMX components are responsible for:

- Hot update and deployment. When a new version of a ruleset is deployed, the JMX layer informs all XUs in the cluster that a new version of the ruleset is available in the database.
- Displaying execution statistics as MBeans Execution statistics so they can then be monitored by using JMX tools like Tivoli.

From the Rule Execution Server console, you can access these MBean attributes and operations through a convenient graphical user interface.

## Management and monitoring stack: Console

- Web-based administration interface
- Application-specific interface to manage business logic, including remote browsing, updating, and deployment of RuleApps
- Central point of the Rule Execution Server architecture



Executing rules with Rule Execution Server

© Copyright IBM Corporation 2019

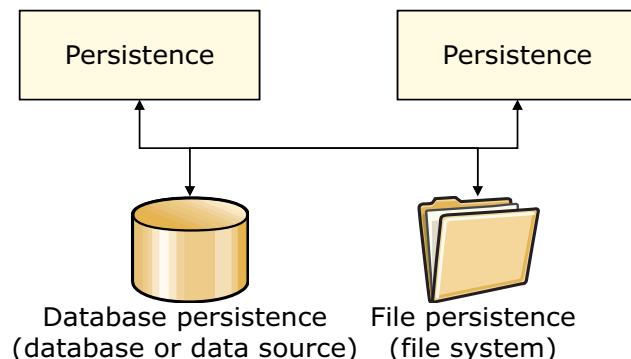
Figure 13-12. Management and monitoring stack: Console

Rule Execution Server is accessed through the Rule Execution Server console, which is a web-based administration interface. The Rule Execution Server console is a graphical user interface that depends on your application server. Use it to manage business logic, including remote browsing, updating, and deployment of RuleApps.

The Rule Execution Server console is the central point of the Rule Execution Server architecture. Many features of Rule Execution Server do not work without the Rule Execution Server console.

## Persistence layer

- Packaged in the management stack and the execution stack
  - Both the management and execution stacks can access the ruleset storage
  - When a new version of a ruleset is added to the management model, a notification of the update is sent through JMX to the XU
  - The XU receives the new resource
- The persistence layer provides a solution for file persistence (file system) or for database persistence (JDBC or data source)
- Possible values for persistence type:
  - `file` for a file persistence in Java SE
  - `datasource` for a database persistence in Java EE
  - `jdbc` for a database persistence in Java SE



[Executing rules with Rule Execution Server](#)

© Copyright IBM Corporation 2019

Figure 13-13. Persistence layer

The persistence layer is packaged in the management stack *and* in the execution stack so that ruleset storage is accessible from both stacks.



### Important

The ruleset persistence settings must be the same in the Execution Unit as in the Rule Execution Server console.

The persistence layer includes database persistence components that provide a solution for database persistence that is based on JDBC or a data source. Database persistence is the default solution, as it works in any architecture.

The persistence layer also includes file persistence components to provide a solution for a file persistence, by using the file system. Use this solution on the Java SE platform when you cannot set up a database. You might also use the file persistence solution when you do not want to set up a database, such as when you develop in Rule Designer.



### Attention

Use database persistence when you deploy to a Java EE cluster environment. If you use file persistence, you risk inconsistency. If, nevertheless, you choose to use file persistence with a clustered Rule Execution Server, you must make sure that all instances have access to a common network file system.

## Ant tasks

- Rule Execution Server provides a series of Management Ant tasks and Persistence Ant tasks for automation purposes
- Management Ant tasks
  - To automate RuleApp deployment and removal in Rule Execution Server
  - To automate Rule Execution Server configuration backup and restoration
- Persistence Ant tasks
  - To directly access the persistence layer to store or remove rulesets and RuleApps directly in the database, without passing through the Management stack (JMX)

Figure 13-14. Ant tasks

Rule Execution Server also supports Management Ant tasks and Persistence Ant tasks that you use for automation purposes. You can use Management Ant tasks to automate RuleApp deployment and removal in Rule Execution Server, and Rule Execution Server configuration backup and restoration.

You can use Persistence Ant tasks to access the persistence layer, and store or remove rulesets and RuleApps directly in the database without using the Management stack (that is, the JMX layer). Because of this direct access, if you use Persistence Ant tasks to deploy rulesets or RuleApps, you cannot have notification or hot deployment, and the Rule Execution Server Console is not aware of this deployment. As a consequence, use these Persistence Ant tasks only if you do not use a Rule Execution Server Console or do not care about hot deployment.

## 13.3. Managed execution in action

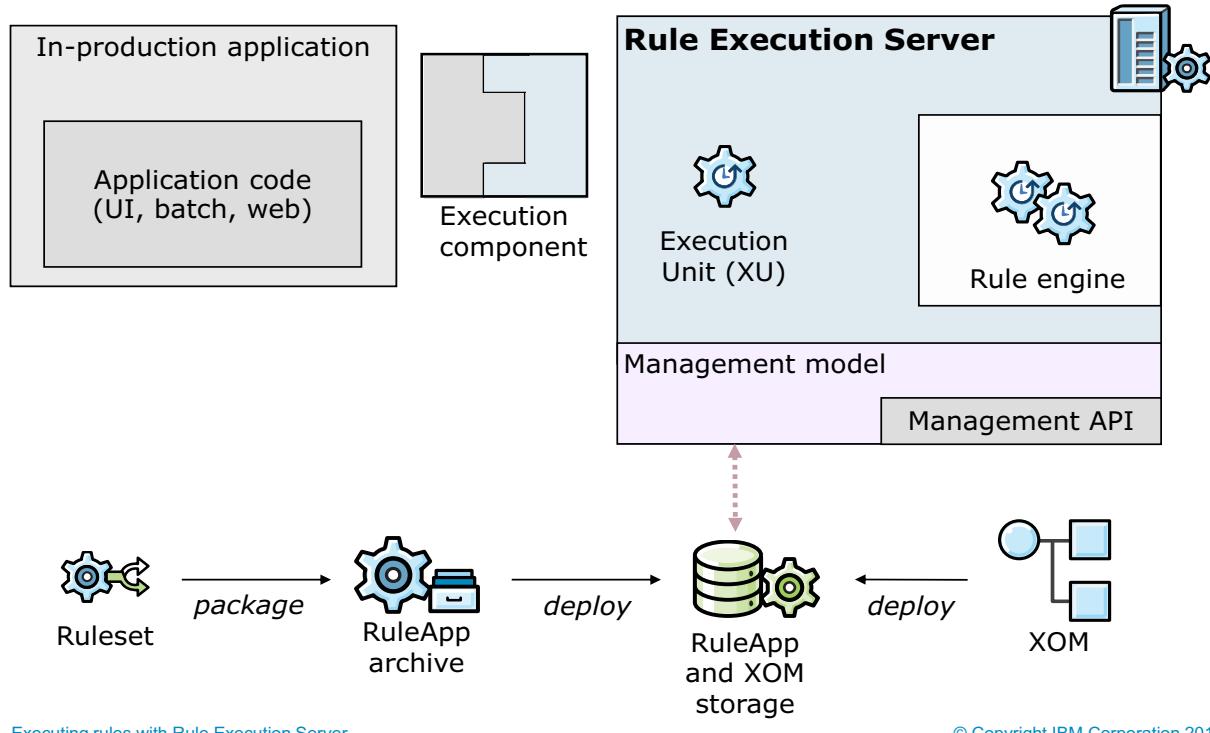
## Managed execution in action

Executing rules with Rule Execution Server

© Copyright IBM Corporation 2019

*Figure 13-15. Managed execution in action*

## Elements of a rule-based solution



Executing rules with Rule Execution Server

© Copyright IBM Corporation 2019

*Figure 13-16. Elements of a rule-based solution*

The various elements of your business rule solution (including the XOM, ruleset, RuleApp, rule engine, API, client application, and execution components of Rule Execution Server) are organized in your enterprise as shown here.

In the following sections, you learn how these elements dynamically interact in different use cases.

## Ruleset parsing

- When requested to execute a ruleset, the XU first retrieves the ruleset from the persistence layer according to the ruleset path, and then parses it
- Ruleset parsing can be *synchronous* or *asynchronous*
- Asynchronous parsing: Executes the ruleset in a timely manner
- Synchronous parsing: Forces the execution to wait for the latest deployed ruleset version
  - When you deploy a new ruleset version, all executions are suspended until that latest version is parsed

[Executing rules with Rule Execution Server](#)

© Copyright IBM Corporation 2019

Figure 13-17. Ruleset parsing

When the client application asks for a ruleset execution, the Execution Unit (XU) reads the ruleset from the persistence layer by using its ruleset path, loads it, and parses it.

By default, rulesets are parsed asynchronously, which means, if a ruleset is not yet parsed, it can still be executed based on its previous version if it exists in the cache.



### Note

Transparent decision services are parsed synchronously by default.

- 
- Keep the default asynchronous parsing when the timely execution of a ruleset is paramount and waiting for parsing of a new ruleset is not an option.
  - Change the default behavior when you must force ruleset executions to use the latest deployed version of the ruleset. With this action, all executions are suspended when a new version of the ruleset is deployed until that newer version is parsed. Therefore, requests to execute an updated ruleset are done only when the new ruleset is parsed and no execution request uses the old ruleset.

You can change the default behavior either by changing the XU deployment descriptor, or by using the API method `IlrSessionRequest.setForceUptodate`.

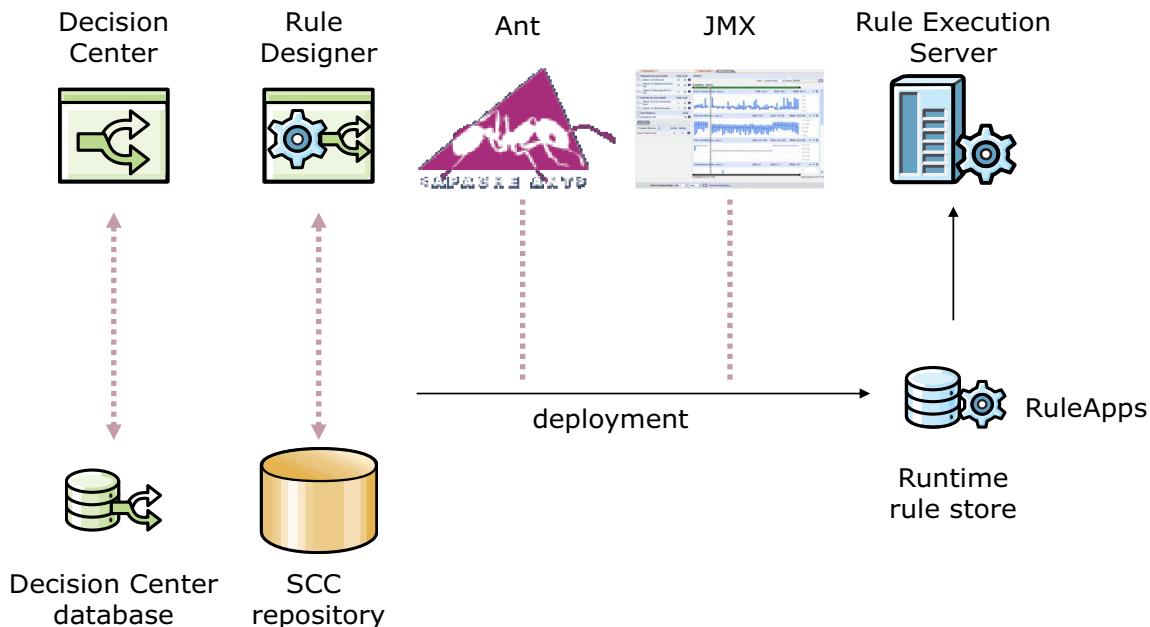
## Ruleset execution

- The XU retrieves a rule engine from the pool of available rule engines, and attaches it to the rule session
- The XU feeds the rule engine with the loaded rulesets, ruleset parameters, and objects in working memory
- The XU then requests the ruleset execution by the rule engine
- After ruleset execution, the XU:
  - Releases the rule engine that is attached to the rule session back to the rule engine pool when the rule session is stateless
  - Keeps the rule engine that is attached to the rule session when the rule session is stateful

Figure 13-18. Ruleset execution



## Ruleset update at run time



Executing rules with Rule Execution Server

© Copyright IBM Corporation 2019

Figure 13-19. Ruleset update at run time

When you deploy a new version of a ruleset that is packaged within a RuleApp, this new version of a ruleset is added to the management model. A notification of the update is sent through JMX to the XU, and the XU receives the new resource. With this mechanism, the new version of the ruleset is immediately available. You do not have to stop and restart the server or the client application.

This *hot deployment* mechanism, as you might expect, is the preferred deployment mechanism in both development and test environments.

Comparable deployment mechanisms exist from Rule Execution Server Console and from Decision Center. You can also trigger deployment by using Ant scripts or JMX tools.

## Runtime class loading

- By default, rulesets get their XOM from the client application class loader
- If the ruleset is associated with a managed Java XOM in Rule Execution Server:
  - The Execution Unit declares a new class loader to load the Java XOM resources for the ruleset (in the persistence layer)
  - The client class loader is used as the parent for the class loader for the managed XOM

Figure 13-20. Runtime class loading

The client class loader is used as the parent for the class loader for the managed XOM. As a consequence, any class with the same package and same class name that is present in both class loaders is taken into the client class loader first. The client class loader has precedence over the class loader for the managed XOM. You cannot reverse this behavior.

## 13.4. Platforms for Rule Execution Server

## Platforms for Rule Execution Server

Executing rules with Rule Execution Server

© Copyright IBM Corporation 2019

*Figure 13-21. Platforms for Rule Execution Server*

## Architecture questions

- Architects must select the deployment platform, which depends on the context of your enterprise
  - Java SE?
  - Java EE?
  - SOA?
  - How does rule execution fit into your software architecture?
- Environment determines:
  - How you integrate Rule Execution Server
  - How client applications request rule execution

Figure 13-22. Architecture questions

Rule Execution Server is based on a modular architecture to accommodate various possible enterprise environments. It is the role of the architects of your business rule application to select the deployment platform, which depends on the environment of your enterprise. It is the role of the administrators to deploy the Rule Execution Server components that are required on this deployment platform. Their decision pathway includes choices such as XML, Java SE or Java EE, EJB or non-EJB, synchronous or asynchronous execution, SOA or not. This choice affects the execution pattern on the client application side, and the API that you use.

## Possible choices: Introduction

| Rule Execution Server deployment                  | Approach to rules                       | Server for deployment                             | Guidelines for client application development   |
|---|---|---|---|
| Not deployed in a web or an application server    | (any)                                   | (not applicable)                                  | <ul style="list-style-type: none"> <li>• Base your application on Java</li> <li>• Use Rule Execution Server Java SE rule session API</li> </ul>   |
| Deployed in a web server or an application server | SOA: Rules are seen as services         | (any)   | <ul style="list-style-type: none"> <li>• Request execution of business rules as a service</li> <li>• Use hosted transparent decision service or monitored transparent decision service</li> </ul> |
|   | Not SOA: Rules are not seen as services | Supported application servers                     | <ul style="list-style-type: none"> <li>• Use Rule Execution Server Java EE API</li> <li>• Can be synchronous or not, local or remote, with or without EJB</li> </ul>                              |
|   |   | Non-supported application servers, or web servers | <ul style="list-style-type: none"> <li>• Use Rule Execution Server Java SE API</li> </ul>   |

Executing rules with Rule Execution Server

© Copyright IBM Corporation 2019

Figure 13-23. Possible choices: Introduction

This section is a guideline that developers, architects, and administrators can follow to determine which platform to use. It also gives you an overview of the various use cases that are supported in Operational Decision Manager.



### Important

This section is not exhaustive, and gives simplified guidelines for training. Some specific cases are not indicated here.

In a simple case, such as during tests, you do not want to deploy Rule Execution Server in a web server or in an application server. Base your application on Java, use Rule Execution Server Java SE Rule Session API, and package Rule Execution Server stacks within your client application. Then, execute the client application and Rule Execution Server stacks within the same JVM.

However, in many cases, you deploy Rule Execution Server either in a *web server* (for example, Apache HTTP Server) or in an *application server* (for example, WebSphere Application Server). In that case, your business rule application considers the deployed business rules either as “services”, if your approach is SOA-based, or not.

- If your business rule application is *SOA-based*, execute your business rules in your client application as *transparent decision services*. You learn more about HTDS later in this unit.
- If your business rule application is not SOA-based, and you do not use your rules as a service, you can base your business rule application on *Java*.
- To deploy Rule Execution Server on a supported application server (for example WebSphere Application Server), use the appropriate Rule Execution Server *Java EE API* (synchronous or not, local or remote, with or without EJB) in your client application.
- If you want to deploy on a non-supported application server or in a web server (for example, Apache HTTP Server), use Java SE instead. In that case, use the *Java SE Rule Session API* in your client application.

## Java SE

- Rule Execution Server can execute rulesets with 100% Java SE code
- It is used to:
  - Run rules from a web server like Tomcat
  - Run batches or run rules from a JMS provider or an enterprise service bus (ESB) that is not Java EE compliant
- When deployed in Java SE, your client application locally accesses the Rule Execution Server Execution components
- In this scenario, your client application uses the simple Java SE rule session API

Figure 13-24. Java SE

Rule Execution Server can execute rulesets with 100% Java SE code.

Many use cases for pure Java SE execution exist. For example, you can run rules from a web server, running batches, or run rules from a JMS provider or an enterprise service bus (ESB) that is based on Java EE.

When deployed in Java SE, your client application accesses the Rule Execution Server Execution components *locally*, that is, packaged as a simple library inside it.

In this case, your client application uses the simple *Java SE Rule Session API*.

You learn more about the required Rule Execution Server components to deploy in Java SE, the simple Java SE Rule Session API, and how to write your client application correspondingly, later in this unit.

## Java EE

- Use Java EE when your business rule application requires services such as transaction management, web containers, security
- In Java EE, your client application has different means to access the Rule Execution Server Execution components

| Required access  | API to use  | Purpose   |
|--|---|---|
| Synchronously and locally, with or without transaction control | Use a POJO rule session                             | For simple packaging and deployment, or a fine grained transaction control, or to use rules outside the EJB container |
| Synchronously with transaction control (locally or remotely)   | Use an EJB rule session (local or remote interface) | For remote client access capabilities, and support for declarative transaction and security descriptors               |
| Asynchronously   | Use message-driven beans (MDB)                      | Use message-driven beans (MDB)  |

Figure 13-25. Java EE

The Java EE platform is based on the Java SE specification. Java EE simplifies enterprise applications by defining and specifying a complete set of common standard services, such as naming, transaction management, concurrency, security, and database access. Java EE also defines a container model, which houses and manages instances of Java EE application components. Containers are in turn housed within Java EE servers.

If your business rule application requires services such as transaction management, web containers, and security, then change to a Java EE application server, rather than add the necessary Java extensions to the Java SE platform. When deployed in Java EE, your client application can access the Rule Execution Server Execution components either synchronously (locally or remotely), or asynchronously:

- Synchronously and locally, with or without transaction control. In both cases, your client application uses a *POJO rule session*. Use a POJO rule session when:
  - You require a simple packaging and deployment or a fine-grained transaction control.
  - You want to use your rules outside the EJB container (in order not to incur the EJB container cost).
- Synchronously with transaction control (locally or remotely). In both cases, your client application uses an *EJB rule session* (local or remote interface). You use an EJB rule session

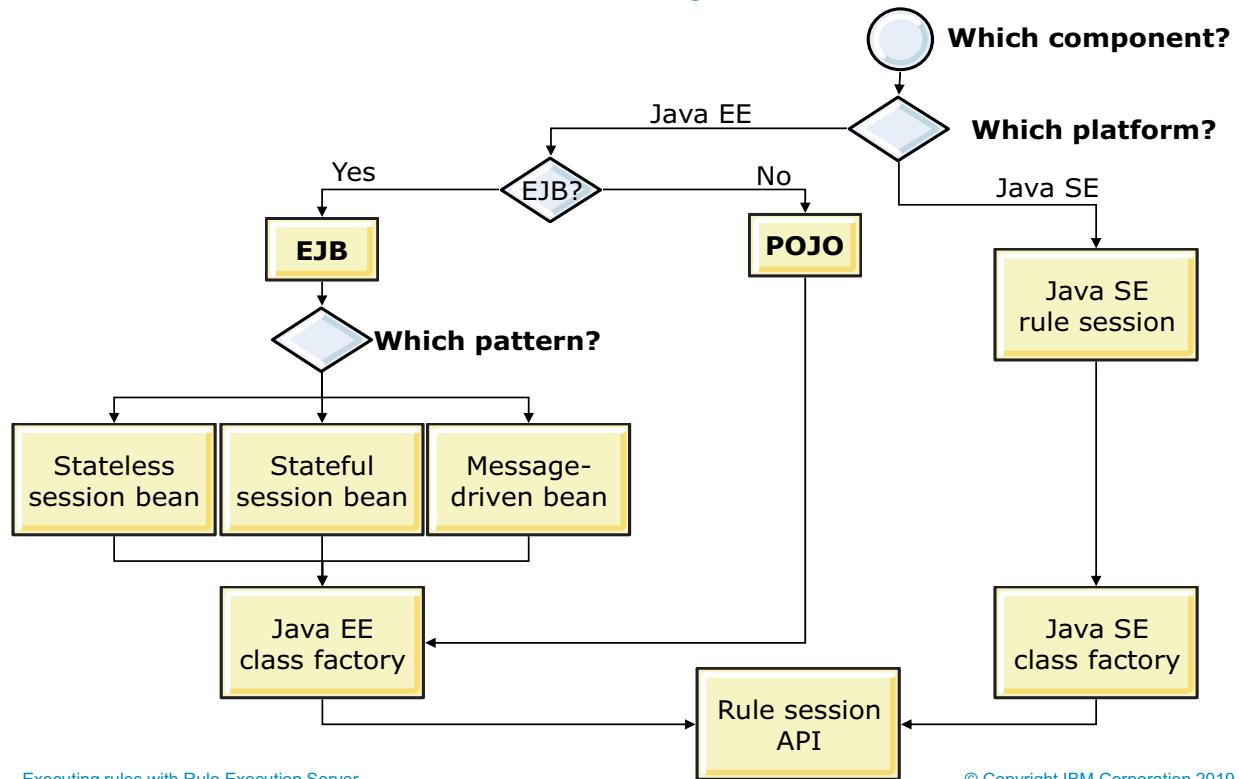
when you require remote client access capabilities, and support for declarative transaction and security descriptors.

- Asynchronously. With POJO or EJB rule sessions, you send and receive Java Message Service (JMS) messages *synchronously*. To receive messages *asynchronously*, use *message-driven* beans (MDB). MDBs provide the scalability to execute rulesets when high latency or high peak load is expected. You learn more about MDBs later in this unit.

Later in this unit, you learn more about the Rule Execution Server components that are required to deploy in Java EE, the POJO rule session, the EJB rule session, and MDBs.



## Java SE and Java EE: A summary



Executing rules with Rule Execution Server

© Copyright IBM Corporation 2019

Figure 13-26. Java SE and Java EE: A summary

The possible choices with the Java Platform (Java SE or Java EE) that were presented in the previous slides are summarized here.

## Transparent decision services

- You can expose ruleset execution as a service
  - A web service with management capabilities
- Transparent:
  - Users do not have to know how it is implemented
  - Users must know only how to access it through HTTP with the XML or JSON formats
- Hosted transparent decision service (HTDS):
  - Use when you want to create a web service with minimal configuration

*Figure 13-27. Transparent decision services*

Operational Decision Manager provides ways to automatically expose ruleset execution as a transparent decision service. A transparent decision service is technically a web service with management capabilities. It drives rule execution, and enables users to access Rule Execution Server through a web service, rather than accessing it directly. It is said to be *transparent* because users do not have to know how it is implemented, just how to access it through HTTP with XML or JSON formats.

The mechanism to create hosted transparent decision services (HTDS) requires only limited configuration. If you want a rapid deployment of a web service that is based on your rules, use that mechanism and create an HTDS. You can use HTDS with both Java XOMs and dynamic XOMs.

By default, when you deploy a RuleApp with a Java XOM, the Java XOM is deployed as a “managed XOM” as well. In that condition, you do not have to embed the Java XOM in the WAR file that is embedded in the HTDS EAR file. If you do not deploy the Java XOM as a managed XOM, then you must package the Java XOM as part of the WAR file in the HTDS EAR file.

Similarly, if you are using a dynamic (XML-based) XOM, the XSD files that define the XML model are packaged within the ruleset by default (which is the default configuration in Rule Designer). If you modify this default behavior, you must package the XSD files that define the dynamic domain in the EAR file as well.

You learn more about transparent decision services later in this unit.

## 13.5. Introducing the Rule Execution Server API

## Introducing the Rule Execution Server API

Executing rules with Rule Execution Server

© Copyright IBM Corporation 2019

*Figure 13-28. Introducing the Rule Execution Server API*

## Introduction

- For environments that are based on Java, the Rule Execution Server does not provide a ready-to-use client application
- To create a Java client application that requests the managed execution of business rules with Rule Execution Server, use the Rule Execution Server API to command the XU

Figure 13-29. Introduction

For Java environments, Rule Execution Server does not provide a ready-to-use client application. You must write the client application code to execute your ruleset, by using the *Rule Execution Server API* to command the Execution Unit (XU). This API is provided to bind an application and call the execution module.

Which part of the API to use depends on the execution pattern that your client application follows, which, in turn, depends on how Rule Execution Server is deployed in your enterprise. In this topic, you discover first the *execution patterns*, and then the API that these patterns are based on.

## Execution patterns: Synchronous

In Java SE:

- Local access
  - Java SE rule session

In a Java EE container:

- Local access with or without transaction control
  - Plain old Java objects (POJO) rule session
- Local access with transaction control at the rule session level
  - Local interface of a stateful or a stateless EJB rule session
- Remote access with transaction control at the rule session level
  - Remote interface of a stateful or a stateless EJB rule session

*Figure 13-30. Execution patterns: Synchronous*

Several execution patterns that are based on Java exist with Rule Execution Server to execute your rules. With these patterns, you can use either Rule Execution Server *rule sessions* or *message-driven beans* (MDB). When you create your client application, the first step is to select which execution pattern you want. This choice affects the way that you implement your client application and the way Rule Execution Server components are deployed. The possible execution patterns, and supporting Rule Execution Server API, include:

- Synchronous local access, in a Java SE (outside a Java EE container): the *Java SE rule session*
- Synchronous local access, with or without transaction control, inside a Java EE container: the *plain old Java objects (POJO) rule session*
- Synchronous local access with transaction control at the rule session level, inside a Java EE container: the local interface of a stateful or a stateless *EJB rule session*
- Synchronous remote access with transaction control at the rule session level, inside a Java EE container: the remote interface of a stateful or a stateless *EJB rule session*

## Execution patterns: Asynchronous

- Asynchronous access, remote, or local
  - Message-driven rule bean (MDB)
  - Stateless asynchronous execution (in Java SE and Java EE POJO modes only)

Figure 13-31. Execution patterns: Asynchronous

You can do asynchronous access, either remote or local, with either:

- A message-driven rule bean (MDB)
- A stateless asynchronous execution, in Java SE and in Java EE POJO modes only

## Rule session API overview (1 of 2)

- When using rule sessions to request ruleset execution within the managed environment of Rule Execution Server, your client application completes steps similar to the next ones:
  1. Get a rule session factory
  2. Create a rule session from the rule session factory
  3. Create a rule session request from the rule session
  4. Configure the rule session request
  5. Execute the rule session request from the rule session
  6. Retrieve the rule session response, and analyze it

Figure 13-32. Rule session API overview (1 of 2)

## Rule session API overview (2 of 2)

- The exact classes that your client application requires to implement this series of steps depend on the chosen execution pattern
- These classes all rely on similar Java interfaces
  - See next slides

Figure 13-33. Rule session API overview (2 of 2)

The exact classes that you use depend on the chosen execution pattern. However, these classes all rely on similar Java interfaces, as explained next.

## Rule session factories

- The rule session factory is the entry point for calling the services of Rule Execution Server with the rule session API
- All the rule session factory classes implement the `ilog.rules.res.session.IlrSessionFactory` interface
- With the `IlrSessionFactory`, you can:
  - Create stateful sessions, stateless sessions, and management sessions
  - Create execution requests
  - Enable interceptors

*Figure 13-34. Rule session factories*

The rule session factory is the entry point for calling the services of Rule Execution Server with the rule session API. Your client application must have a rule session factory to create a rule session. All the rule session factory classes implement the `ilog.rules.res.session.IlrSessionFactory` interface. The methods of the `IlrSessionFactory` interface enable your client application to create stateful, stateless, and management sessions, to ask for rule session request execution, and to enable interceptors. All rule session factory objects implement the `IlrSessionFactory` interface, and provide your client application with a uniform API.

Because of this uniformity, the migration path from a pure Java SE environment to a full Java EE environment is simplified. Therefore, you can develop your applications in Java SE, and move to Java EE with relatively few code changes.

You explore the `IlrSessionFactory` interface more practically, when you create a client application with Rule Execution Server in Java SE, later in this unit.

## Rule sessions

- A rule session
  - Is a runtime connection between a client and a rule engine
  - Provides a mechanism to access the list of all the rulesets that are registered with the factory
  - Defines the type of session that the client establishes
  - Manages a class loader to enable the XU to execute a ruleset on any XOM
  - Provides a service to handle XML parameters

Figure 13-35. Rule sessions

A rule session is a runtime connection between a client and a rule engine. Rule sessions provide a mechanism to access the list of all the rulesets that are registered with the provider. They define the type of the session that a client establishes. They manage a class loader to enable the Execution Unit (XU) to execute a ruleset on any XOM.

## Stateless rule sessions

- A stateless rule session handles no state
- With a stateless rule session, you can:
  - Set input parameters
  - Get output parameters
  - Not change the state of the objects in working memory
- You can also use stateless rule session to execute rulesets asynchronously
- Stateless rule sessions are instances of classes that implement the `ilog.rules.res.session.IlrStatelessSession` interface

Figure 13-36. Stateless rule sessions

A stateless rule session handles no state. Between two method calls, it does not maintain any data or engine information. You can set input parameters and get output parameters, but you cannot access the working memory. Because of this limitation, a stateless rule session has higher performance than a stateful rule session. A stateless rule session is an instance of a class that implements the `ilog.rules.res.session.IlrStatelessSession` interface.

## Stateful rule sessions

- A stateful rule session is linked to a single ruleset path and authorizes access to the working memory
- With a stateful rule session, you can:
  - Set input parameters
  - Get output parameters
  - Change the state of the objects in working memory
- Use stateful rule sessions when your application must insert and update objects in the working memory or retract objects from it
- Stateful rule sessions are instances of classes that implement the `ilog.rules.res.session.IlrStatefulSession` interface

Figure 13-37. Stateful rule sessions

A stateful rule session allows your application to set input parameters, get output parameters, and change the state of the objects in working memory by using the API. You use a stateful rule session when your application must insert and update objects in the working memory or retract objects from it. A stateful rule session is an instance of a class that implements the `ilog.rules.res.session.IlrStatefulSession` interface.

## Rule session requests

- From the rule session, create a rule session request, configure it, and have it executed
- A rule session request is an instance of a class that implements the `ilog.rules.res.session.IlrSessionRequest` interface
- With a rule session request, you can do the following important operations:
  - Define the ruleset path
  - Set and get the ruleset parameters required for the execution
  - Define the traces to generate during the execution
  - Manage the Decision ID

*Figure 13-38. Rule session requests*

From the rule session, the client application can create a *rule session request*, configure it, and have it executed. A rule session request is an instance of a class that implements the `ilog.rules.res.session.IlrSessionRequest` interface. By using a rule session request, your client application can do the following important operations, but is not limited to these operations:

- Define the ruleset path with `setRulesetPath` and `getRulesetPath`.
- Set and get the parameters required for the execution, for example with `getInputParameters` and `setInputParameters`.
- Define the traces to generate during the execution.
- Manage *Decision IDs*, with `getExecutionID` and `setExecutionID`.

A Decision ID is a `String` object that identifies the results of a ruleset execution. If you do not define it, a default Decision ID is automatically generated, equal to the ID of the Execution Unit (XU) connection. You can override this default Decision ID by using the `IlrSessionRequest.setExecutionID` method. You can use Decision IDs to associate rule session requests with rule session responses. You can also use Decision IDs to audit rules.

## Rule session requests: Decision ID

- A Decision ID is a `String` object that identifies the results of a ruleset execution
- If you do not define it, a default Decision ID is automatically generated equal to the ID of the XU connection
- You can override this default Decision ID in the rule session request
- You can use the Decision ID to:
  - Associate rule session requests with rule session responses
  - Audit rules, for example, to filter out among multiple ruleset executions

Figure 13-39. Rule session requests: Decision ID

## Rule session responses

- After the execution of the rule session request, retrieve the rule session response
- A rule session response is an instance of a class that implements the `ilog.rules.res.session.IlrSessionResponse` interface
- With a rule session response, you can do the following important operations:
  - Retrieve the output ruleset parameters
  - Retrieve the Decision ID
  - Get traces generated during the execution

Figure 13-40. Rule session responses

After executing the rule session request by using the appropriate API of the rule session, the client application retrieves the *rule session response*. A rule session response is an instance of a class that implements the `ilog.rules.res.session.IlrSessionResponse` interface. By using a rule session response, your client application can do the following important operations, but is not limited to these operations:

- Retrieve the output parameters, with `getOutputParameters`
- Retrieve the Decision ID, with `getExecutionID`
- Get traces generated during the execution

## Ruleset path for execution (1 of 2)

- Ruleset path uniquely identifies a ruleset within a RuleApp
- In client application, use the ruleset path to indicate which ruleset to execute
- **Reminder:** The ruleset path in the application must match the ruleset name and RuleApp name as they are *known in Rule Execution Server*
- The ruleset path can be canonical or non-canonical

Figure 13-41. Ruleset path for execution (1 of 2)

You already learned what the ruleset path is when you learned how to create a RuleApp. At the deployment stage, you learned that the ruleset path is a way to uniquely identify a ruleset within a RuleApp.



### Important

When the RuleApp is deployed to Rule Execution Server, its name might differ from the name in the Rule Designer project or in Decision Center. Any character that is not authorized is removed. The same modifications might apply to the name of the deployed ruleset. The ruleset path for execution is based on the RuleApp name and the ruleset name as known in Rule Execution Server, and must contain only authorized characters.

## Ruleset path for execution (2 of 2)

- Canonical
  - Specify both the ruleset version and the RuleApp version
  - Example:  
RuleApp-name/RuleApp-version/ruleset-name/ruleset-version
  - Use when you must execute a specific version
  
- Non-canonical (preferred)
  - Version is not specified
  - Example:  
RuleApp-name/ruleset-name  
RuleApp-name/RuleApp-version/ruleset-name  
RuleApp-name/ruleset-name/ruleset-version
  - Rule Execution Server chooses appropriate ruleset to execute by selecting latest activated version of the ruleset or RuleApp, depending on which versions are not specified

Figure 13-42. Ruleset path for execution (2 of 2)

When deployed, you can identify rulesets within RuleApps by using the *canonical ruleset path*, where the versions for the ruleset and for the RuleApp are indicated. At the execution stage, the ruleset path plays the same identification role: your client application uses the ruleset path to indicate which ruleset to execute in the rule session request.

However, in your client application, you can write a *non-canonical ruleset path*, where you omit the version of the ruleset, the version of the RuleApp, or both versions, such as in the following examples:

```
RuleApp-name/RuleApp-version/ruleset-name
RuleApp-name/ruleset-name/ruleset-version
RuleApp-name/ruleset-name
```

If your application specifies a non-canonical ruleset path, Rule Execution Server selects for execution the latest activated version of the ruleset, or of the RuleApp, depending on which versions you did not specify.

In most cases, you use a non-canonical ruleset path in your client application, which enables Rule Execution Server choose the appropriate ruleset to execute.

You use the canonical (fully specified) ruleset path only if you want this specific version of the ruleset to execute.

## Traces

- When you execute your business rule application, your application generates different types of traces:
  - *Log traces*: Traces that are generated in the log file
  - *Output traces*: Traces that business rules print in the standard output file
  - *Decision traces*: Data that the rule engine generates to trace how the ruleset executes
- You cannot programmatically retrieve log traces
- You can programmatically retrieve output traces and decision traces

*Figure 13-43. Traces*

When you execute your business rule application, your application generates different types of traces:

- *Log traces*

Ruleset log traces are generated in the log file, for example, by executing instructions like `System.out.println` in your client application.

You cannot retrieve the log traces programmatically.

- *Output traces*

Ruleset output traces are generated when actions of business rules are executed that print text in the standard output file. For example, an action of a business rule can print text in the standard output file by executing the BAL keyword `print`.

You can retrieve the ruleset output traces programmatically, by using `IIRSessionResponse.getRulesetExecutionOutput`. This call returns a `String`.

- *Decision traces*

Ruleset decision traces are data that the rule engine generates to describe how the ruleset executes.

With decision traces, your client application can retrieve the duration of the ruleset execution, the number of rules that are fired or not fired, and other information.

You can retrieve the ruleset output traces programmatically, as you see in the following slide.

## Decision traces: How to

- Enable ruleset decision traces:  
`IlrSessionRequest.setTraceEnabled(true)`
- Get the trace filter, and use it to set the traces to filter:  
`IlrSessionRequest.getTraceFilter`
- Get the decision traces after the ruleset execution:  
`IlrSessionResponse.getRulesetExecutionTrace`, which returns an `IlrExecutionTrace` object
- Get the decision trace data from the `IlrExecutionTrace` object

*Figure 13-44. Decision traces: How to*

To obtain ruleset decision trace, your client application must use the rule session request and the rule session response as follows:

1. Enable ruleset decision traces, by calling the `IlrSessionRequest.setTraceEnabled(true)` method.
2. Filter the required ruleset decision traces by using the appropriate methods of the object that `IlrSessionRequest.getTraceFilter` returns, for example, `setInfoExecutionDuration(true)`.
3. Get the decision traces after the ruleset execution by calling the `IlrSessionResponse.getRulesetExecutionTrace` method, which returns an instance of the `IlrExecutionTrace` class.



### Important

If the decision traces are not enabled, the call to `IlrSessionResponse.getRulesetExecutionTrace` returns a `null` value, and no decision trace data is available.

4. Get the multiple decision trace data by calling the appropriate methods of the `IIRExecutionTrace` object, such as:

```
getExecutionDuration  
getTotalRulesFired  
getTotalRulesNotFired  
getExecutionEvents  
getTotalTasksExecuted  
getRulesNotFired
```

## Decision traces: Complement

- To get all decision traces for rules that were executed with an execution mode other than RetePlus:
  - Define the `ruleset.sequential.trace.enabled` property in your ruleset
  - Set the value of the `ruleset.sequential.trace.enabled` property to `true`

Figure 13-45. Decision traces: Complement

If your ruleset contains rule tasks that are executed with an execution mode *other than RetePlus*, you must also define the `ruleset.sequential.trace.enabled` property in your ruleset. Set its value to `true` when you want to have all decision traces about the list of rules.

Despite similar names, do not confuse this `ruleset.sequential.trace.enabled` property, which affects ruleset decision traces *at the client level*, with the `ruleset.trace.enabled` property, which asks to dump traces *at the server level*. When the `ruleset.trace.enabled` property is defined and set to `true`, traces are generated at the server level and logged in the server log file, in a way similar to *Log traces* for the client application. Such server log traces cannot be retrieved programmatically.

## 13.6. Executing rules in Java SE

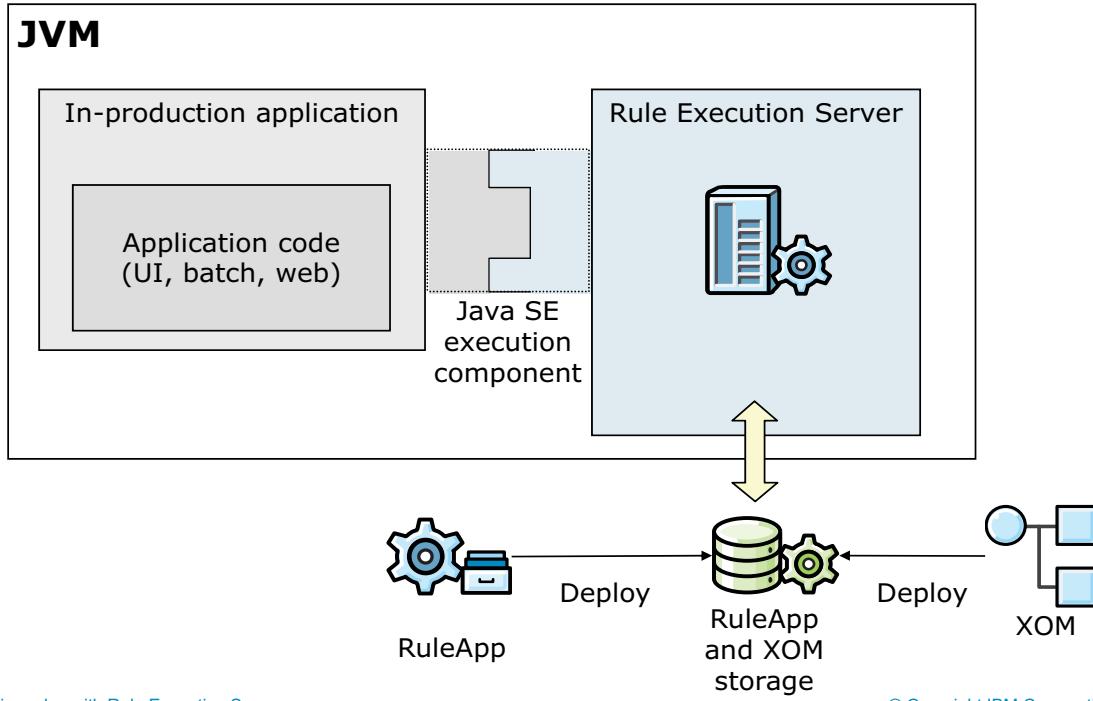
## Executing rules in Java SE

Executing rules with Rule Execution Server

© Copyright IBM Corporation 2019

Figure 13-46. Executing rules in Java SE

## Rule execution in Java SE



Executing rules with Rule Execution Server

© Copyright IBM Corporation 2019

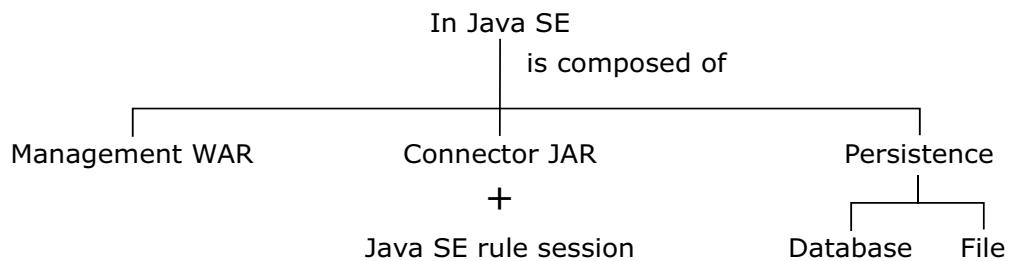
*Figure 13-47. Rule execution in Java SE*

When your client application asks for the managed execution of your ruleset with Rule Execution Server that is deployed in Java SE, the application and Rule Execution Server both run on the same JVM.

For web-based applications, keeping the rule engine in the same JVM as the servlet container helps minimize the object serialization between the JVM hosting the servlet container and the JVM hosting the rule engine. However, because JCA specification is not implemented in a Java SE environment, the application server does not manage the pool of rule engines of the XU. This pool is a fixed-size list from where a new rule engine is found for each new request.

## Deployed Rule Execution Server artifacts in Java SE

- Package your Java XOM into your application and deploy it along with the application
  - At run time, your application class loader makes the XOM objects available to the rule execution environment



*Figure 13-48. Deployed Rule Execution Server artifacts in Java SE*

In Java SE:

- The Rule Execution Server Console is inside a web server, which is packaged as a management WAR.
- The XU JCA connector is packaged as a .jar file (library).
- The Rule Execution Server persistence layer can be either file-based or database-based.
- The client application uses the Java SE rule session API.

For the rule engine to access the Java XOM, package your Java XOM into your application and deploy it along with the application. At run time, your application class loader makes the XOM objects available to the rule execution environment, which differs for a dynamic XOM.

## Required Rule Execution Server API in Java SE

- The client application uses the Java SE rule session API
- Create a simple Java SE rule session, by using an instance of the `ilog.rules.res.session.IlrJ2SESessionFactory` class
- With this factory, create either a stateless or a stateful Java SE rule session
- You can also create an `IlrManagementSession` to manipulate the repository for RuleApps and rulesets

*Figure 13-49. Required Rule Execution Server API in Java SE*

To handle Java SE requests between the application and Rule Execution Server, your client application creates a simple Java SE rule session by using an instance of the `ilog.rules.res.session.IlrJ2SESessionFactory` class.

- The `IlrJ2SESessionFactory` class is the class that implements the `IlrSessionFactory` interface that is dedicated to the creation of Java SE rule sessions.
- With this factory, you create either a stateless or a stateful Java SE rule session.

You can also create an `IlrManagementSession`. With such a *management session*, you can manipulate the repository for RuleApps and rulesets, and retrieve metadata. You can create a client application to execute your ruleset in the Java SE environment by starting from empty classes. However, Rule Designer also provides a wizard to create the base Java classes of a Java SE-based client application that you then must complete.

## Example with a stateless rule session in Java SE

- In this example, the ruleset has one IN OUT ruleset parameter that is called `report`, of class `Report`

```
IlrSessionFactory factory = new IlrJ2SESessionFactory();
IlrStatelessSession session =
    factory.createStatelessSession();
IlrSessionRequest sessionRequest = factory.createRequest();
sessionRequest.setRulesetPath("/RuleAppName/rulesetName");
sessionRequest.setTraceEnabled(true);
sessionRequest.getTraceFilter().setInfoAllFilters(true);
Map inputParameters = new HashMap ();
Report in_report = new Report(); // no-arg constructor
inputParameters.put("report", in_report);
sessionRequest.setInputParameters(inputParameters);
IlrSessionResponse sessionResponse =
    session.execute(sessionRequest);
Report out_report =
    (Report)sessionResponse.getOutputParameters().get("report");
```

Figure 13-50. Example with a stateless rule session in Java SE

This example shows a possible skeleton for a client application that uses a stateless rule session in Java SE.

In this example, the ruleset has one IN OUT ruleset parameter that is called `report`, of class `Report`.

- Get a *rule session factory*:

```
IlrSessionFactory factory = new IlrJ2SESessionFactory();
```

- Create a *rule session* from the rule session factory:

```
IlrStatelessSession session = factory.createStatelessSession();
```

- Create a *rule session request* from the rule session:

```
IlrSessionRequest sessionRequest = factory.createRequest();
```

4. Configure the rule session request; in particular, set the initial value of the report ruleset parameter (IN):

```
sessionRequest.setRulesetPath(...);  
sessionRequest.setTraceEnabled(true);  
sessionRequest.getTraceFilter().  
setInfoAllFilters(true);  
Map inputParameters = new HashMap();  
Report in_report = new Report(...);  
inputParameters.put("report", in_report);  
sessionRequest.setInputParameters(inputParameters);
```

5. Execute the rule session request from the rule session and retrieve the *rule session responses*:

```
IlrSessionResponse sessionResponse = session.execute(sessionRequest);
```

6. Analyze the *rule session responses*; in particular, get the modified value of the report ruleset parameter (OUT):

```
Report out_report = (Report)  
sessionResponse.getOutputParameters().get("report");
```

## 13.7. Executing rules in Java EE

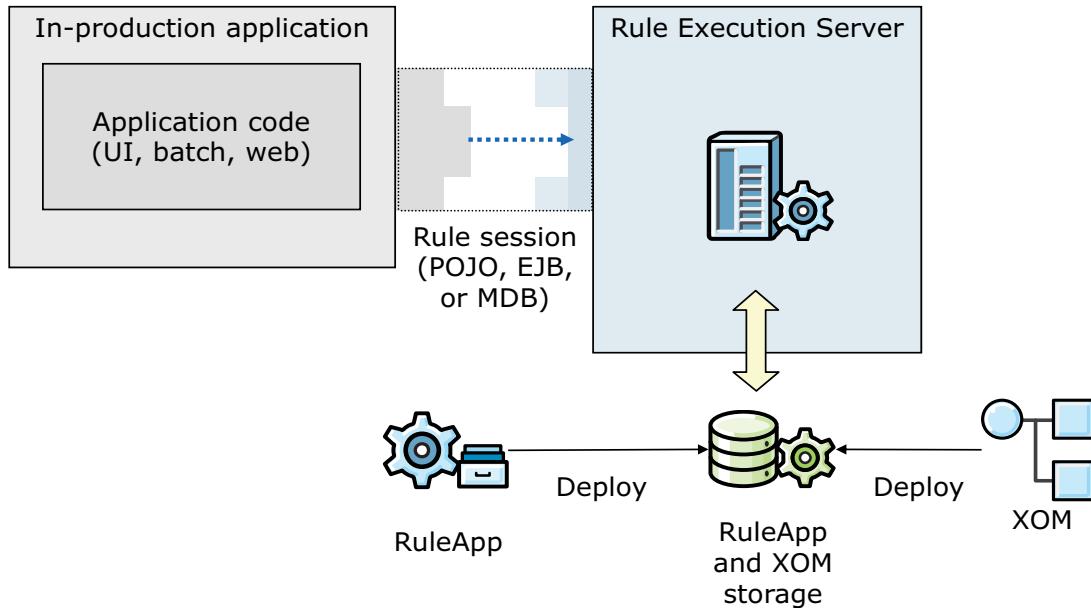
## Executing rules in Java EE

Executing rules with Rule Execution Server

© Copyright IBM Corporation 2019

*Figure 13-51. Executing rules in Java EE*

## Rule execution in Java EE



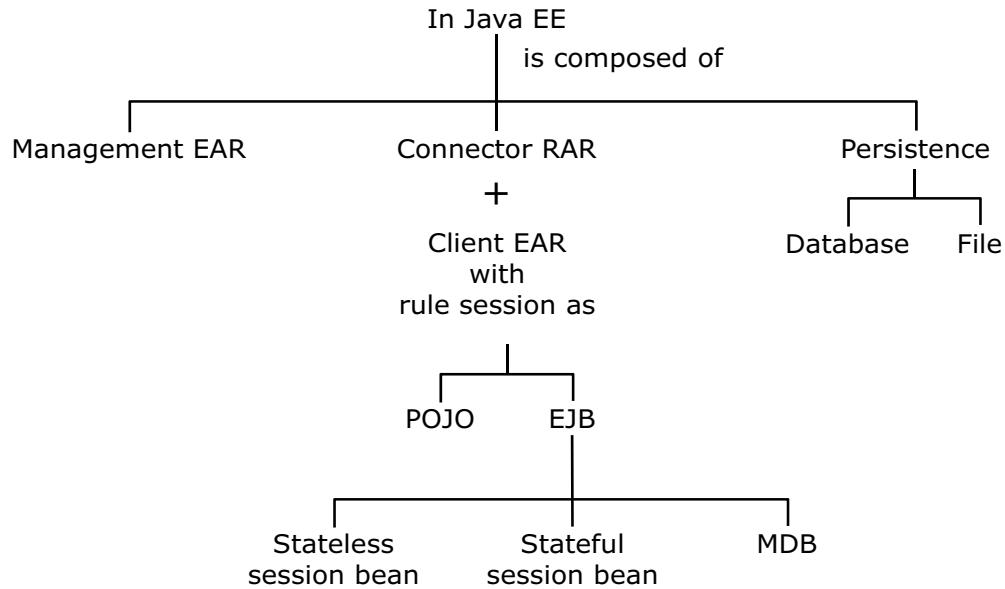
Executing rules with Rule Execution Server

© Copyright IBM Corporation 2019

Figure 13-52. Rule execution in Java EE

When your client application asks for the managed execution of your ruleset with Rule Execution Server that is deployed in Java EE, the application and Rule Execution Server might run on different JVMs.

## Deployed Rule Execution Server artifacts in Java EE



Executing rules with Rule Execution Server

© Copyright IBM Corporation 2019

Figure 13-53. Deployed Rule Execution Server artifacts in Java EE

In Java EE:

- The Rule Execution Server Console is packaged as a management EAR file for application servers.
- The connected XU JCA is packaged as a RAR (JCA resource archive), not as a .jar file (library) as in Java SE.
- The Rule Execution Server persistence layer is deployed in the same way as for Java SE.
- The client application uses the Java EE rule session API (POJO, EJB, MDB).

## Required API in Java EE

- The client application uses:
  - POJO rule session API
  - EJB rule session API (stateless or stateful)
  - MDB

Figure 13-54. Required API in Java EE

To handle Java EE requests between the application and Rule Execution Server, your client application must use a POJO rule session, an EJB rule session, or a message-driven bean.

## Java EE POJO rule session

- Create a POJO rule session by using an instance of the `ilog.rules.res.session.IlrPOJOSessionFactory` class
  - The `IlrPOJOSessionFactory` class implements the `IlrSessionFactory` interface that is dedicated to the creation of POJO rule sessions
- With this factory, create either a stateless or a stateful POJO rule session
  - You can also create management sessions (`IlrManagementSession`)

Figure 13-55. Java EE POJO rule session

You create a POJO rule session by using an instance of the `ilog.rules.res.session.IlrPOJOSessionFactory` class. The `IlrPOJOSessionFactory` class is the class that implements the `IlrSessionFactory` interface that is dedicated to the creation of Java EE POJO rule sessions. With this factory, you can create stateless or stateful POJO rule sessions and management sessions.

## Java EE EJB rule session

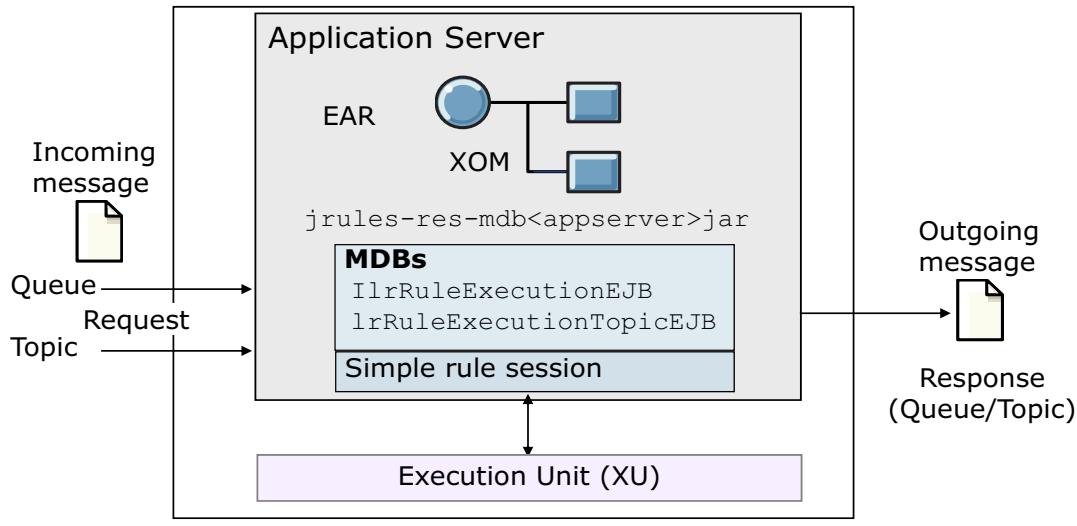
- Create an EJB rule session by using an instance of the `ilog.rules.res.session.IlrEJB3SessionFactory` class
  - The `IlrEJB3SessionFactory` class implements the `IlrSessionFactory` interface that is dedicated to the creation of EJB rule sessions
- With this factory, create either a stateless or a stateful EJB rule session
  - You can also create management sessions (`IlrManagementSession`)

Figure 13-56. Java EE EJB rule session

You create an EJB rule session by using an instance of the `ilog.rules.res.session.IlrEJB3SessionFactory` class. The `IlrEJB3SessionFactory` class is the class that implements the `IlrSessionFactory` interface that is dedicated to the creation of Java EE EJB rule sessions. With this factory, you can create stateless or stateful EJB rule sessions and management sessions.

## Java EE message-driven bean (MDB)

- An enterprise bean that allows Java EE applications to process messages asynchronously
- An instance of an MDB calls the XU when a JMS message arrives and posts the results of the rule engine processing to a JMS destination
  - The real invocation of the rule engine is delegated to a simple rule session



Executing rules with Rule Execution Server

© Copyright IBM Corporation 2019

Figure 13-57. Java EE message-driven bean (MDB)

A message-driven rule bean, or MDB, is an enterprise bean that allows Java EE applications to process messages asynchronously. Unlike rule sessions, clients do not access message-driven rule beans through interfaces. An instance of a message-driven rule bean calls the Execution Unit (XU) when a Java Message Service (JMS) message arrives and posts the results of the rule engine processing to a JMS destination. The call of the rule engine is delegated to a simple rule session. Rule execution JMS messages contain a ruleset path and parameters. The message-driven rule beans take charge of receiving messages and calling a simple rule session. The stateless rule session takes charge of sending response messages, if any.

From a requester point of view, asynchronous messaging means that one process or thread sends a message to a destination and expects no reply. From a consumer point of view, asynchronous means that a message is received and a reply is not sent immediately to the sender. The sender can rely on features such as guaranteed delivery to make sure that the message arrives.

## 13.8. Executing rules as transparent decision services

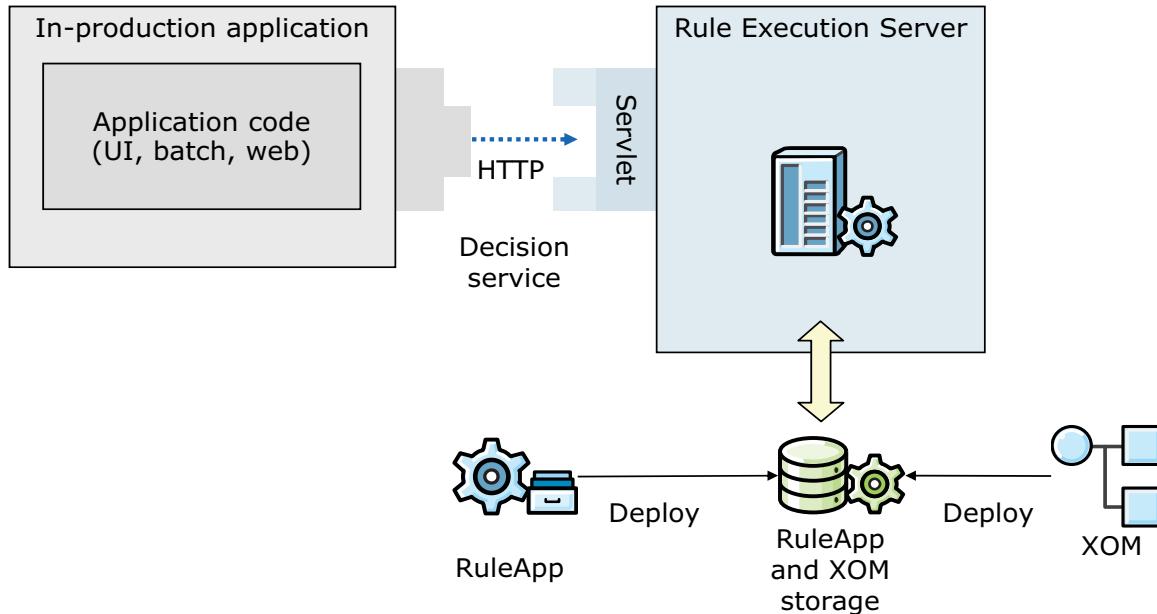
## Executing rules as transparent decision services

Executing rules with Rule Execution Server

© Copyright IBM Corporation 2019

*Figure 13-58. Executing rules as transparent decision services*

## Rules as transparent decision services



[Executing rules with Rule Execution Server](#)

© Copyright IBM Corporation 2019

Figure 13-59. Rules as transparent decision services

By using hosted transparent decision services, you can access Rule Execution Server through a web service.

A transparent decision service helps development teams deploy business rules as fully formed web services and weave them into service-oriented architecture (SOA) platforms.

To execute rules as decision services, the application is installed remotely. Requests are made to Rule Execution Server through HTTP by using transparent decision services.

An HTDS is essentially a ruleset that is deployed as a web service, which is installed on the same application server as Rule Execution Server.

## Hosted transparent decision service

- Is essentially a ruleset that is deployed as a web service, which is installed on the same application server as Rule Execution Server
- Is an execution component linked to a ruleset path with a JMX management bean (MBean)
- Rule Execution Server automatically exposes deployed rulesets as a web (decision) service
- Hosted transparent decision services can run synchronously or asynchronously
- To create an HTDS:
  - Install Rule Execution Server on your web or application server
  - Deploy the HTDS ruleset archive to the same server

Figure 13-60. Hosted transparent decision service

A hosted transparent decision service is an execution component that is linked to a ruleset path with a JMX management bean (MBean). To use a hosted transparent decision service, install Rule Execution Server on your web server or application server, and deploy the hosted transparent decision service archive to the same server.

Rule Execution Server automatically exposes any deployed rulesets as a web service, regardless of the underlying type of XOM (Java or dynamic). A *Web Services Description Language (WSDL)* file is generated for each deployed ruleset archive. These rulesets can be exposed as a web service without any code deployment.

## Unit summary

- Describe the Rule Execution Server architecture
- Describe the platforms in which Rule Execution Server can be deployed
- Explain the APIs that are used to create client applications that request ruleset execution with Rule Execution Server

Figure 13-61. Unit summary

## Review questions

1. True or False: The XU uses the execution components to execute a ruleset.
2. True or False: With the Rule Execution Server console, you can manage deployed RuleApps and XOMs.
3. True or False: You can use a non-canonical ruleset path to request the execution of a ruleset by Rule Execution Server.
4. What can the Execution Unit (XU) do? Choose all that apply.
  - A. Manage the rule engines
  - B. Load rulesets
  - C. Pass data between the application and the rule engine
  - D. Create several rule engine instances



Figure 13-62. Review questions

Write your answers here:

- 1.
- 2.
- 3.
- 4.

## Review answers

1. True or False: The XU uses the execution components to execute a ruleset.  
The answer is True.
  
2. True or False: With the Rule Execution Server console, you can manage deployed RuleApps and XOMs.  
The answer is True.
  
3. True or False: You can use a non-canonical ruleset path to request the execution of a ruleset by Rule Execution Server.  
The answer is True.
  
4. What can the Execution Unit (XU) do? Choose all that apply.
  - A. Manage the rule engines
  - B. Load rulesets
  - C. Pass data between the application and the rule engine
  - D. Create several rule engine instancesThe answer is A, B, C, and D.

Executing rules with Rule Execution Server

© Copyright IBM Corporation 2019

Figure 13-63. Review answers

## Exercise: Exploring the Rule Execution Server console

Executing rules with Rule Execution Server

© Copyright IBM Corporation 2019

*Figure 13-64. Exercise:*

## Exercise introduction

- Work with Rule Execution Server console tools
- Manage RuleApps and rulesets through the Rule Execution Server console



Figure 13-65. Exercise introduction

# Unit 14. Auditing and monitoring ruleset execution

## Estimated time

01:00

## Overview

In this unit, you learn how to audit and monitor ruleset execution with Decision Warehouse.

## How you will check your progress

- Review
- Exercise

## Unit objectives

- Audit the execution of rulesets with Decision Warehouse
- Monitor ruleset execution with the Rule Execution Server console

*Figure 14-1. Unit objectives*

## Topics

- Auditing ruleset execution
- Monitoring ruleset execution

Auditing and monitoring ruleset execution

© Copyright IBM Corporation 2019

*Figure 14-2. Topics*

## 14.1. Auditing ruleset execution

## Auditing ruleset execution

Auditing and monitoring ruleset execution

© Copyright IBM Corporation 2019

*Figure 14-3. Auditing ruleset execution*

## Auditing ruleset execution

- Auditing your rules might be required:
  - When rules are used in applications to make decisions that are based on real data, you might need to review details of ruleset execution
  - Storing and generating ruleset execution traces helps auditors see all rules that are associated with a decision
- To audit your rule execution, use Decision Warehouse
- IBM ODM on Cloud
  - ODM on Cloud does not support Decision Warehouse

Figure 14-4. Auditing ruleset execution

After your rules are in production and your application is producing decisions based on real user data, you need the ability to determine which decisions were taken and how they were reached.

Details about executed rules can help users, such as an auditor, to understand what happened in the decision result.

This section describes how you can audit decision execution with Decision Warehouse.

## Decision Warehouse

- Decision Warehouse provides a means to store, filter, and view rule execution activity
- When you enable ruleset monitoring, Rule Execution Server generates ruleset decision traces behind the scene, and saves them in Decision Warehouse
- Decision Warehouse stores decision traces in a database
- A trace contains information about how a decision was made: the executed ruleflow, the path of executed ruleflow tasks, and the rules fired

Auditing and monitoring ruleset execution

© Copyright IBM Corporation 2019

*Figure 14-5. Decision Warehouse*

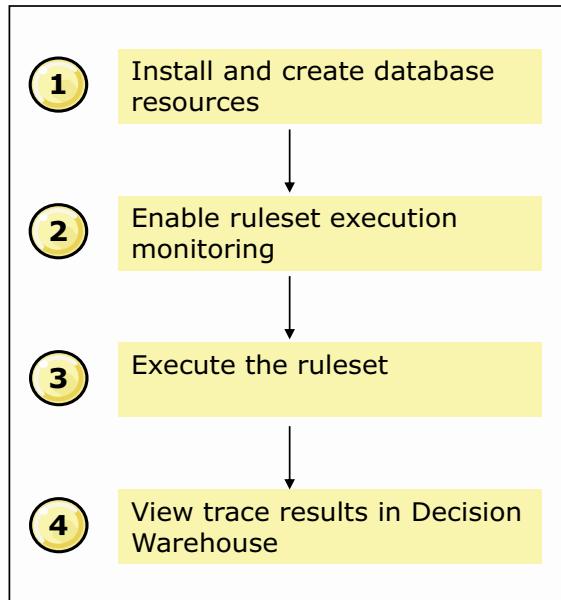
In Decision Warehouse, you can view stored decision traces.

When you enable ruleset monitoring, Rule Execution Server generates decision traces and saves them to Decision Warehouse. Decision Warehouse stores them in a database.

The trace contains information about how the decision was made, including the executed ruleflow, the path taken through the ruleflow, and the specific rules that were fired.

These details are intended to help users understand what happened as a result of executing the ruleset.

## Default use of Decision Warehouse



Auditing and monitoring ruleset execution

© Copyright IBM Corporation 2019

Figure 14-6. Default use of Decision Warehouse

This workflow illustrates the default use of Decision Warehouse, without any customization.

## Step 1: Install and create database resource

- Complete the installation of Rule Execution Server and Decision Warehouse
- This installation is typically done through the Rule Execution Server console Installation Manager

Auditing and monitoring ruleset execution

© Copyright IBM Corporation 2019

*Figure 14-7. Step 1: Install and create database resource*

As a first step, you install and create the database resource to complete the installation of Rule Execution Server and Decision Warehouse on your application server.

## Step 2: Enable rule execution monitoring

- To have traces, you must enable ruleset monitoring
  - After traces are enabled, Rule Execution Server generates ruleset decision traces behind the scene and saves them in Decision Warehouse
- To do so, set the following ruleset properties:
  - `monitoring.enabled`
  - `rulesetSEQUENTIAL.trace.enabled` (if the ruleset contains rule tasks with the Sequential or Fastpath execution modes)
  - `ruleset.BOM.enabled` (optional)

Figure 14-8. Step 2: Enable rule execution monitoring

Next, you enable monitoring of ruleset execution. This step requires that you set ruleset properties, as listed on the slide.

You already saw how to manage ruleset properties in Rule Designer and Business console.

In the next topic, you learn a new way of defining these properties in Rule Execution Server console.

## Step 3: Execute the ruleset

- Execute your ruleset by using:
  - Decision Validation Services
  - A client application
  - A hosted transparent decision service

Auditing and monitoring ruleset execution

© Copyright IBM Corporation 2019

*Figure 14-9. Step 3: Execute the ruleset*

As a third step, you execute the ruleset to verify that the traces were correctly generated and stored.



## Step 4-a: View stored decision traces

- Sign in to the Rule Execution Server console by using the `resMonitor` or `resAdmin` role
- In Rule Execution Server console, click the **Decision Warehouse** tab
- Click **Search**

65 Decision(s) found Display by 10

| Decision ID                          | Date                | Ruleset Version   | Number of rules fired | Decision Trace                        | Processing Time (ms) |
|--------------------------------------|---------------------|---|-----------------------|---------------------------------------|----------------------|
| Sab23322-97ef-4787-b892-58166e39cebc | 2009-03-26 18:02:58 | /prodra1238086969350_90b385fe_4e4c_4842_b408_4d1e9dcadfd6 /1.0/prodrs1238086969350_90b385fe_4e4c_4842_b408_4d1e9dcadfd6/1.0 | 1                     | <a href="#">View Decision details</a> | 416                  |
| 50a3601b-20fa-4f78-abf7-6935f6d4516c | 2009-03-19 15:30:10 | /prodra1237473006170_519ba66c_0ad6_4142_a3e9_3d7df42c7945 /1.0/prodrs1237473006170_519ba66c_0ad6_4142_a3e9_3d7df42c7945/1.0 | 1                     | <a href="#">View Decision details</a> | 35                   |
| 6c44f221-5d0e-486f-89d9-ceab48042d84 | 2009-03-19 15:30:10 | /prodra1237473006170_519ba66c_0ad6_4142_a3e9_3d7df42c7945 /1.0/prodrs1237473006170_519ba66c_0ad6_4142_a3e9_3d7df42c7945/1.0 | 0                     | <a href="#">View Decision details</a> | 37                   |
| a18b5329-ce90-4ea5-9a37-fb01eb01e3e6 | 2009-03-19 15:30:08 | /prodra1237473006170_519ba66c_0ad6_4142_a3e9_3d7df42c7945 /1.0/prodrs1237473006170_519ba66c_0ad6_4142_a3e9_3d7df42c7945/1.0 | 1                     | <a href="#">View Decision details</a> | 205                  |
| 385ee32d-6bc8-49b1-a648-cd51c0c6f07d | 2009-03-19 15:27:48 | /prodra1237472858960_f57cf85_481b_4e0e_a50a_9bd94a70e73e /1.0/prodrs1237472858960_f57cf85_481b_4e0e_a50a_9bd94a70e73e/1.0   | 1                     | <a href="#">View Decision details</a> | 172                  |
| 6602fed-6c46-4fe5-8bc8-f10839340a28  | 2009-03-19 15:27:48 | /prodra1237472858960_f57cf85_481b_4e0e_a50a_9bd94a70e73e /1.0/prodrs1237472858960_f57cf85_481b_4e0e_a50a_9bd94a70e73e/1.0   | 0                     | <a href="#">View Decision details</a> | 47                   |
| 41565ac8-2a56-47c4-b5ce-e8d925c8c9fa | 2009-03-19 15:27:47 | /prodra1237472858960_f57cf85_481b_4e0e_a50a_9bd94a70e73e /1.0/prodrs1237472858960_f57cf85_481b_4e0e_a50a_9bd94a70e73e/1.0   | 1                     | <a href="#">View Decision details</a> | 505                  |
| 57705e85-ff38-45b3-a716-eb0a45fc676  | 2009-03-13 11:57:58 | /prodra1236941862805_eee830e0_cefa_4105_a0be_7851d0bc552 /1.0/prodrs1236941862805_eee830e0_cefa_4105_a0be_7851d0bc552/1.0   | 4                     | <a href="#">View Decision details</a> | 1138                 |
| 64efc105-58cd-4e71-8350-e550b957a610 | 2009-03-06 19:22:52 | /prodra1236363766960_fad3c26c_ab69_46d9_9763_8d4ca0a422fc /1.0/prodrs1236363766960_fad3c26c_ab69_46d9_9763_8d4ca0a422fc/1.0 | 3                     | <a href="#">View Decision details</a> | 18                   |
| f91fe910-2585-4d18-987a-dbc9e06bf77  | 2009-03-06 19:22:52 | /prodra1236363766960_fad3c26c_ab69_46d9_9763_8d4ca0a422fc /1.0/prodrs1236363766960_fad3c26c_ab69_46d9_9763_8d4ca0a422fc/1.0 | 3                     | <a href="#">View Decision details</a> | 13                   |

1 - 10 out of 65 results ◀◀◀ 1 2 3 4 5 6 7 ▶▶▶

Auditing and monitoring ruleset execution

© Copyright IBM Corporation 2019

Figure 14-10. Step 4-a: View stored decision traces

After executing the ruleset, you check that the results are stored in Decision Warehouse.

To see your decision traces, you use the **Decision Warehouse** tab in the Rule Execution Server console.

## Step 4-b: Filter stored decision traces

- Define filters on the data source so that trace information is displayed only for the events or decisions in which you are interested
- You can filter traces by:
  - Executed ruleset path
  - Decision ID
  - Fired rules
  - Executed tasks
  - Values of the input parameters
  - Values of the output parameters
  - Execution dates
  - Execution times
- Click **Search** to execute the filter you specify
- Click **Clear** to remove the filters

Figure 14-11. Step 4-b: Filter stored decision traces

To filter the results that are visible in Decision Warehouse, you can use the query feature to find results according to specific criteria.



## Step 4-c: View decision details and fired rules

The screenshot shows the IBM Decision Center interface. At the top, a message says "1 Decision(s) found". Below it is a table with columns: Decision ID, Date, Ruleset Version, Number of rules fired, Decision Trace, and Processing Time (ms). A row in the table corresponds to a decision with ID 64a98426-1fc8-4bbd-853f-4a21a5dc5b34, fired 8 rules, and took 94 ms. A "View Decision details" link is highlighted with a red box. Below the table, two panes are shown: "Execution Details" and "Decision Trace". The "Execution Details" pane displays summary information: Decision ID, Date, Executed ruleset path, Processing Time (ms), Number of rules fired, and Number of tasks executed. The "Decision Trace" pane shows a hierarchical tree of ruleflow tasks and artifacts. A specific node in the trace, "computation.initialCorporateScore", is highlighted with a red box. A callout box on the left lists four steps: 1. List of taken decisions, 2. Execution details, 3. Decision trace, and 4. Links to artifacts in Decision Center.

| Decision ID                          | Date                | Ruleset Version                | Number of rules fired | Decision Trace | Processing Time (ms) |
|--------------------------------------|---------------------|--------------------------------|-----------------------|----------------|----------------------|
| 64a98426-1fc8-4bbd-853f-4a21a5dc5b34 | 2011-12-19 19:12:06 | /loanRuleApp/3.0/loanrules/1.0 | 8                     |                | 94                   |

1 - 1 out of 1 results

**Execution Details**

**Decision Trace**

1. List of taken decisions  
2. Execution details  
3. Decision trace  
4. Links to artifacts in Decision Center

Auditing and monitoring ruleset execution

© Copyright IBM Corporation 2019

Figure 14-12. Step 4-c: View decision details and fired rules

For each decision trace, you can click the **View Decision Details** link, which accesses information details, such as which rule tasks and rules were fired, and the input and output parameter values.

The **Execution Details** pane provides overview information, and the **Decision Trace** pane gives the list of ruleflow tasks and rule artifacts that were executed.

## 14.2. Monitoring ruleset execution

## Monitoring ruleset execution

Auditing and monitoring ruleset execution

© Copyright IBM Corporation 2019

*Figure 14-13. Monitoring ruleset execution*

## Monitoring ruleset execution

- With the Rule Execution Server console, you can:
  - Enable the debug mode
  - Enable ruleset monitoring
  - Generate ruleset execution statistics
  - Test ruleset execution
  - View execution-related events in the XU log

Auditing and monitoring ruleset execution

© Copyright IBM Corporation 2019

*Figure 14-14. Monitoring ruleset execution*

Rule Execution Server console provides various tools to help you test and monitor execution.

## Debug mode

- You can enable or disable the debug mode on a ruleset
- By default, the debug mode is disabled on rulesets
- When you enable the debug mode for a specific ruleset, that ruleset, when called, opens in the debugger in a remote Rule Designer session
  - You can use Rule Designer to remotely debug your ruleset execution
  - For example, by putting breakpoints in the ruleset
- The debug URL points to the remote computer where Rule Designer runs

Auditing and monitoring ruleset execution

© Copyright IBM Corporation 2019

Figure 14-15. Debug mode

You can enable or disable the debug mode on a ruleset. When you enable the debug mode for a specific ruleset, that ruleset opens remotely in the Rule Designer debugger.

## Ruleset monitoring options (1 of 5)

- To enable ruleset monitoring, you learned that you must define some ruleset properties, and the ways to do so
- You can also do so in the Monitoring Options pane of the Rule Execution Server console
  - The **monitoring options** section summarizes the trace options that are stored in Decision Warehouse
  - Click **Edit** in this pane to select the appropriate options

Figure 14-16. Ruleset monitoring options (1 of 5)

You can enable ruleset execution monitoring and use filters to specify what is stored in Decision Warehouse when you execute the ruleset.

## Ruleset monitoring options (2 of 5)

The screenshot displays two side-by-side panels for managing ruleset monitoring options. Both panels include a header with a gear icon and the number of properties (2 or 3).

**Left Panel (2 properties):**

- Properties:** monitoring.enabled, ruleset.sequential.trace.enabled
- Upload properties from file:** Choose file: [Browse...]
- Hide Monitoring Options:**
  - monitoring options:** Enable tracing in the Decision Warehouse (checkbox checked). Store the values of the ruleset parameters to:  BOM format with optional filter (radio button unselected)  Native format (radio button selected).

**Right Panel (3 properties):**

- Properties:** monitoring.enabled, ruleset.bom.enabled (highlighted), ruleset.sequential.trace.enabled
- Upload properties from file:** Choose file: [Browse...]
- Hide Monitoring Options:**
  - monitoring options:** Enable tracing in the Decision Warehouse (checkbox checked). Store the values of the ruleset parameters to:  BOM format with optional filter (radio button selected)  Native format (radio button unselected).

Auditing and monitoring ruleset execution

© Copyright IBM Corporation 2019

Figure 14-17. Ruleset monitoring options (2 of 5)

If you set the property `ruleset.bom.enabled` to `true`, the input and output data is serialized and stored in a BOM representation of the data in XML format.

If you set it to `false`, two things can happen:

- If the ruleset is based on a dynamic XOM, the input and output parameters are stored as XML.
- If the ruleset is based on a Java XOM, the `toString` method of the ruleset parameter type converts or serializes the list of objects in the parameters into a memory buffer.

Because these properties can affect performance, make sure you know what the options do.

If you want to turn off BOM serialization, you can select **Native format** instead.

# IBM Training



## Ruleset monitoring options (3 of 5)

Hide Properties

4 properties

| Select All               | Name                            | Value                                       |
|--------------------------|---------------------------------|---|
| <input type="checkbox"/> | monitoring.enabled              | true  |
| <input type="checkbox"/> | monitoring.inout.filters        | borrower.some.heavy.property,loan.longfield |
| <input type="checkbox"/> | ruleset.bom.enabled             | true  |
| <input type="checkbox"/> | rulesetSEQUENTIAL,trace.enabled | true  |

properties 1 - 4 of 4

prev 10 next 10

**Upload properties from file**

Choose file:      Override ex

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



Auditing and monitoring ruleset execution

© Copyright IBM Corporation 2019

Figure 14-18. Ruleset monitoring options (3 of 5)

If you use **BOM format with an optional filter**, you can also set a filter on the objects that the parameters use to remove information about these objects from the trace.



## Ruleset monitoring options (4 of 5)

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  
 List of Rules Not Fired  
 Bound Object by Rule  
 System Properties  
 Working Memory with optional filter:

Auditing and monitoring ruleset execution

© Copyright IBM Corporation 2019

- The possible trace options are as follows:
  - 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
  - List of Rules Not Fired
  - Bound Object by Rule
  - System Properties
  - Working Memory with optional filter

Figure 14-19. Ruleset monitoring options (4 of 5)

On this slide, you see a list of some of the possible trace options.

IBM Training 

## Ruleset monitoring options (5 of 5)

| Select All               | Name                            | Value   |
|--------------------------|---------------------------------|---|
| <input type="checkbox"/> | monitoring.enabled              | <input checked="" type="checkbox"/> true  |
| <input type="checkbox"/> | monitoring.filters              | <input checked="" type="checkbox"/> INFO_EXECUTION_DATE=true,INFO_EXECUTION_DURATION=true |
| <input type="checkbox"/> | monitoring.inout.filters        | <input checked="" type="checkbox"/> borrower.some.heavy.property,loan.longfield           |
| <input type="checkbox"/> | ruleset.bom.enabled             | <input checked="" type="checkbox"/> true  |
| <input type="checkbox"/> | rulesetSEQUENTIAL.trace.enabled | <input checked="" type="checkbox"/> true  |

properties 1 - 5 of 5      prev 10 next 10

**Upload properties from file**

Choose file:      Override existing properties

**Hide Monitoring Options**

 **monitoring options**

✓ Tracing in the Decision Warehouse is currently enabled

Ruleset parameters will be stored as BOM XML (with filter borrower.some.heavy.property,loan.longfield)

The following execution traces will be stored:

- Execution Date
- Execution Duration



Auditing and monitoring ruleset execution

© Copyright IBM Corporation 2019

Figure 14-20. Ruleset monitoring options (5 of 5)

You can filter which traces are stored in Decision Warehouse by using the `monitoring.filters` property on your ruleset.

If you remove the filters property, all possible traces are included.

## Ruleset statistics (1 of 3)

- Ruleset statistics provide information about ruleset execution such as the number of times a ruleset was executed and how long the execution took
- In the Rule Execution Server console, you see the ruleset statistics in the Ruleset Statistics View

| Server  | Execution Unit Name | Statistics               |                                      |                |
|---|---------------------|--------------------------|--------------------------------------|----------------|
|   |                     | Metric                   | Ruleset Execution                    | Task Execution |
|  SamplesCell - SamplesNode - SamplesServer | default             | Count                    | 1                                    | Not Available  |
|   |                     | Total Time (ms)          | 47                                   | Not Available  |
|   |                     | Average Time (ms)        | 47.0                                 | Not Available  |
|   |                     | Min. Time (ms)           | 47                                   | Not Available  |
|   |                     | Max. Time (ms)           | 47                                   | Not Available  |
|   |                     | Last Execution Time (ms) | 47                                   | Not Available  |
|   |                     | First Execution Date     | Dec 21, 2011 4:36:00 PM<br>GMT+01:00 | Not Available  |
|   |                     | Last Execution Date      | Dec 21, 2011 4:36:00 PM<br>GMT+01:00 | Not Available  |

Auditing and monitoring ruleset execution

© Copyright IBM Corporation 2019

Figure 14-21. Ruleset statistics (1 of 3)

You can also generate statistics on the previous executions of a ruleset.

The Ruleset Statistics View provides a ruleset execution statistics table for each execution unit (XU) in the configuration. It also shows consolidated statistics on the entire cluster so you can see the number of times a ruleset was executed and the execution duration.

## Ruleset statistics (2 of 3)

- A single execution provides results for either the **Ruleset Execution** column or the **Task Execution** column, depending on the execution mode:
  - **Ruleset Execution:** The application calls the `IlrContext.fireAllRules` method for the ruleset
  - **Task Execution:** The application calls the `IlrContext.executeTask` method for the ruleset

Figure 14-22. Ruleset statistics (2 of 3)

## Ruleset statistics (3 of 3)

- Regular execution statistics and debug execution statistics are not mixed:
  - When the ruleset is in debug mode, statistics are only on debug executions
  - If you disable the debug mode, ruleset statistics are reset

Figure 14-23. Ruleset statistics (3 of 3)

To avoid any confusion, regular execution statistics and debug statistics are not mixed.

## Test of ruleset execution

- The Rule Execution Server console provides a web testing interface for you to test rulesets that are associated with a managed XOM
- In this testing interface, you can enter ruleset parameter values and call the deployed ruleset
  - You can construct the Java input parameters by using the options that are provided in the interface, or you can temporarily set the ruleset property `ruleset.managedxom.uris` to the location of the Java XOM files

Figure 14-24. Test of ruleset execution

You can test your ruleset execution by using a web interface. This interface is not a programmatic interface and is not designed for automated execution. You only use this feature when the console is deployed in a Java Enterprise environment where an execution unit is reachable through a JNDI lookup.

You can use this interface to verify that your deployment conforms to the configuration by checking that the result of the XU lookup diagnostic test is green.

If you add the `ruleset.managed.xom.uris` property to a ruleset, all execution requests use this same property value. However, this is probably not the intended behavior that you want in a production environment.

## Logged events on Execution Units

- In the Rule Execution Server console, you can view the events that are logged on the XUs that were used to execute a ruleset
- You can also modify how the events information is displayed

*Figure 14-25. Logged events on Execution Units*

You can view the events that are logged on the XU that were used to execute a ruleset, and customize how that information is displayed.

## Unit summary

- Audit the execution of rulesets with Decision Warehouse
- Monitor ruleset execution with the Rule Execution Server console

Auditing and monitoring ruleset execution

© Copyright IBM Corporation 2019

*Figure 14-26. Unit summary*

## Review questions

1. True or False: To have decision traces, you must enable ruleset monitoring.
2. True or False: If you deploy the RuleApp from Rule Designer, you have links to rule artifacts in Rule Designer available from the selected decision in Decision Warehouse.
3. True or False: In the Rule Execution Server console, you can test any of the deployed rulesets.



Figure 14-27. Review questions

## Review answers

1. True or False: To have decision traces, you must enable ruleset monitoring.  
**The answer is True.**
  
2. True or False: If you deploy the RuleApp from Rule Designer, you have links to rule artifacts in Rule Designer available from the selected decision in Decision Warehouse.  
**The answer is False. If you deploy the RuleApp from Decision Center, you have links to rule artifacts in Decision Center available from the selected decision in Decision Warehouse.**
  
3. True or False: In the Rule Execution Server console, you can test any of the deployed rulesets.  
**The answer is False. In the Rule Execution Server console, you can test only the deployed rulesets that are associated with a managed XOM.**



Auditing and monitoring ruleset execution

© Copyright IBM Corporation 2019

Figure 14-28. Review answers

## Exercise: Auditing ruleset execution through Decision Warehouse

Auditing and monitoring ruleset execution

© Copyright IBM Corporation 2019

*Figure 14-29. Exercise: Auditing ruleset execution through Decision Warehouse*

## Exercise introduction

- Enable monitoring for ruleset execution
- Retrieve decision traces through Decision Warehouse
- Optimize Decision Warehouse
- Delete trace data from Decision Warehouse



Auditing and monitoring ruleset execution

© Copyright IBM Corporation 2019

*Figure 14-30. Exercise introduction*

---

# Unit 15. Working with the REST API

## Estimated time

00:45

## Overview

This unit teaches you how to use the REST API for ruleset execution and how to work with API Connect.

## How you will check your progress

- Review
- Exercise

## Unit objectives

- Describe the ruleset execution REST API
- Expose a decision service as an API

*Figure 15-1. Unit objectives*

## Topics

- Overview of the ruleset execution REST API
- Testing ruleset execution with the REST API
- Working with OpenAPI and API Connect

*Figure 15-2. Topics*

## 15.1. Overview of the ruleset execution REST API

## Overview of the ruleset execution REST API

Working with the REST API

© Copyright IBM Corporation 2019

*Figure 15-3. Overview of the ruleset execution REST API*

## REST API for ruleset execution

- Rule Execution Server provides a Representational State Transfer (REST) API for ruleset execution
  - Provides XML generation and XSD validation
  - Execute rulesets with XML format through the HTTP protocol
- The REST service for ruleset execution provides these benefits:
  - No client library or complex configuration is required to interact with a remote Rule Execution Server instance
  - Work across platforms or from various client applications
  - Easy to switch from local to remote Rule Execution Server execution
- You can test REST API ruleset execution requests in Rule Execution Server console

Figure 15-4. REST API for ruleset execution

You can test ruleset execution in Rule Execution Server through the REST API. This API provides XML payload sample generation

- XML and XSD validation
- Ruleset execution services

With these REST services, you do not need to add any client libraries or perform complex configuration, which is typically required with the JMX API.

By using the REST API, you can work across environments or from other client applications, such as JavaScript clients. Also, you can easily switch from local execution to a remote Rule Execution Server instance.

To verify that your ruleset execution requests use well-formatted XML, you can test the REST API requests in the Rule Execution Server console. You can also use REST API to integrate ruleset execution with other IBM products.

## Endpoint URIs

- The REST API includes URIs that represent the rulesets

- URI format**

`http://host:port/DecisionService/rest/v1/rulesetPath?options`

- host:port/DecisionService:** Host address and port for the HTDS application
- /rest/v1:** Context root and version of the REST service
- rulesetpath:** Can be either canonical or non-canonical
- options:** Options for generation of WADL file
  - inline:** Generates and displays a WADL file with the XSD schema included
  - zip:** Generates a compressed file for download

Figure 15-5. Endpoint URIs

The endpoint URIs represent the rulesets, which are the Rule Execution Server resources for the REST execution service.

The first part of the URI consists of the host address and port of the hosted transparent decision service (HTDS) application.

In the next part, /rest/v1, rest means the REST service context root and v1 is the version number of the REST service.

In the ruleset path, the short ruleset path or the canonical ruleset path is expected. A canonical ruleset path includes the name and version number of the rule application, followed by the name and version number of the ruleset. A short ruleset path leaves out one or both version numbers, which would be the names of the rule application and ruleset.

The valid value for the last options can be inline or zip parameter for the Web Application Description Language (WADL) generation. If you set the inline parameter, the XSD schema is included in the WADL and displayed to you right away. Otherwise, use the zip option to download the compressed file and view it locally.

## HTTP methods

- GET method
  - Generate a sample XML payload
  - Retrieve the WADL for a specified ruleset
- POST method
  - Create an execution request
  - Validate an XML payload
- HTTP header fields
  - The **Accept-Language** header field is equivalent to the **accept-language** URI parameter for a list of languages that are valid in response message
  - The **Content-Type** field accepts only: **application/xml**

Figure 15-6. HTTP methods

The REST API supports HTTP GET and POST methods. You use the GET method to generate the sample XML payload or retrieve the WADL representation for the ruleset.

You use the POST method to validate an XML payload and to execute the ruleset. The request body for the execution request contains the XML payload, so before sending the request, it is good practice to first validate it.

For the HTTP header, the **Accept-Language** field is the same as the **accept-language** URI parameter. It takes a list of languages as valid values.

Only XML format is supported, so the **Content-Type** field accepts only the **application/xml** value.

## Request and response schema

- Request contains:
  - The ruleset IN and INOUT parameters in alphabetical order
  - A Decision ID (optional)
  - A trace filter (optional)
- Execution response (XML) contains:
  - The ruleset INOUT and OUT parameters, in alphabetical order
  - The Decision ID
  - The trace, if specified in the request
- Validation responses are returned in JSON format
  - Responses to valid requests contain an empty JSON list []
  - Responses to invalid requests describe the type and location of the error
- Example of validation response to invalid request:  

```
{"type": "Error", "line": 9, "column": 22, "message": "cvc-datatype-valid.1.2.1: '5d' is not a valid value for 'integer'."}
```

[Working with the REST API](#)

© Copyright IBM Corporation 2019

*Figure 15-7. Request and response schema*

In the ruleset execution REST service, requests and responses follow different schemas, depending on whether the ruleset XOM is based on XML classes or on Java classes. The schema determines how the types are serialized.

With the ruleset signature, the request part is composed of the following elements:

- The IN and INOUT parameters of the ruleset, in alphabetical order.
- An optional decision ID if you want to set it to a specific value and an optional trace filter.

The execution response consists of:

- The INOUT and OUT parameters of the ruleset, in alphabetical order.
- The decision ID, either the default value or the value that you set in the request.
- If you set trace filter in request, the trace is returned in response.

Notice that the execution response is sent in XML format. The XML payload is analyzed against the generated XSD files.

The validation response is returned in JSON format.

- If the request is valid, the response is an empty JSON list [].
- If the request is invalid, the tool returns the list of errors.

- Each response to an invalid request includes:
  - **Type:** Error, Fatal, or Warning
  - **Line:** The number of the line that contains the error in the XML file
  - **Column:** The column number that contains the error in the XML file
  - **Message:** Description of error

The example on the slide shows an error in line 9, column 22. Based on the XML schema data type validation rule, the value `5d` is not a valid integer.

## XML serialization of the XOM

- Primitive types
  - Java types and classes that can be written as inline strings
  - The ruleset parameter name (in the parameter namespace) is used to serialize a primitive type as a ruleset parameter
- Dynamic XOMs
  - Root element depends on how the ruleset parameters are defined
  - If the parameter is an XML element in the ruleset signature, it is used as the root element name in requests and responses
  - If the parameter is a complex or primitive Java type, the parameter name is used as the root element name of the parameter in requests and responses, within the configured parameter namespace
- Java XOMs
  - The ruleset parameter name (in the parameter namespace) is used as the name of the XML root element
  - Arrays are supported, including simple and multi-dimensional arrays

*Figure 15-8. XML serialization of the XOM*

In the REST service for ruleset execution, primitive Java types, XSD types, and Java XOM classes are serialized differently.

For the primitive types, the REST service XML element that is used to serialize the primitive type is the name of the ruleset parameter in the parameter namespace. Examples of primitive types include `boolean`, `string`, or simple classes like `java.lang.Boolean`.

For dynamic XSD type XOMs, the request and response XML root element depends on how the ruleset parameters are defined. It is serialized from the original XSD schema.

If the dynamic XOM ruleset parameter is an XML element, then this XML element name becomes the XML request root element name.

If the dynamic XOM ruleset parameter is the Java type, the parameter name is used as the root element name in the request and response.

For Java XOMs, basically the ruleset parameter name is in use. The Java array type is also supported, for both simple and multidimensional arrays.

## WADL representation

- You generate WADL representation to write the XML payload of a specific ruleset execution request
  - URI:  
`http://{host}:{port}/DecisionService/rest/v1/ {rulesetPath}/wadl`
- A set of XSD files are also generated, but separated from WADL file
  - Optional parameters:
    - `inline`: The `.xsd` files are included in the `.wadl` file
    - `zip`: The `.xsd` files and `.wadl` file are generated in a compressed file
- Response result:
  - The WADL file
    - One GET method function: XML payload sample generation
    - Two POST method functions: XML payload validation and ruleset execution
  - XSD files to describe XML representation of the input and output objects

Figure 15-9. WADL representation

The WADL representation contains all the information of the request and response elements. You can use the WADL representation as a reference while you develop your client application.

By default, a set of XSD files are generated and they are separated from the WADL file. If you specify the `inline` option, the XSD file contents are included in the WADL file. Alternatively, to get WADL code and its XSD files to a compressed file, add the `zip` parameter.

If it is a valid request, you get a WADL representation of the request and response documentation. It contains one GET method function for XML payload sample generation, and two POST method functions for XML payload validation and ruleset execution. The attached XSD files describe the XML representation of the input and output objects.

## Using the REST execution API for ruleset execution

1. Ensure that the ruleset is deployed
2. Generate a sample XML payload
  - Example:  
GET `http://host:port/DecisionService/rest/v1/rulesetPath/xml`
  - Use the sample XML fragment as a starting point for writing an XML request
3. Validate the XML structure
  - Example:  
POST  
`http://host:port/DecisionService/rest/v1/rulesetPath/validate`
4. Send the execution request
  - Example:  
POST  
`http://host:port/DecisionService/rest/v1/rulesetPath`
5. Expect the execution response to be returned in XML format

*Figure 15-10. Using the REST execution API for ruleset execution*

First, you make sure that the ruleset is deployed.

Then, you use the REST service to generate a sample XML payload. You use the sample XML fragment as a starting point to write an XML request.

When your XML request is ready, you can validate its XML structure by posting a “validate” request.

After validating the XML request, you can then send the request as the payload of an HTTP call by using POST method. The execution response is sent back in XML format. If the request is not valid, error messages are returned in JSON format.

## 15.2. Testing ruleset execution with the REST API

## Testing ruleset execution with the REST API

Working with the REST API

© Copyright IBM Corporation 2019

*Figure 15-11. Testing ruleset execution with the REST API*



## Executing a ruleset (1 of 2)

- Open Rule Execution Server console to the **Explorer** tab

The screenshot shows the IBM Rule Execution Server console interface. At the top, there's a blue header bar with the IBM logo and the text "IBM Rule Execution Server console". Below the header, a navigation bar has several tabs: Home, Explorer (which is highlighted with a red box), Decision Warehouse, Diagnostics, Server Info, and REST API. Underneath the tabs, a breadcrumb trail reads "Explorer > RuleApps > RuleApp > Ruleset". On the right side of the header, there's a link "Skip to main content".

- In the Ruleset View page for a deployed ruleset, retrieve the HTDS description file

The screenshot shows the "Ruleset View" page for a deployed ruleset. At the top, there's a toolbar with various icons and buttons, including "Test Ruleset", "View Statistics", "View Execution Units", "Upload Ruleset Archive", "Add Managed URI", "Add Property", "Edit", and "Retrieve HTDS Description File" (which is highlighted with a red box). Below the toolbar, the URL is displayed as "/miniloanruleapp/1.0/miniloanrules/1.0".

Working with the REST API

© Copyright IBM Corporation 2019

*Figure 15-12. Executing a ruleset (1 of 2)*

To access the REST API test tool, you start from the **Explorer** tab in Rule Execution Server console.

In the Ruleset View for the deployed ruleset, click **Retrieve HTDS Description File**.



## Executing a ruleset (2 of 2)

- When you retrieve the HTDS description file for your ruleset, you select the **REST** option for testing

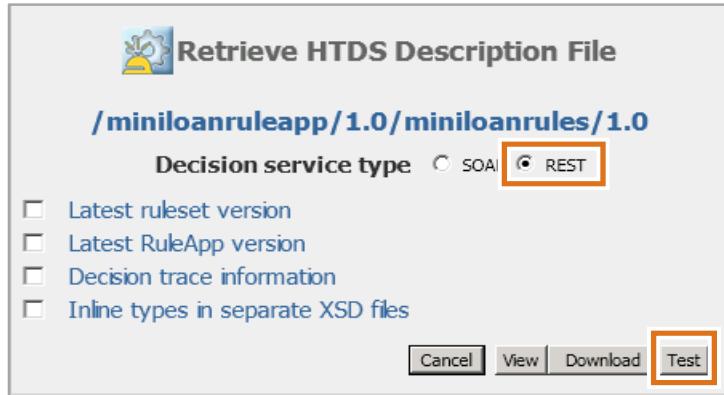


Figure 15-13. Executing a ruleset (2 of 2)

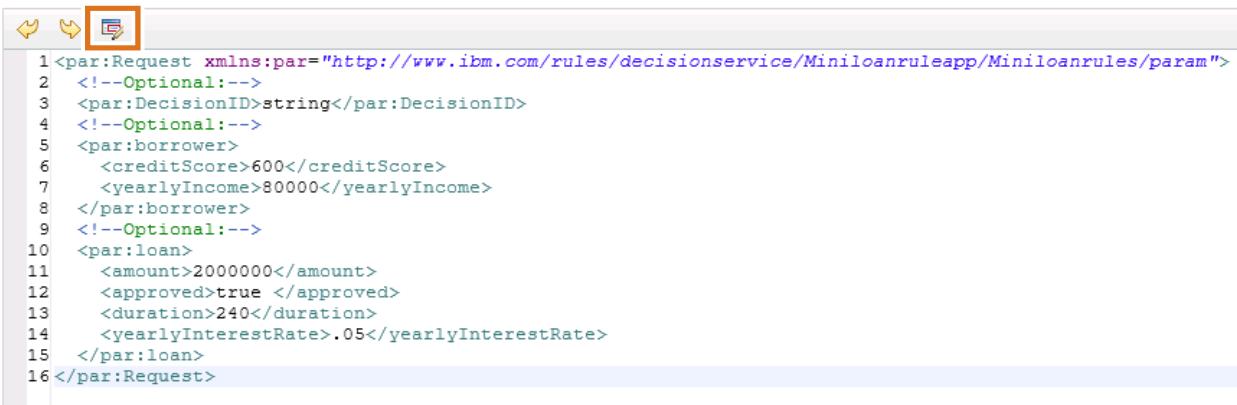
In the dialog that opens, you select the **REST** option and click **Test**.

## Validating the XML execution request

- Create the execution request in XML
- Before sending the request, you validate the XML structure

### Decision Service : /miniloanruleapp/1.0/miniloanrules/1.0 REST Service

Execution Request:



```

1 <par:Request xmlns:par="http://www.ibm.com/rules/decisionservice/Miniloanruleapp/Miniloanrules/param">
2   <!--Optional:-->
3   <par:DecisionID>string</par:DecisionID>
4   <!--Optional:-->
5   <par:borrower>
6     <creditScore>600</creditScore>
7     <yearlyIncome>80000</yearlyIncome>
8   </par:borrower>
9   <!--Optional:-->
10  <par:loan>
11    <amount>2000000</amount>
12    <approved>true </approved>
13    <duration>240</duration>
14    <yearlyInterestRate>.05</yearlyInterestRate>
15  </par:loan>
16 </par:Request>

```

Working with the REST API

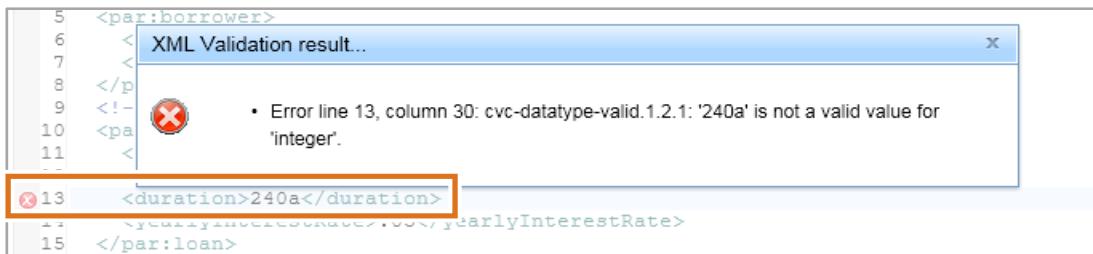
© Copyright IBM Corporation 2019

*Figure 15-14. Validating the XML execution request*

The test tool opens where you can create the execution request. Before you send the request, you can validate the XML structure in this interface.

## Validation response in JSON

- If an error is detected in the request, the validation response shows the type and location of the error in JSON format



The screenshot shows a code editor with XML code. A tooltip window titled "XML Validation result..." is open over line 13. The tooltip contains an error message: "Error line 13, column 30: cvc datatype-valid.1.2.1: '240a' is not a valid value for 'integer'." Below the tooltip, the XML code is shown with line 13 highlighted by a red box. The code is as follows:

```

5  <par:borrower>
6    <!-- XML Validation result... -->
7    <!--
8    </p>
9    <!--
10   <pa>
11     <!--
12       <duration>240a</duration>
13       <!--
14         <yearlyInterestRate>.00</yearlyInterestRate>
15   </par:loan>

```

- In this case, the expected value is an integer, but a letter was accidentally added in line 13, which caused an error

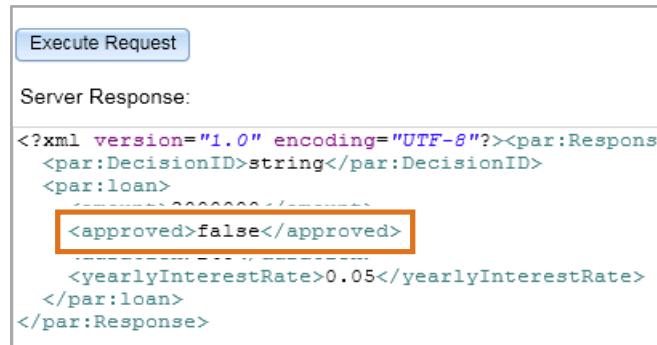
*Figure 15-15. Validation response in JSON*

If your request is not formatted correctly, you get a validation response in JSON format.

In this example, you see an error in line 13, column 30 where the expected value is an integer, but a letter was added to the line, which caused an error.

## Execution response

- After sending the validated request, the execution response is returned in XML format
  - In this case, the **approved** attribute indicates successful results



```

Execute Request

Server Response:
<?xml version="1.0" encoding="UTF-8"?><par:Response
  <par:DecisionID>string</par:DecisionID>
  <par:loan>
    -----
    -----
    <approved>false</approved>
    -----
    <yearlyInterestRate>0.05</yearlyInterestRate>
  </par:loan>
</par:Response>

```

- Understanding how to test ruleset execution REST API can be useful to develop your client application

*Figure 15-16. Execution response*

After sending a validated request, the execution response is returned in XML format. In the example here, the approved attribute is set to false.

During the exercise, you see how to use the REST API to test ruleset execution.

## 15.3. Working with OpenAPI and API Connect

## Working with OpenAPI and API Connect

Working with the REST API

© Copyright IBM Corporation 2019

*Figure 15-17. Working with OpenAPI and API Connect*

## OpenAPI (1 of 2)

- OpenAPI is a standardized version of the Swagger specification
  - Language-independent way to present REST APIs
- With ODM Decision Connect, reusable decisions can be exposed as OpenAPI-based public APIs to be invoked directly by applications
- Create OpenAPI definitions directly from the Rule Execution Server console
  - Use OpenAPI definitions to prototype and invoke ODM decisions
  - Imported OpenAPI definitions to IBM API Connect to become managed APIs

Figure 15-18. OpenAPI (1 of 2)

## OpenAPI (2 of 2)

- Decision services that are deployed to Rule Execution Server can be exposed as web services with REST/JSON endpoints
  - Describe the RESTful API for this web service to facilitate use by calling applications
  - Turn the API into a managed API for use with Decision Connect or an API management platform
- OpenAPI specification (Swagger) allows API consumers to easily understand and use APIs
  - To describe RESTful web service, create a Swagger file based on specification format
- To publish the decision as an API, it must be transformed from a web service interface to having a Swagger 2.0 API documented in YAML

Figure 15-19. OpenAPI (2 of 2)

When you deploy a decision service in Rule Execution Server, the decision service can be exposed as a web service with REST/JSON endpoints. To describe and document a RESTful web service, you must create a Swagger file that is based on the specification format. The Swagger file can then be used to integrate the related web service.

Before the decision can be published to the API catalog in API Connect, it must be transformed from having a web services interface with HTDS to having a Swagger 2.0 API documented in YAML.

## Swagger description file

- Swagger 2.0 file
  - 15 standardized sections
  - Three sections are mandatory
- Main categories for sections (mandatory sections are denoted with **bold** face):
  - Header section: **swagger**
  - Metadata around the API being described: **info**, tags, externalDocs
  - Global configuration: host, basePath, schemes
  - API description: consumes, produces, **paths**, definitions, parameters, responses
  - Security-related descriptions: securityDefinitions, security
- Swagger 2.0 supports two file formats:
  - JSON
  - YAML

Figure 15-20. *Swagger description file*

## Defining parameters (1 of 2)

- All input and input/output parameters must be collated into a single parameter in the HTTP POST operation of the Swagger path
- Example:
  - Input: borrower
  - Input–Output: loan
  - Parameters are combined as: Request
  - Swagger path becomes:

```
parameters:  
  - name: Request  
    description: Loan request payload.  
    required: true  
    in: body  
    schema:  
      $ref: '#/definitions/Request'
```

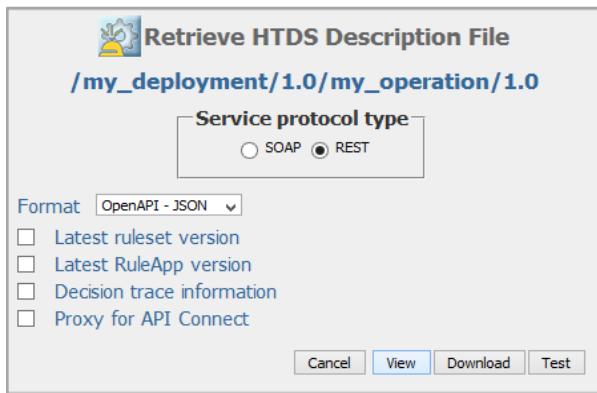
- **Note:** Indentation in YAML files determines the nesting structure. Blank characters at the start of a line are not ignored.

Figure 15-21. Defining parameters (1 of 2)



## Defining parameters (2 of 2)

- To ensure that the parameter is correctly defined, use REST test tool in Rule Execution Server console with JSON format



- The Execution Request in JSON format shows required attributes and expected format

Execution Request: JSON

```

1 {
2   "loan": {
3     "amount": 3,
4     "duration": 3,
5     "yearlyInterestRate": 10517320,
6     "yearlyRepayment": 3,
7     "approved": false,
8     "messages": [
9       "string"
10    ],
11  },
12  "borrower": {
13    "name": "string",
14    "creditScore": 3,
15    "yearlyIncome": 3
16  }
17 }
```

Working with the REST API

© Copyright IBM Corporation 2019

Figure 15-22. Defining parameters (2 of 2)

For all these classes, object types must be created in the Swagger file. All their required attributes must be created as properties with the appropriate type and format.

From these examples, you see the required attributes that need to be described in the Swagger file. For example, the Borrower object requires: name, creditScore, yearlyIncome.

## Response

- All input/output and output parameters are collated into a single response object that is associated with the HTTP request success code 200
- Example response:

```
responses:  
  '200':  
    description: A loan evaluation response  
    schema:  
      $ref: '#/definitions/Response'  
  default:  
    description: Unexpected error  
    schema:  
      $ref: '#/definitions/Error'
```

Figure 15-23. Response

## Using the Swagger file with IBM API Connect

- You can import and run APIs by using IBM API Connect on IBM Cloud
  - Cloud applications introspect the APIs through management tools and invoke them in a secure manner
- To work with APIs, use API Designer
- You must have an IBM Cloud account to work with API Connect and API Designer
  - To install API Designer, you need Node.js installed

Figure 15-24. Using the Swagger file with IBM API Connect

The Swagger file for one or several decision services is the base document to create a managed API in a management platform such as IBM API Connect. Most of the Swagger file content can be reused directly (operations and definitions description in particular). API Connect then acts as a gateway to ODM Rule Execution Server and transmits API calls to that ODM server.

For more information about API Connect, see:

- [www.ibm.com/support/knowledgecenter/en/SSFS6T/com.ibm.apic.overview.doc/api\\_management\\_overview.html](http://www.ibm.com/support/knowledgecenter/en/SSFS6T/com.ibm.apic.overview.doc/api_management_overview.html)
- [developer.ibm.com/apiconnect](http://developer.ibm.com/apiconnect)
- <https://console.bluemix.net/docs/services/apiconnect/index.html#index>

## Unit summary

- Describe the ruleset execution REST API
- Expose a decision service as an API

*Figure 15-25. Unit summary*

## Review questions

1. True or False: As a starting point for writing a ruleset execution request, generate a sample XML payload by retrieving the WADL file.
2. True or False: To make sure that you create a valid ruleset execution request in well-formatted XML, first send a “validate” request.
3. True or False: All execution and validation responses are returned in JSON format.
4. True or False: Decision services can be exposed as APIs.



Working with the REST API

© Copyright IBM Corporation 2019

Figure 15-26. Review questions

Write your answers here:

- 1.
- 2.
- 3.
- 4.

## Review answers

1. True or False: As a starting point for writing a ruleset execution request, generate a sample XML payload by retrieving the WADL file.  
The answer is True.
2. True or False: To make sure that you create a valid ruleset execution request in well-formatted XML, first send a “validate” request.  
The answer is True.
3. True or False: All execution and validation responses are returned in JSON format.  
The answer is False. Execution responses are returned in XML. Responses to validation requests are returned in JSON format.
4. True or False: Decision services can be exposed as APIs.  
The answer is True.



Working with the REST API

© Copyright IBM Corporation 2019

Figure 15-27. Review answers

## Exercise: Executing rules as a hosted transparent decision service (HTDS)

Working with the REST API

© Copyright IBM Corporation 2019

Figure 15-28. Exercise: Executing rules as a hosted transparent decision service (HTDS)

## Exercise introduction



- Retrieve a WSDL description and call the decision service in a generated client
- View and test a ruleset in REST by using an OpenAPI in the Rule Execution Server console
- Retrieve an OpenAPI description for API Connect, and run the decision service through API Connect

Figure 15-29. Exercise introduction

---

# Unit 16. Introducing decision governance

## Estimated time

01:30

## Overview

In this unit, you learn how to identify governance issues and use Operational Decision Manager features to support decision governance.

## How you will check your progress

- Review

## Unit objectives

- Explain governance issues and good practices
- Identify Operational Decision Manager features that support decision governance
- Describe how to implement the decision governance framework

*Figure 16-1. Unit objectives*

## Topics

- What is decision governance?
- Operational Decision Manager support for governance
- Decision governance framework

*Figure 16-2. Topics*

# 16.1. What is decision governance?

## What is decision governance?

Introducing decision governance

© Copyright IBM Corporation 2019

*Figure 16-3. What is decision governance?*

## What is decision governance?

- Management of the lifecycle of decision logic, from initial development through to deployment and maintenance
- Provides an organizational framework that instills confidence in all stakeholders
  - Development team might hesitate to hand over control of decisions to business users
  - Business users might hesitate to accept control for fear of breaking something
- Goal
  - Ensure that business and development teams collaborate effectively
  - Ensure that project outcome meets expectations

[Introducing decision governance](#)

© Copyright IBM Corporation 2019

Figure 16-4. What is decision governance?

Governance is a broad term that involves not only defining processes, but also maintaining those processes and being able to audit them.

Decision governance is management of the decision logic lifecycle, from initial development through to deployment and maintenance.

By using the BRMS approach, business users can bypass the IT team when they edit business policies so that updates can be implemented, tested, and deployed to production with minimal dependence on IT. However, during implementation of a decision management solution, the development team might hesitate to hand over control of decisions to business users. And business users might also hesitate to accept control for fear of breaking something. Governance provides an organizational framework that instills confidence in all stakeholders.

The goal of governance is to ensure that business and development teams collaborate effectively, and that the project outcome meets expectations.

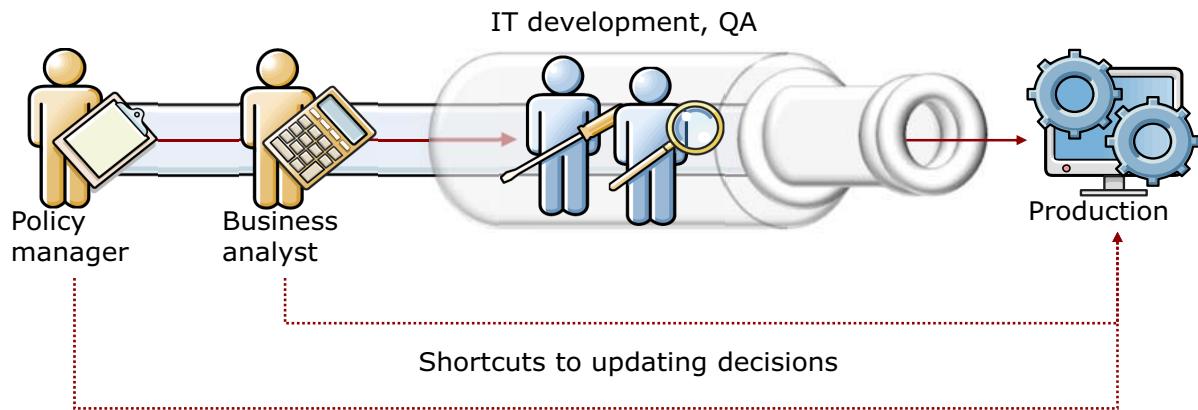
## Why decision governance is required

- Rules and event artifacts play a dual role within an organization
  - From the business perspective, they represent critical business logic
  - From the development perspective, they are part of the actual software that runs on enterprise data
- Decision management must span business and IT teams

*Figure 16-5. Why decision governance is required*

Decision governance must span both business and IT teams so that they can work together seamlessly and efficiently from their different perspectives and tools.

## Traditional approach to maintenance



Introducing decision governance

© Copyright IBM Corporation 2019

*Figure 16-6. Traditional approach to maintenance*

In the traditional approach, the IT development-QA pair can represent a bottleneck. Business policy updates, which are the new requirements that come from the business team, must go through that bottleneck.

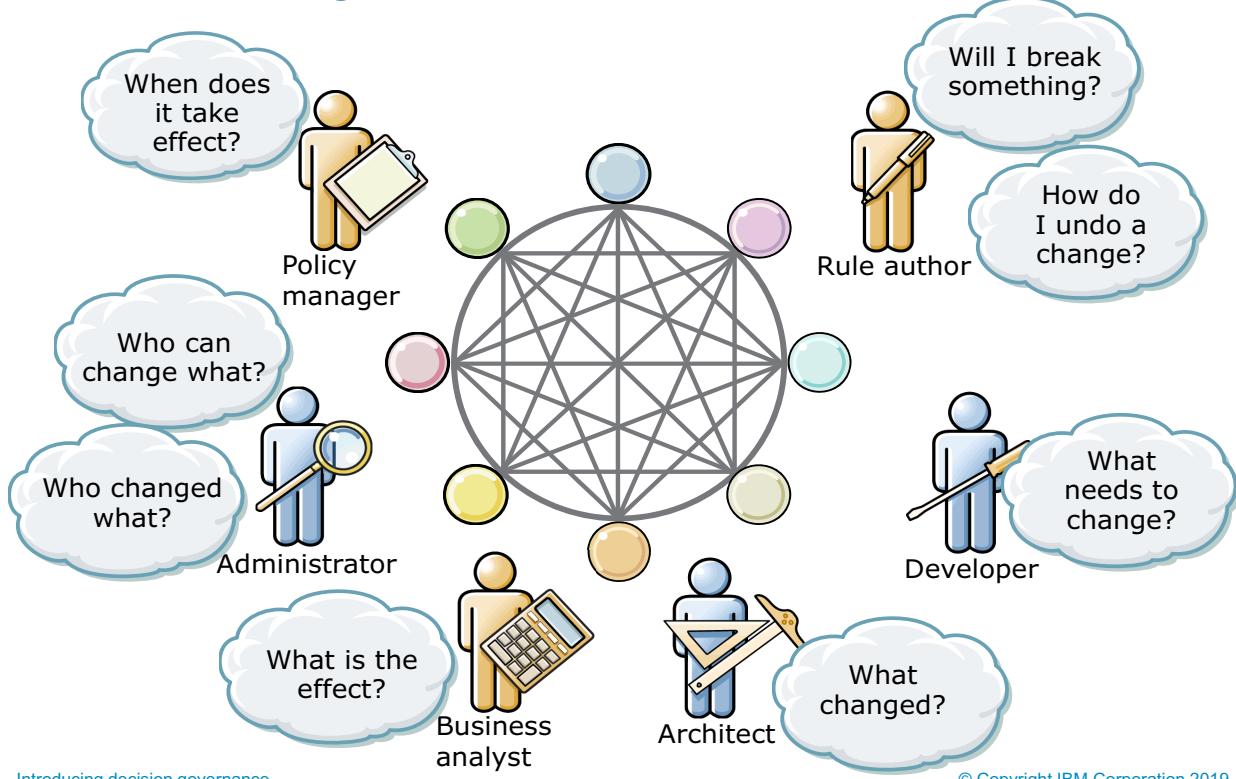
For some short-term changes, such as promotions and specials, the cycle from requirements to production is so long that updates cannot be implemented, as promotions expire before they can be deployed.

One of the main benefits of using the BRMS approach is to avoid the bottleneck by proposing these shortcuts:

- Bypass the IT group by starting with the business analyst, and push a business policy update to production.
- Business policy manager pushes updates directly to production. This direct route requires that business users have enough control of the application that they can do requirements gathering, analysis, implementation, testing, and deployment of the changes.



## Decision management increases communication



Introducing decision governance

© Copyright IBM Corporation 2019

Figure 16-7. Decision management increases communication

The agile nature of the BRMS approach brings business and technical teams together regularly. The rules provide a common vocabulary for both stakeholders, resulting in increased visibility and flow of information to more people. Increased communication encourages stakeholders to take more ownership of problems and be willing to solve them.

- Business users can consult and update the business policies.
- IT support can monitor the execution of rules, and investigate when users flag problems.
- The development team can enhance the applications with new rulesets and customizations as the application grows.
- The QA team can test the rule and event artifacts and debug the application.
- Business analysts can review the vocabulary, create rules, and simulate the effect of changes.

This approach, which facilitates concurrent enhancements and updates, puts all stakeholders close to the application and requires that they work closely together. The iterative nature of Agile Business Rule Development (ABRD) involves frequent questions that must be quickly answered and implemented.

Examples:

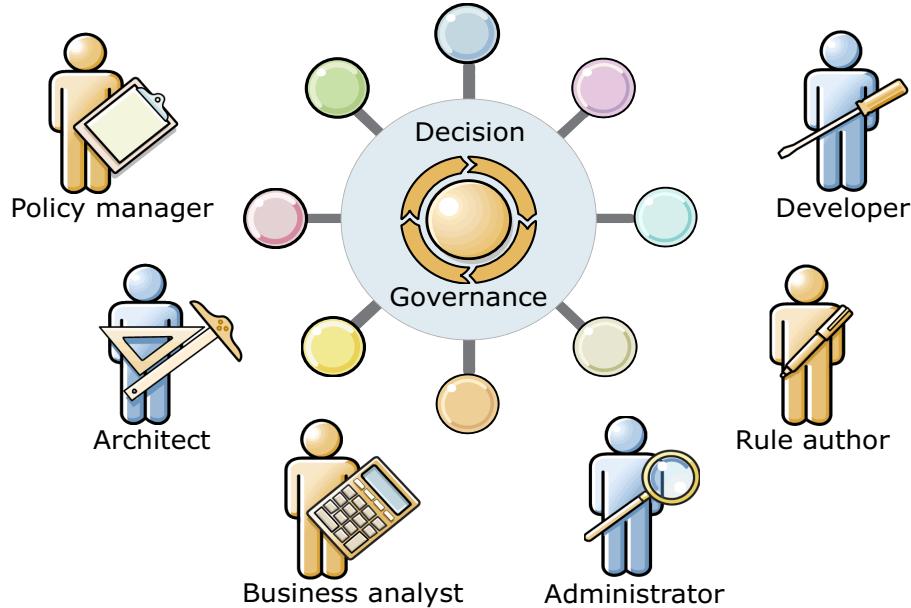
- Application lifecycle: How is this rule linked to the application?

- Permission management and security: Who is authorized to view, modify, or deploy this rule?
- Business policy applicability: Is this rule applicable in this version of the policy?
- Business safety: Are the rules that are deployed to production under control?

However, the advantages of increased agility can be lost when they are not properly governed. Conflicting interests, partial information, and office politics can lead to careless decisions when no process or authority with a holistic view of the system manages change.

## Decision governance goal

- Governance adds a layer of discipline to communication and change management



Introducing decision governance

© Copyright IBM Corporation 2019

*Figure 16-8. Decision governance goal*

Decision governance prevents the escalation of problems by providing a discipline layer to communication and change management for application maintenance.

Governance imposes a structured interaction through a set of well-defined processes.

## Definition of project governance

- A purpose
  - Charter and goals clearly stated
- A definition of stakeholders
  - With their roles and responsibilities
- A process and a set of activities
- An assignment of the roles
- An entity to manage (govern) the process
- A demonstration that the process is consistently executed

Introducing decision governance

© Copyright IBM Corporation 2019

*Figure 16-9. Definition of project governance*

While governance implementation might vary from one organization to another and from one project to another, a basic definition of governance for a project includes the elements that are listed here.

## Business Rule Management group

- Stewards of the governance processes:
  - Ensure that governance processes are properly defined and enforced
  - Address any issue that affects the project
  - Provide overall direction and advice to project managers
- Formed from among the stakeholders:
  - Business analysts and policy managers who can contribute to the governance objectives
  - Represent various stakeholders and facilitate communication between these entities (project management office, quality management, subject matter expert, and others)
- Other responsibilities:
  - Identifying business decision requirements within the organization
  - Ensuring consistency of rules across departments, functions, locations, and applications
  - Training and mentoring
  - Becoming a Center of Excellence

*Figure 16-10. Business Rule Management group*

As already noted, one of the first steps to implementing governance is to define roles that are based on your organizational structure, and to set up a dedicated team to manage the business rules.

A **business rule management group** acts as a steward of the governance processes, ensuring that governance processes are properly defined and enforced. This group can be designated as a *Rule Governance Center of Excellence*.

Members must be able to address any issue that affects the project, and provide overall direction and advice to project managers.

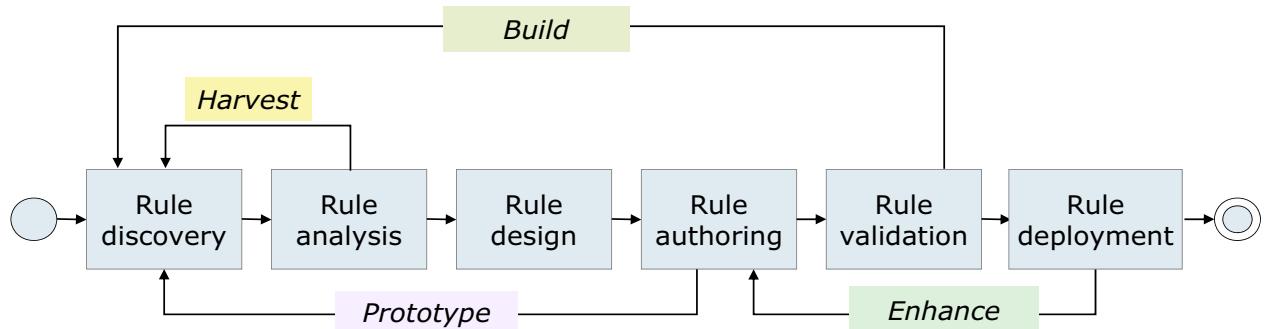
The group is formed from core team members who are involved in initial development of the application, including business analysts and policy managers, who can contribute to the governance objectives. Include members that represent various entities and facilitate communication between these entities to ensure that expectations are met.

This group might take on more rule-related responsibilities, including identifying business rule needs within the company; ensuring consistency of business rules across departments, functions, locations, and applications; and training and mentoring.

You can avoid potential issues by establishing balanced representation of the stakeholders, and ensuring that they have a clear understanding of the purpose and motivation for such a group.

# Governance and agile development

- Successful decision management projects adapt project methodology to be more iterative
    - Frequent requirement and model changes during early phases of project
    - Business owners provide early feedback on implementation
    - Respond to change instead of following a plan



## Introducing decision governance

© Copyright IBM Corporation 2019

*Figure 16-11. Governance and agile development*

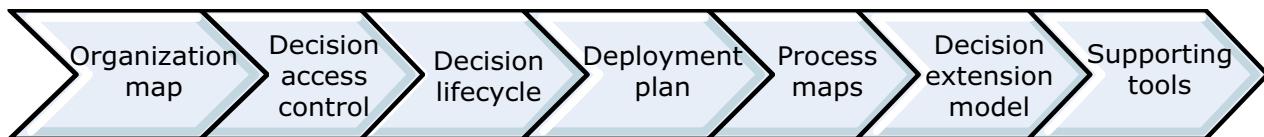
As you saw earlier in this course, the overall development of the decision follows an agile and iterative approach.

The Agile Business Rule Development (ABRD) process includes a sequence of phases. Each phase includes iterations on a set of activities, with a goal to deliver a workable set of rules (first on paper, then as a prototype in Designer, and finally, published to Decision Center).

For more information about using the ABRD methodology, see the IBM contribution to the Eclipse Process Framework Project at the Eclipse website: [www.eclipse.org/epf](http://www.eclipse.org/epf)

## Implementing governance

- The agile business rule development (ABRD) methodology defines a set of tasks to help implement governance
  - Develop the organization map
  - Assign responsibility and access control
  - Define the decision lifecycle
  - Define target deployment platforms, and who can deploy to which environment
  - Define each process with a Business Process Modeling Notation (BPMN) map
  - Implement a decision extension model
  - Design customizations to support the processes in Decision Center and Designer



[Introducing decision governance](#)

© Copyright IBM Corporation 2019

Figure 16-12. Implementing governance

- Develop the organization map

To develop the organization map, you first identify the project stakeholders, including internal and external groups, and try to understand their relationships along with how information flows between these groups.

An organization map defines roles, and can also involve setting up a team that is dedicated to decision management.

- Assign ruleset responsibility and access control

Assigning an owner to a ruleset defines who is responsible for authoring and reviewing rules within that ruleset. The owner can create a table that outlines who has permission to *create*, *read*, *update*, or *delete* rules.

- Define the decision lifecycle

Defining the lifecycle determines a status for each phase of development (such as “validated” or “deployed”), and defines who can promote a rule from one status to another.

The lifecycle forces the rule and event artifacts to go through a specific set of phases, which ensures that the artifacts pass through a testing phase. It also ensures that only certain roles have permission to do specific actions on an artifact at each phase of the lifecycle.

- Define target deployment platforms, and who can deploy to which environment

Deployment platforms are determined according to testing requirements and application requirements. Planning the rule deployment controls how the rulesets are deployed to different server platforms: test, staging, and production.

- Define each process with a Business Process Modeling Notation (BPMN) map

Processes include:

- Change management process
  - Authoring process
  - Testing process
  - Deployment process
  - Execution process
  - Retirement process
- Implement a decision extension model
  - Design customizations to support the processes in Decision Center and Designer

Operational Decision Manager supports extending the rule model and customizing Decision Center to facilitate use of metadata and custom properties.

## 16.2. Operational Decision Manager support for governance

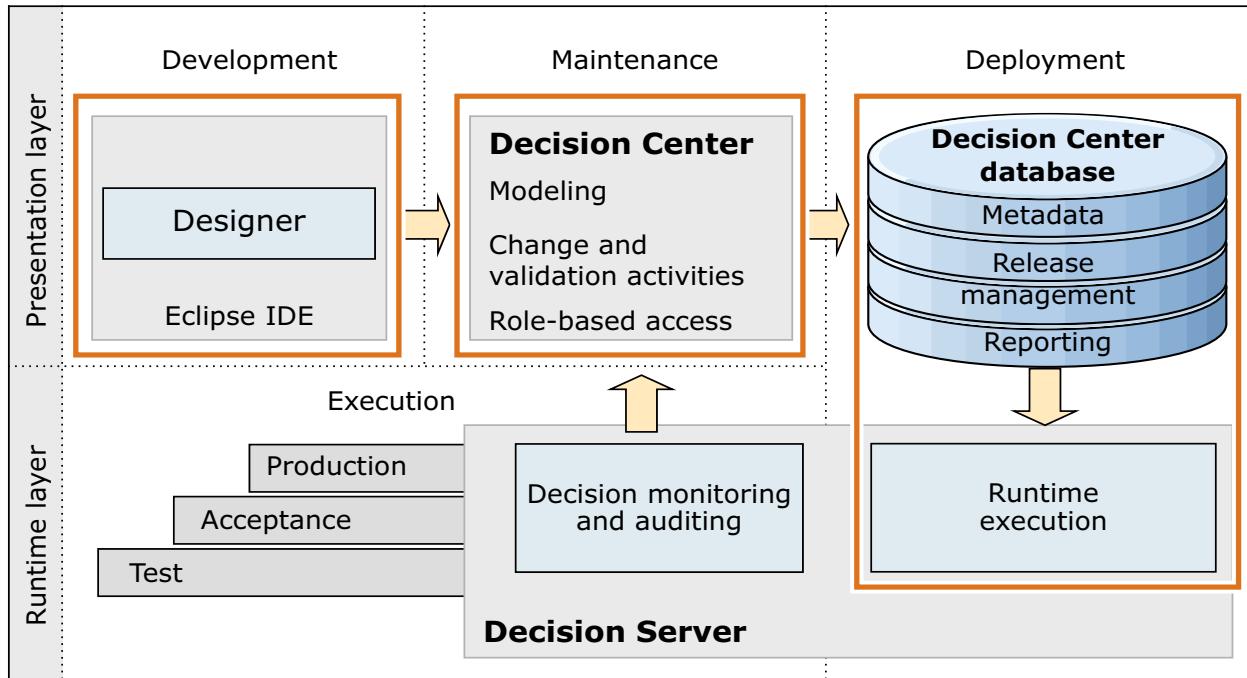
## Operational Decision Manager support for governance

Introducing decision governance

© Copyright IBM Corporation 2019

*Figure 16-13. Operational Decision Manager support for governance*

## Decision lifecycle in Operational Decision Manager (1 of 2)



[Introducing decision governance](#)

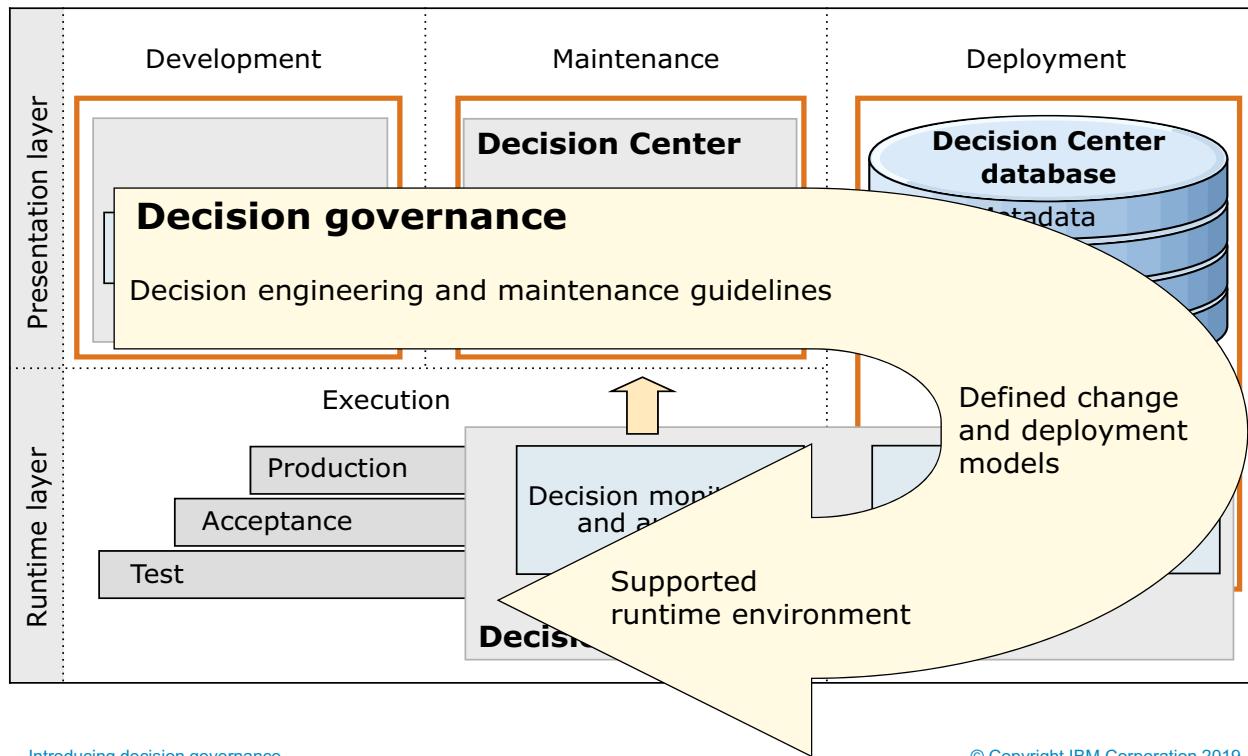
© Copyright IBM Corporation 2019

Figure 16-14. Decision lifecycle in Operational Decision Manager (1 of 2)

As seen throughout the course, ODM provides a set of mechanisms and tools to facilitate change while enforcing governance at the level of the individual decision artifacts and at the level of the overall decision.

- Developers can work in a single Eclipse-based environment.
- Decision Center provides a complete operational decision management environment for business users. You can easily manage the level of access and control to enable various stakeholders to author, edit, transition, and deploy rules that are stored in the Decision Center database.
- The Decision Center database provides integrated storage for rule and event artifacts as a central “source of truth.” A rich metadata layer is provided for decision logic that specifies all the custom properties that define how rules are used and how they interact with other rules in generating decisions.
- Decision Server provides a centralized execution environment to support streamlined change deployments.

## Decision lifecycle in Operational Decision Manager (2 of 2)



Introducing decision governance

© Copyright IBM Corporation 2019

Figure 16-15. Decision lifecycle in Operational Decision Manager (2 of 2)

Coupled with the tools and deployment cycle that are depicted here, the suggested practices that are outlined in ABRD methodology ensure project success.

The lifecycle for rule and event artifacts is implemented through status management, access control, and permissions. Notice that this fine-grained level of governance must be defined carefully so that it does not become a hindrance to the change process.

Governance at the artifact level, while enforcing a division of responsibilities for the artifact authoring task, does not provide a vision of what happens at the decision level to manage change. Therefore, on top of artifact-level governance, a decision management solution must define the change lifecycle and governance at the level of the decision itself. The goal is to define who is responsible for which task, at what phase, and in which environment. The process defines change management of a business policy update, from its initial request by the policy manager to its deployment in the production environment, through to evaluation of the newly deployed business decision implementation. Decision changes follow a cycle of *define, deploy, measure, and update*.

## Discussion

How can you apply governance?

Introducing decision governance

© Copyright IBM Corporation 2019

Figure 16-16. Discussion

Before continuing, take a moment to consider what you learned in this unit by answering the following questions.

1. Which tools and features did you work with during this course?
2. Considering the role or roles that you play in the decision management solution, which tools do you expect to work with?
  - Are you developing both rule and event artifacts?
  - Are you updating and authoring new business rules or event rules?
  - Are you involved in deployment to test or production servers and monitoring execution?
3. Are you working with business rules, events, or both?
  - How might you store and manage these artifacts and assets?
  - Are you responsible for synchronizing artifacts across business and technical environments?
4. Based on what you learned during this course, how do you anticipate the application of governance principles and lifecycle management through the tools:
  - At the artifact level?

- At the decision level?

## 16.3. Decision governance framework

## Decision governance framework

Introducing decision governance

© Copyright IBM Corporation 2019

*Figure 16-17. Decision governance framework*

## Decision governance framework overview

- A well-defined decision change process that is supported by Decision Center tools
  - User roles and permissions define who can do what
- Rule Designer
  - Publish rule projects as decision services
- Decision Center Business console
  - Work with decision service releases
  - Create, assign, and complete change activities
  - Create, assign, and complete validation activities
  - Perform validation tasks (tests and simulations)
  - Approve change and validation activities
  - Approve releases for deployment
  - Deploy completed releases to production

[Introducing decision governance](#)

© Copyright IBM Corporation 2019

Figure 16-18. Decision governance framework overview

The decision governance framework uses tools in Rule Designer and Decision Center to provide a ready-to-use, prescriptive approach to change management and rule governance. It helps business users manage, report, and govern changes to rules and decisions.

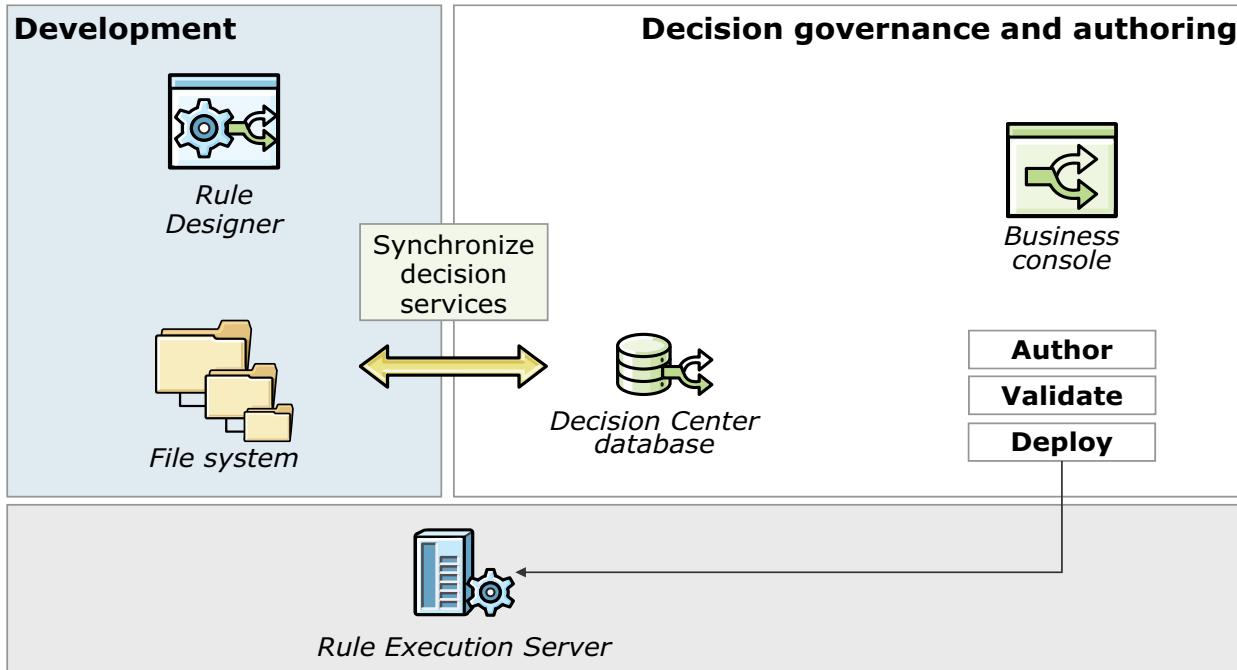
The decision governance framework is based on decision service *releases*, which follow a well-defined change process that is supported by Decision Center tools. Tasks that business users complete depend on user roles, such as rule author or tester.

For effective workflow control, you set permissions for each phase to determine who can work with the rule during a particular phase, and who can promote an artifact from one phase to another.

Governance framework includes:

- Decision service releases
- Change activities
- Validation activities (testing)
- Deployment of releases

## Recall: Operational Decision Manager tools



Introducing decision governance

© Copyright IBM Corporation 2019

Figure 16-19. Recall: Operational Decision Manager tools

The decision governance framework in Operational Decision Manager encompasses the following tasks:

- **Synchronizing**

IT users use Rule Designer to publish a set of rule projects as a decision service that business users can access through Business console.

- **Authoring**

Business analysts and rule authors use Business console to create **change activities** and author rules within a release.

- **Validating**

Policy managers use Business console to create **validation activities** to track and manage test plans for a release. Assigned testers create and run test suites and simulations in Business console.

- **Deploying**

After all change and validation activities are complete and approved, policy managers approve the completion of the release. The release is ready for deployment. Release owners or administrators can create deployment configurations and deploy decision services from Business console to Rule Execution Server.

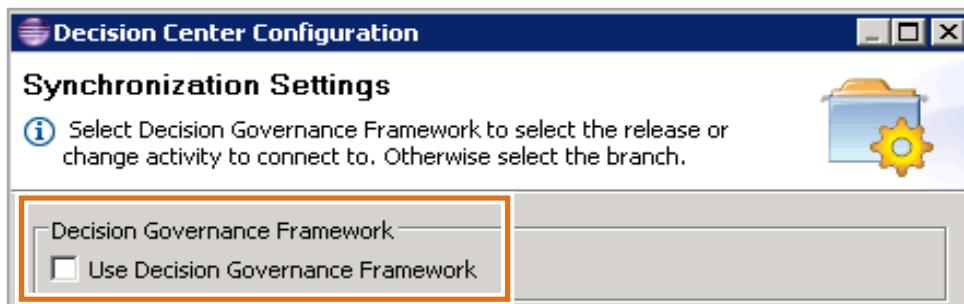
- **Administering**

Use Business console to manage user access and permissions. Additional administrative tools in Enterprise console are used to manage projects in the database. The Decision Center database provides snapshot and version features that support the auditing and rollback of business rules.



## Publishing decision services from Rule Designer

- When you publish a decision service to Decision Center, connect to Decision Center from the top-level main rule project
- Select **Use Decision Governance Framework** check box



[Introducing decision governance](#)

© Copyright IBM Corporation 2019

*Figure 16-20. Publishing decision services from Rule Designer*

To publish the decision service, you connect to Decision Center from the main rule project.

After the connection is established, you select the **Use Decision Governance Framework** option to enable governance in Decision Center. During the publish operation, you see the list of dependent rule projects in the decision service that are also published.

The decision service is published as an initial release named `InitialRelease` that is protected from being modified. In the Business console, a release is created to begin editing.

## Decision Center: Governance in Business console

Use the governance framework in a decision service release

- Assign and complete change activities
- Assign and complete validation activities
- Approve work that is performed on activities
- Assign and complete deployment tasks
- Perform validation tasks
- Deploy new decision service releases

[Introducing decision governance](#)

© Copyright IBM Corporation 2019

*Figure 16-21. Decision Center: Governance in Business console*

Through Decision Center, business users access decision insight capabilities that help them understand how and where change affects decisions within operational systems. They can also simulate how decisions affect the outcomes of transactions, orders, customer interactions, or processes. Role-based access control supports concurrent access of decision artifacts while enforcing security.

Decision Center provides several lifecycle control features:

- Permission management: User roles and associated permissions determine the ability to create, change, or see rules
- Version control and history
  - Every version is kept in the database (even after deletion)
  - Ensures traceability
- Workflow control through rule status properties
  - Depending on permissions, users can change the status property as the rule passes through the lifecycle phases
- Deployment to Rule Execution Server with baseline management to track deployment history (auditability)

## States and user roles

- Governance in Decision Center is based on:
  - The states of decision service releases and activities
  - The user roles of participants who work on these releases and activities
- States
  - The state of a decision service release or activity determines what work can be done and by whom
  - Transition from one state to another can generate automatic snapshots or merges
- Roles
  - User roles have permissions for specific tasks that can be done within the context of a release

*Figure 16-22. States and user roles*

The state of a release or activity can be one of:

- **In Progress**
  - **Ready for Approval**
  - **Complete**
- **Canceled**
- **Rejected**

User roles include the following categories.

- All releases and activities have an **owner** and an **approver** role
- Change activities also have an **author** role
- Validation activities have a **tester** role
- Users who have administrator privileges can carry out the user operations of all roles

## Releases

- Captures and traces all changes to a decision service that are related to a purpose and period in time
  - A **purpose** is a set of business-driven goals
  - A **period in time** has a beginning date and an end date
- Releases cannot be edited directly
  - Changes are managed through governance framework change activities and validation activities
- Work on the release ends when it is completed, approved, and deployed

[Introducing decision governance](#)

© Copyright IBM Corporation 2019

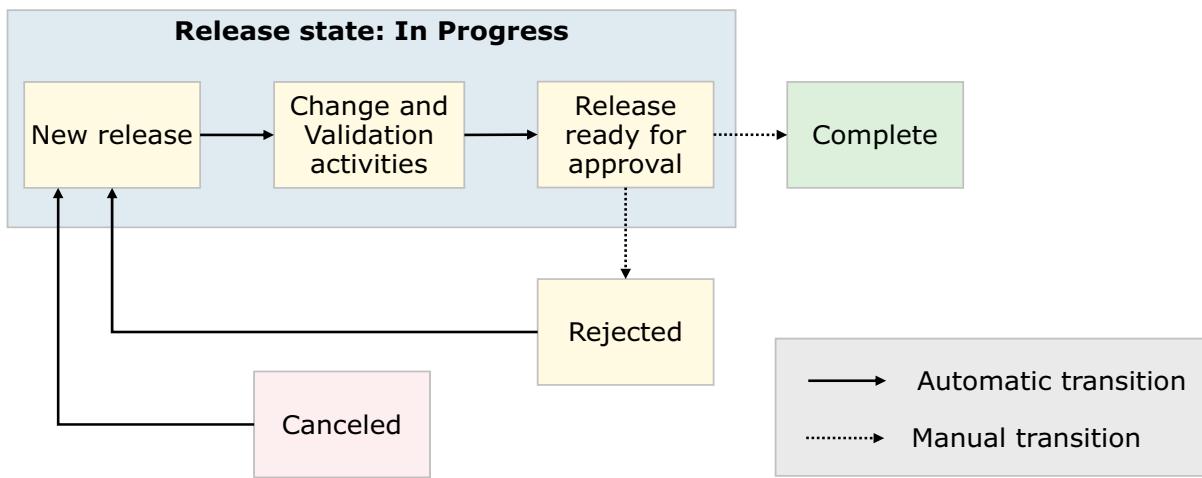
Figure 16-23. Releases

A release has the following tabs.

- **Activities:** The **Activities** tab lists the change and validation activities for the decision service release.
- **Rules:** The **Rules** tab shows the current state of the rules that are contained in Training Release. You can view the content of these rules, but you cannot edit them directly from the release branch.
- **Tests:** The **Tests** tab lists the test suites that are created for the decision service and links to test results.
- **Simulations:** From the **Simulations** tab, you can run simulations in the change and validation activities of a decision service release. Simulations can help determine how changes to business rules or data affect the results of the business rule application before it is deployed.
- **Deployments:** The **Deployments** tab lists the deployment configurations that are available for the decision service. Changes from change activities can be deployed to non-production servers. However, only completed releases can be deployed to a production environment.
- **Snapshots:** The **Snapshots** tab lists the decision service snapshots, which capture the state of a branch at a previous moment in time. Snapshots can be consulted, compared, and restored, but not edited.

## Release governance

- Release governance involves management of the overall release lifecycle
  - A release is the container for rule content that is going to be deployed to production
  - Rules are maintained through change and validation activities within a release



Introducing decision governance

© Copyright IBM Corporation 2019

Figure 16-24. Release governance

The user who creates a release:

- Sets the owner of the release
- Sets the goals of the release
- Sets the date when the release must be completed
- Assigns one or more participants as the approver of the release

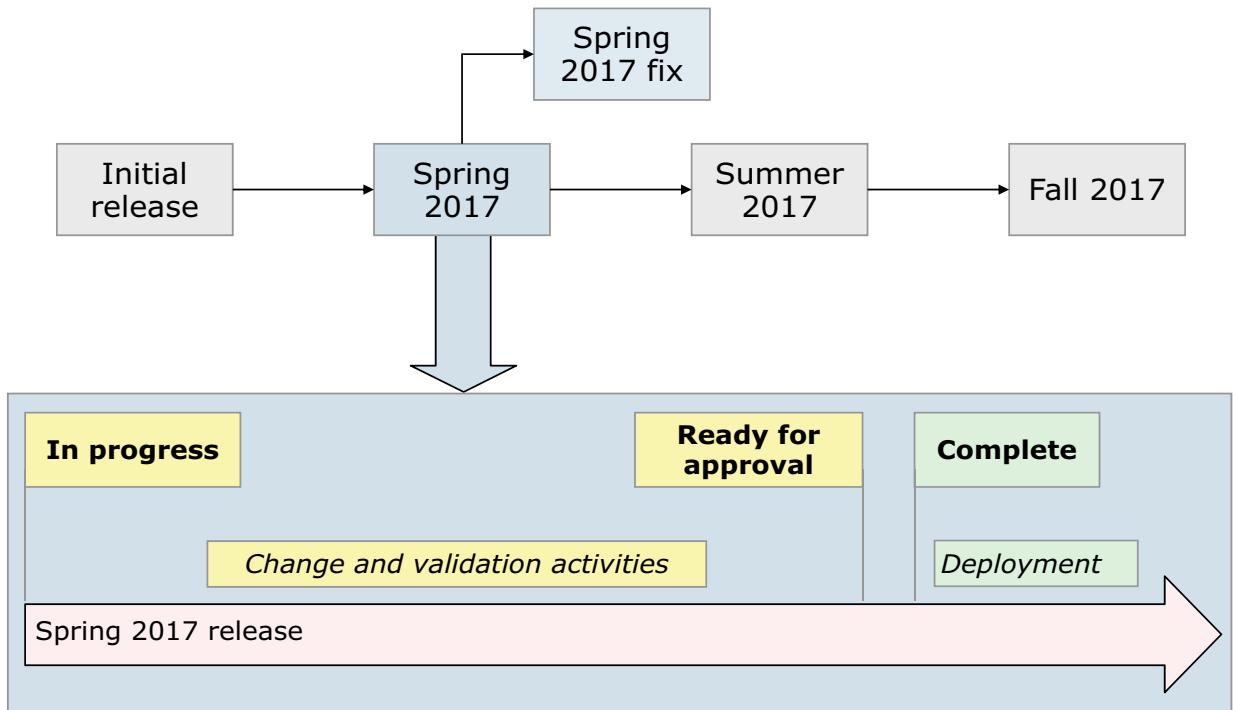
When a release is created, Decision Center automatically creates a snapshot. Some release-related tasks are manual. When the release is in the **In Progress** state, the owner can:

- Change the owner of the release
- Change the goals of the release
- Change the due date of the release
- Create change and validation activities

After all release activities are complete:

- The release owner changes the state of the release to **Ready for Approval**
- Approvers can then approve or reject the changes to the release

## Release sequence



Introducing decision governance

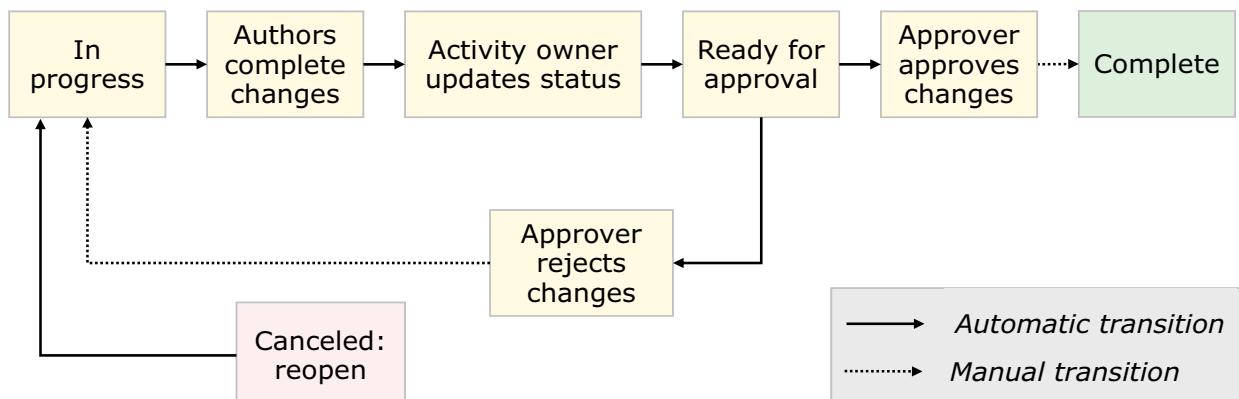
© Copyright IBM Corporation 2019

*Figure 16-25. Release sequence*

An initial release is automatically created when a decision service is published from Rule Designer. The content of the new release is based on the content of another complete release.

## Change activity governance

- Changes to the release are initiated through the creation and completion of change activities
    - When a change activity is created, Decision Center takes a snapshot to record the starting state of the activity
    - Change activities also have states and transitions that are predefined



## Introducing decision governance

© Copyright IBM Corporation 2019

*Figure 16-26. Change activity governance*

Rule authors are responsible for editing and updating the rules so that they align with the goals of the release. After rule authors finish their work, they can change *their* status to **Finished**.

When all the authors finish their work, the owner of the change activity sets the state of the change activity to **Ready for Approval**.

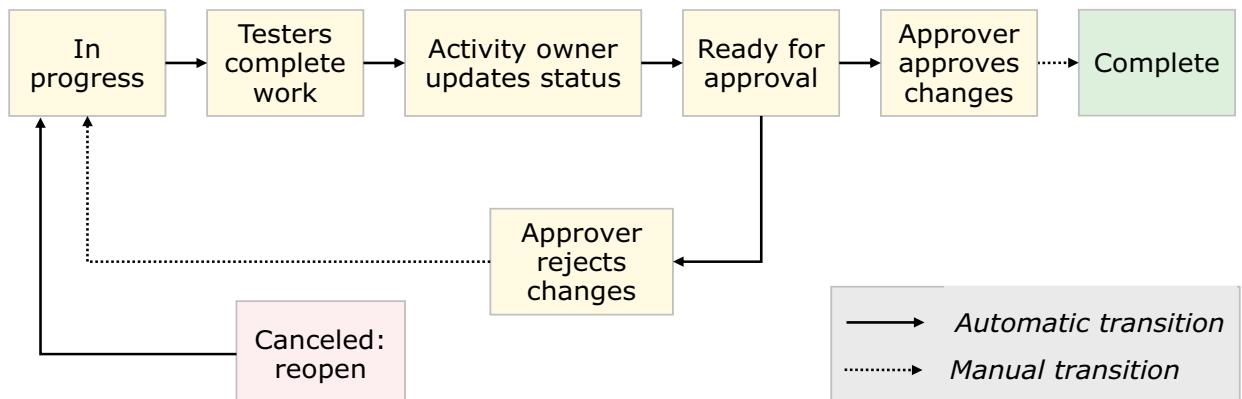
Then, the assigned approvers approve or reject the changes. If the change activity was not assigned approvers by the release owner, the change activity is automatically approved.

After all approvers approve the change activity, Decision Center takes another snapshot to record the state of the rules. Decision Center then merges the changes back into the release. The change activity is then marked **Complete**.

If a change activity is rejected, it returns to the state of **In Progress**.

## Validation activity governance

- After change activities are completed, the release is validated through the creation and completion of validation activities
  - Policy managers create validation activities to track and manage test plans and test results for release to production
  - Validation tasks include running tests and simulations



Introducing decision governance

© Copyright IBM Corporation 2019

Figure 16-27. Validation activity governance

Testers run different tests and simulations that are aimed at validating a release, and note the result in the test plan. After testers complete their work, and all the change activities of the release are **Complete**, testers change their status to **Finished**.

The owner of the validation activity sets the validation activity state to **Ready for Approval**, at which point the approvers approve or reject the activity.

After all the approvers approve the validation activity, Decision Center sets the state of the validation activity to **Complete**.

## Release deployment

- After all activities in a release are complete, the release owner approves the release itself
- Approved releases can be deployed from Business console
- Only completed releases can be deployed to production
  - Change activities and branches can be deployed, but only to non-production environments

Introducing decision governance

© Copyright IBM Corporation 2019

Figure 16-28. Release deployment

The **Deployments** tab lists the deployment configurations that are available for the decision service. Changes from change activities can be deployed to non-production servers. However, only completed releases can be deployed to a production environment.

When the decision service is deployed, a deployment snapshot is created. Snapshots can be consulted, compared, and restored, but not edited.

## Unit summary

- Explain governance issues and good practices
- Identify Operational Decision Manager features that support decision governance
- Describe how to implement the decision governance framework

*Figure 16-29. Unit summary*

## Review questions

1. True or False: Rules play a dual role as business assets that represent business logic and as part of the actual software that runs on enterprise data.
2. Governance encompasses which of these tasks? Select all that apply.
  - A. Defining expectations and assigning responsibilities
  - B. Establishing efficient collaboration between the business and IT teams
  - C. Selecting a group of productive developers to be in charge of business rule management and reporting their decisions to business stakeholders
3. True or False: Implementing decision governance is outside the scope of the IT department.
4. True or False: With the decision governance framework, business users cannot deploy decision services from Business console.

[Introducing decision governance](#)

© Copyright IBM Corporation 2019

Figure 16-30. Review questions

Write your answers here:

- 1.
- 2.
- 3.
- 4.



## Review answers (1 of 2)

1. True or False: Rules play a dual role as business assets that represent business logic and as part of the actual software that runs on enterprise data.  
The answer is True.
  
2. Governance encompasses which of these tasks? Select all that apply.
  - A. Defining expectations and assigning responsibilities
  - B. Establishing efficient collaboration between the business and IT teams
  - C. Selecting a group of productive developers to be in charge of business rule management and reporting their decisions to business stakeholders

The answer is A and B. C is incorrect, Include business and developer stakeholders to ensure balanced representation for decision making.



Figure 16-31. Review answers (1 of 2)

## Review answers (2 of 2)

3. True or False: Implementing decision governance is outside the scope of the IT department.

**The answer is False.** Implementing rule governance must include collaboration from both business and IT stakeholders to provide an organizational framework that instills confidence in all stakeholders.



4. True or False: With the decision governance framework, business users cannot deploy decision services from Business console.

**The answer is False.** Business users with administrative permissions can create or edit deployment configurations and deploy decision services from Business console.

Figure 16-32. Review answers (2 of 2)

---

# Unit 17. Course summary

## Estimated time

00:30

## Overview

This unit summarizes the course and provides information for future study.

## Unit objectives

- Explain how the course met its learning objectives
- Access the IBM Training website
- Identify other IBM Training courses that are related to this topic
- Locate appropriate resources for further study

[Course summary](#)

© Copyright IBM Corporation 2019

*Figure 17-1. Unit objectives*

## Course objectives

- Describe the benefits of implementing a decision management solution with Operational Decision Manager
- Identify the key user roles that are involved in designing and developing a decision management solution, and the tasks that are associated with each role
- Describe the development process of building a business rule application and the collaboration between business and development teams
- Set up and customize the Business Object Model (BOM) and vocabulary for rule authoring
- Implement the Execution Object Model (XOM) that enables rule execution
- Orchestrate rule execution through ruleflows
- Author rule artifacts to implement business policies

[Course summary](#)

© Copyright IBM Corporation 2019

*Figure 17-2. Course objectives*

## Course objectives

- Debug business rule applications to ensure that the implemented business logic is error-free
- Set up and customize testing and simulation for business users
- Package and deploy decision services to test and production environments
- Integrate decision services for managed execution within an enterprise environment
- Monitor and audit execution of decision services
- Work with Operational Decision Manager features that support decision governance

[Course summary](#)

© Copyright IBM Corporation 2019

*Figure 17-3. Course objectives*

## To learn more on the subject

- IBM Training website:  
[www.ibm.com/training](http://www.ibm.com/training)
- IBM Knowledge Center for Operational Decision Manager V8.10  
[www.ibm.com/support/knowledgecenter/en/SSQP76 8.10.0](http://www.ibm.com/support/knowledgecenter/en/SSQP76_8.10.0)
- IBM Developer  
[Developer forum \(dw Answers\)](#)
- IBM Developer recommended reading  
<https://developer.ibm.com/odm/docs/recommended-reading>
- IBM Developer Videos  
<https://developer.ibm.com/odm/videos/>

Course summary

© Copyright IBM Corporation 2019

Figure 17-4. To learn more on the subject



## Earn an IBM Badge

- Completing this course prepares you to take an IBM Badge test
- Use IBM Badges to share verified proof of your IBM credentials
- Find your Badge test on this site:
  - <https://www.ibm.com/services/learning/ites.wss/zz-en?pageType=badgesearch>
  - Search: *IBM Operational Decision Manager V8.10 Developer*



Course summary

© Copyright IBM Corporation 2019

Figure 17-5. Earn an IBM Badge

## Enhance your learning with IBM resources

*Keep your IBM Cloud skills up-to-date*

- IBM offers resources for:
  - Product information
  - Training and certification
  - Documentation
  - Support
  - Technical information



- To learn more, see the IBM Cloud Education Resource Guide:
  - [www.ibm.biz/CloudEduResources](http://www.ibm.biz/CloudEduResources)

Figure 17-6. Enhance your learning with IBM resources

## Unit summary

- Explain how the course met its learning objectives
- Access the IBM Training website
- Identify other IBM Training courses that are related to this topic
- Locate appropriate resources for further study

[Course summary](#)

© Copyright IBM Corporation 2019

*Figure 17-7. Unit summary*



## Course completion

**You have completed this course:**

Developing Rule Solutions in IBM Operational Decision Manager V8.9.2

**Any questions?**



[Course summary](#)

© Copyright IBM Corporation 2019

*Figure 17-8. Course completion*

# Appendix A. List of abbreviations

|             |                                     |
|-------------|-------------------------------------|
| <b>ABRD</b> | Agile Business Rule Development     |
| <b>API</b>  | application programming interface   |
| <b>ATM</b>  | automatic teller machine            |
| <b>B2X</b>  | BOM to XOM mapping                  |
| <b>BA</b>   | business analyst                    |
| <b>BAL</b>  | Business Action Language            |
| <b>BEP</b>  | business event processing           |
| <b>BOM</b>  | business object model               |
| <b>BPM</b>  | business process management         |
| <b>BPMN</b> | Business Process Modeling Notation  |
| <b>BQL</b>  | Business Query Language             |
| <b>BRM</b>  | business rule management            |
| <b>BRMS</b> | business rule management system     |
| <b>CICS</b> | Customer Information Control System |
| <b>CPU</b>  | central processing unit             |
| <b>CRM</b>  | customer relationship management    |
| <b>CVS</b>  | Concurrent Versions System          |
| <b>Db</b>   | Database                            |
| <b>DHCP</b> | Dynamic Host Configuration Protocol |
| <b>DVS</b>  | Decision Validation Services        |
| <b>DW</b>   | Decision Warehouse                  |
| <b>EAR</b>  | enterprise archive                  |
| <b>EE</b>   | Enterprise Edition (Java EE)        |
| <b>EJB</b>  | Enterprise JavaBeans                |
| <b>ERC</b>  | edition revision code               |
| <b>ESB</b>  | enterprise service bus              |
| <b>GRL</b>  | Guided Rule Language                |
| <b>GUI</b>  | graphical user interface            |
| <b>HTDS</b> | hosted transparent decision service |
| <b>HTTP</b> | Hypertext Transfer Protocol         |

|                |   |
|----------------|---|
| <b>IBM</b>     | International Business Machines Corporation |
| <b>ICP</b>     | IBM Cloud Private                           |
| <b>IDE</b>     | integrated development environment          |
| <b>IP</b>      | Internet Protocol                           |
| <b>IQL</b>     | Intellirule Query Language                  |
| <b>IRL</b>     | ILOG Rule Language                          |
| <b>IT</b>      | information technology                      |
| <b>JAR</b>     | Java archive                                |
| <b>Java EE</b> | Java Platform, Enterprise Edition           |
| <b>Java SE</b> | Java Platform, Standard Edition             |
| <b>JAXB</b>    | Java Architecture for XML Binding           |
| <b>JCA</b>     | Java EE Connector Architecture              |
| <b>JDBC</b>    | Java Database Connectivity                  |
| <b>JDK</b>     | Java Development Kit                        |
| <b>JMS</b>     | Java Message Service                        |
| <b>JMX</b>     | Java Management Extension                   |
| <b>JNDI</b>    | Java Naming and Directory Interface         |
| <b>JRE</b>     | Java Runtime Environment                    |
| <b>JSON</b>    | JavaScript Object Notation                  |
| <b>JVM</b>     | Java virtual machine                        |
| <b>KPI</b>     | key performance indicator                   |
| <b>LAN</b>     | local area network                          |
| <b>LOB</b>     | line of business                            |
| <b>MBean</b>   | management bean                             |
| <b>MDB</b>     | message-driven bean                         |
| <b>ODM</b>     | Operational Decision Manager                |
| <b>POJO</b>    | plain old Java object                       |
| <b>POM</b>     | Project Object Model                        |
| <b>POS</b>     | Point of sale                               |
| <b>PVU</b>     | Processor Value Unit                        |
| <b>QA</b>      | quality assurance                           |
| <b>RAR</b>     | resource adapter archive                    |
| <b>RES</b>     | Rule Execution Server                       |
| <b>REST</b>    | Representational State Transfer             |

|             |   |
|-------------|---|
| <b>RFID</b> | radio frequency identification  |
| <b>RMI</b>  | Remote Method Invocation  |
| <b>RQL</b>  | Rule Query Language   |
| <b>SCA</b>  | Service Component Architecture  |
| <b>SCC</b>  | source code control   |
| <b>SDK</b>  | software development kit  |
| <b>SDO</b>  | Service Data Object   |
| <b>SE</b>   | Standard Edition (Java SE)  |
| <b>SME</b>  | subject matter expert   |
| <b>SOA</b>  | service-oriented architecture   |
| <b>SOAP</b> | A lightweight, XML-based protocol for exchanging information in a decentralized, distributed environment. SOAP can be used to query and return information and invoke services across the Internet. |
| <b>SPSS</b> | Statistical Product and Service Solutions   |
| <b>SSN</b>  | Social Security Number  |
| <b>SSP</b>  | Scenario Service Provider   |
| <b>TCP</b>  | Transmission Control Protocol   |
| <b>TRL</b>  | Technical Rule Language   |
| <b>UI</b>   | user interface  |
| <b>UML</b>  | Unified Modeling Language   |
| <b>URI</b>  | Uniform Resource Identifier   |
| <b>URL</b>  | Uniform Resource Locator  |
| <b>VIN</b>  | vehicle identification number   |
| <b>WADL</b> | Web Application Description Language  |
| <b>WAR</b>  | web archive   |
| <b>WSDL</b> | Web Services Description Language   |
| <b>WSE</b>  | Workgroup Server Edition  |
| <b>WTDS</b> | web transparent decision service  |
| <b>XML</b>  | Extensible Markup Language  |
| <b>XOM</b>  | execution object model  |
| <b>XSD</b>  | XML Schema Definition   |
| <b>XU</b>   | Execution Unit  |
| <b>YAML</b> | YAML Ain't Markup Language  |
| <b>z/OS</b> | Z Series Operating System   |



---

# Appendix B. Appendix: IBM ODM on Cloud

## Estimated time

00:15

## Overview

This unit describes Operational Decision Manager on Cloud.

## Introduction to IBM ODM on Cloud

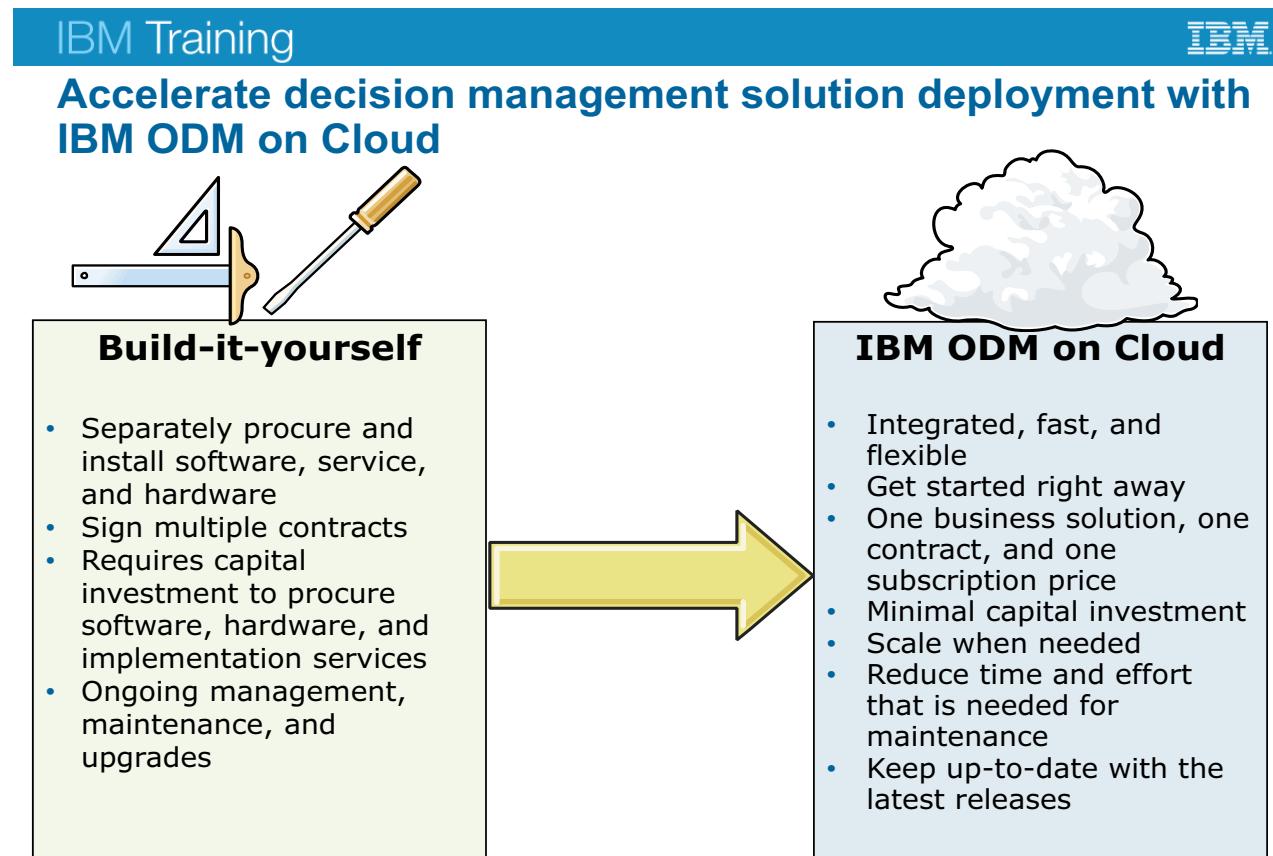
- Enterprise-grade ODM cloud service for development, testing, and production
- Cloud-based, collaborative, and role-based environment
  - Capture, automate, and manage frequently occurring, repeatable rules-based business decisions
- Ready-to-use development, test, and production environments are available
- Monthly subscription plans
- Available exclusively on IBM Cloud infrastructure
- Managed by IBM
- Artifacts that are created with IBM ODM on Cloud are compatible with IBM ODM on-premises product

Figure B-1. Introduction to IBM ODM on Cloud

## IBM ODM on Cloud

- Targets born-on-the-cloud projects and hybrid cloud scenarios
- Born-on-the-cloud project:
  - Development, testing, and production in the cloud
  - Prefer cloud solutions to on-premises solutions
  - Urgency for implementation
- Pilot project
  - Proving business value
  - Try a newer version of IBM Operational Decision Manager
- Development
  - Organization can manage the production solution on-premises, but needs to get started quickly to meet go-live dates

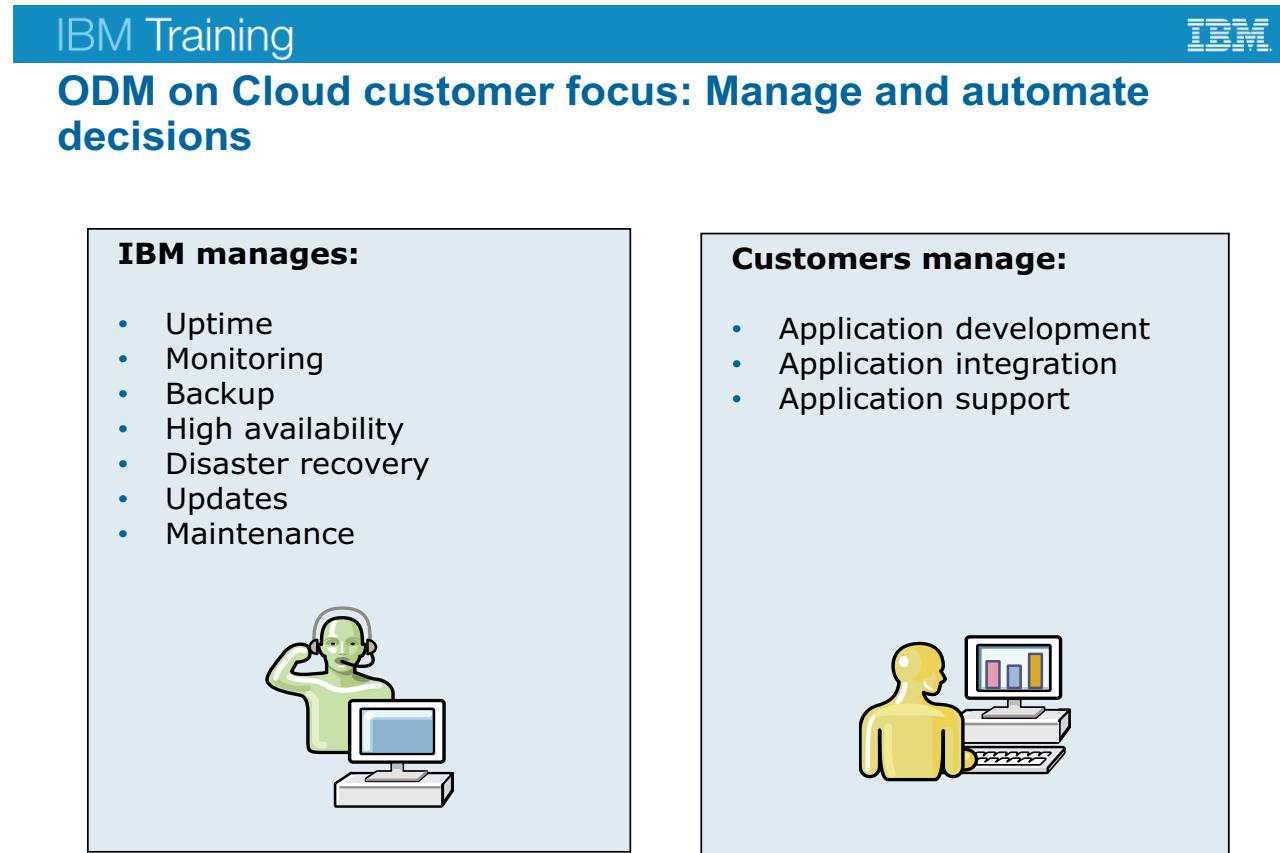
Figure B-2. IBM ODM on Cloud



Appendix: IBM ODM on Cloud

© Copyright IBM Corporation 2019

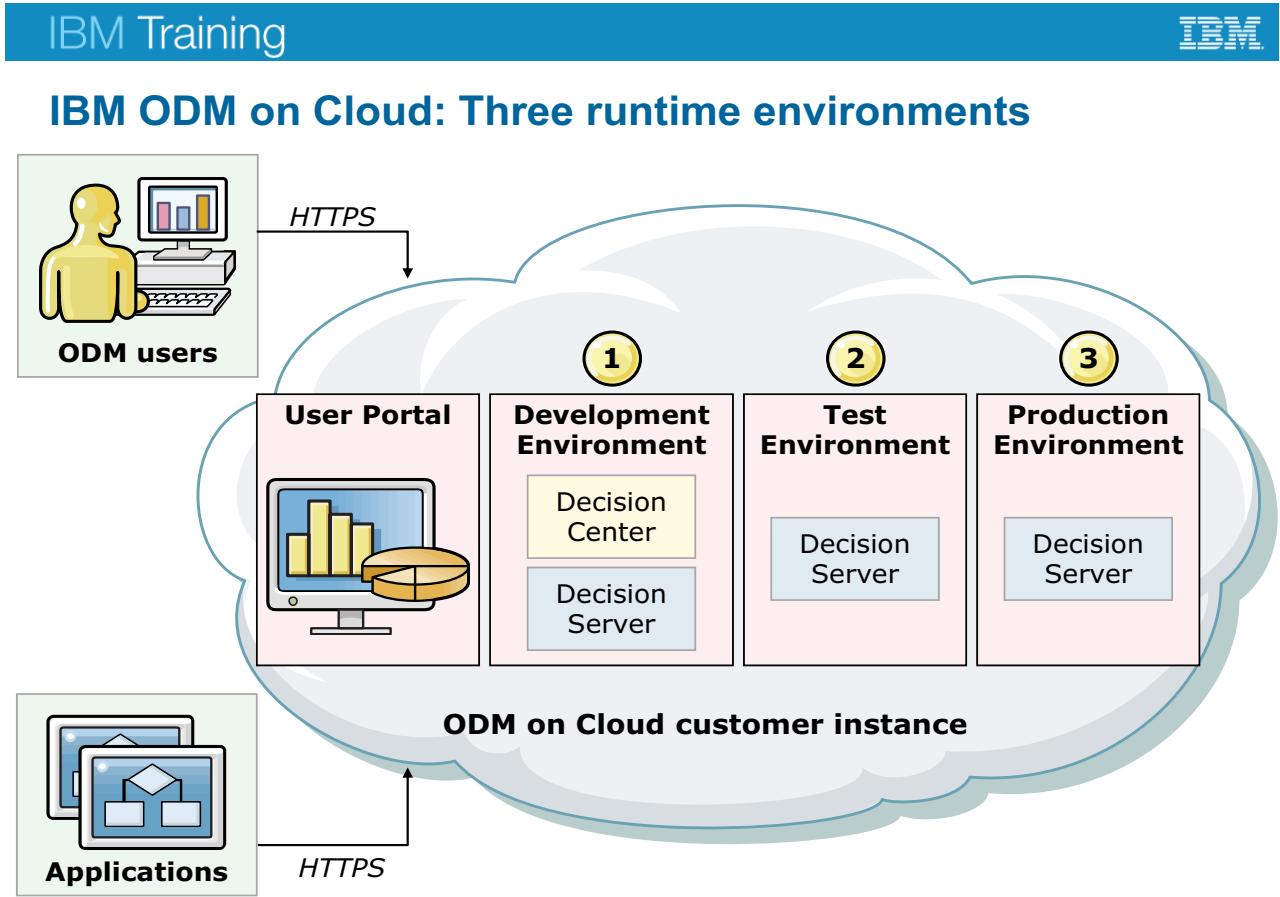
Figure B-3. Accelerate decision management solution deployment with IBM ODM on Cloud



Appendix: IBM ODM on Cloud

© Copyright IBM Corporation 2019

Figure B-4. ODM on Cloud customer focus: Manage and automate decisions



Appendix: IBM ODM on Cloud

© Copyright IBM Corporation 2019

*Figure B-5. IBM ODM on Cloud: Three runtime environments*

IBM ODM on Cloud provides three runtime environments for decision management:

1. Development
2. Test
3. Production

In this diagram:

- **ODM users** include developers, business analysts, business users, and rule authors who access Rule Designer, Decision Center, and the various user consoles.
- **Applications** are applications that call deployed decision services.

# IBM Training



## Workflow

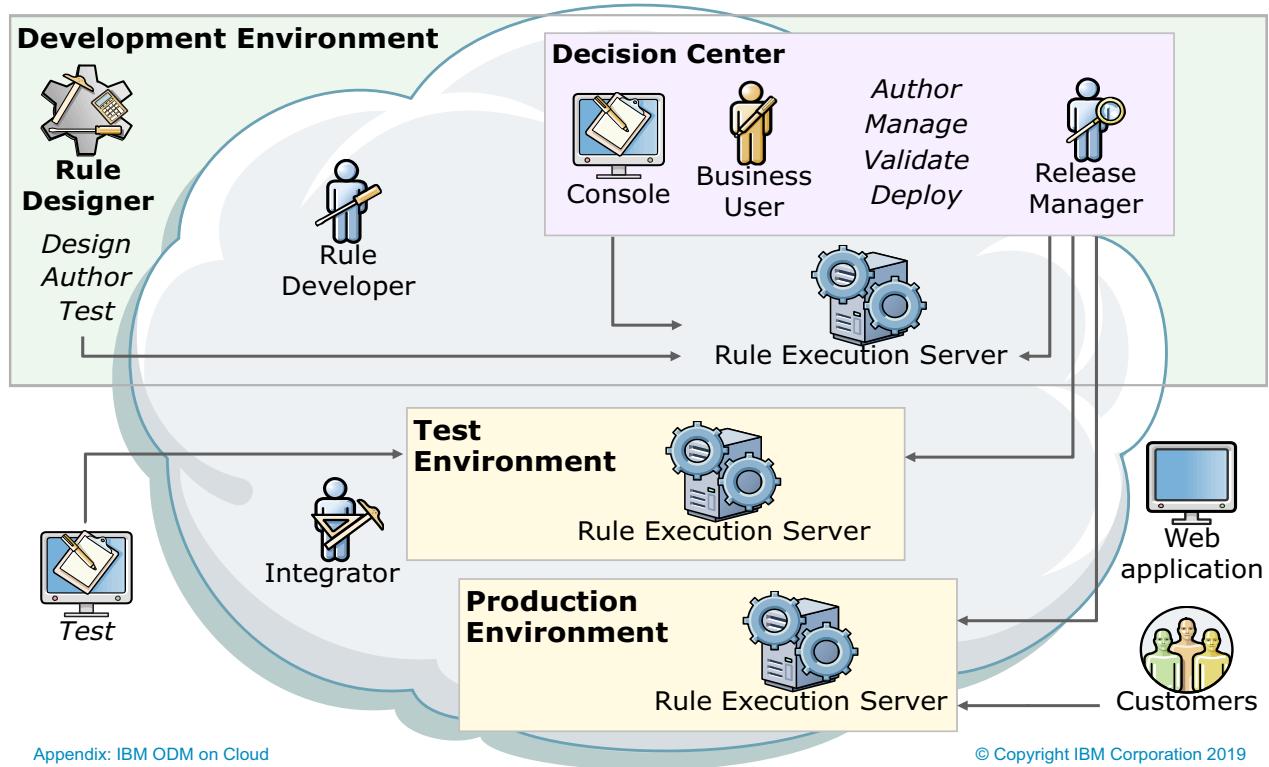


Figure B-6. Workflow

This diagram illustrates how the predefined user roles in IBM ODM on Cloud interact with the product components during the lifecycle of a business rules application.

IBM Training 

## IBM ODM on Cloud free trial

- Free trial for IBM ODM on Cloud is available
- Go to the following website and click **Try for free** to sign up:  
[www.bpm.ibmcloud.com/odm](http://www.bpm.ibmcloud.com/odm)



Appendix: IBM ODM on Cloud

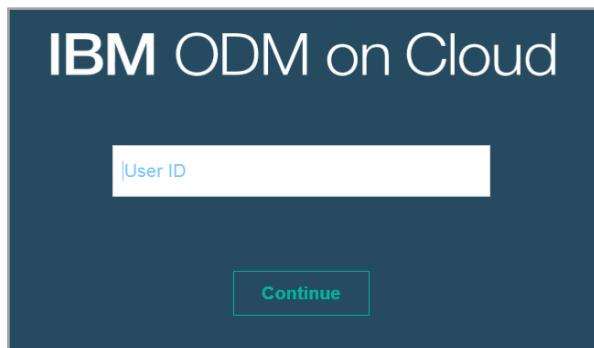
© Copyright IBM Corporation 2019

Figure B-7. IBM ODM on Cloud free trial



## Activating access and logging in to IBM ODM on Cloud

- Welcome email includes the following information:
  - Link to activate ODM on Cloud access
  - Link to ODM on Cloud instance
- Activation link is tied to a specific email
- After activating access, you can log in to your ODM on Cloud instance



Appendix: IBM ODM on Cloud

© Copyright IBM Corporation 2019

Figure B-8. Activating access and logging in to IBM ODM on Cloud

The screenshot shows the IBM ODM on Cloud user portal interface. At the top, there's a blue header bar with the text "IBM Training" on the left and the IBM logo on the right. Below the header is a main content area with a white background and a dark blue header bar labeled "APPLICATIONS". This section is divided into three main environment sections: "Development Environment", "Test Environment", and "Production Environment".

- Development Environment:** Contains four items:
  - Rule Designer**: Build a decision service to author and run the business rules that implement your business decisions. Includes "Download" and "More info" buttons.
  - Decision Center Business console**: Author, test, govern and deploy decision services. Includes "Launch" and "More info" buttons.
  - Decision Center Enterprise console**: Perform occasional administration or advanced management activities. Includes "Launch" and "More info" buttons.
  - Rule Execution Server console**: Manage and monitor decision services deployed from Rule Designer or Decision Center as part of your development activities. Includes "Launch" and "More info" buttons.
- Test Environment:** Contains one item:
  - Rule Execution Server console**: Monitor decision services deployed from Decision Center used for performance, system, and integration testing activities. Includes "Launch" and "More info" buttons.
- Production Environment:** Contains one item:
  - Rule Execution Server console**: Monitor decision services deployed from Decision Center used in the context of a production application. Includes "Launch" and "More info" buttons.

Appendix: IBM ODM on Cloud

© Copyright IBM Corporation 2019

*Figure B-9. IBM ODM on Cloud user portal*

After you log in to IBM ODM on Cloud, you see the user portal. The applications that you can access depend on your user role.

Use the IBM ODM on Cloud user portal to start the Operational Decision Manager modules that you can access in the three different environments: development, test, and production. You can start the Decision Center consoles and Rule Execution server, and download Rule Designer from the user portal.



## Using Rule Designer

- Click **Download** from the user portal
- Two options for Rule Designer:
  - Stand-alone: Download a version of Rule Designer that is configured for use with IBM ODM on Cloud
  - Plug-ins: If you already use Eclipse, you can download Rule Designer plug-ins to use with your Eclipse installation
- Start Rule Designer by double-clicking `eclipse.exe`



Appendix: IBM ODM on Cloud

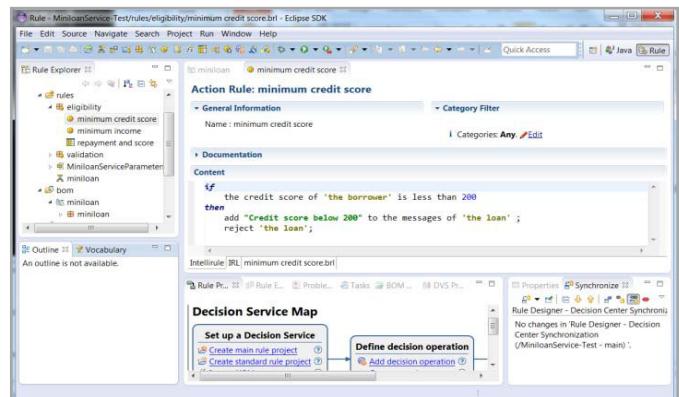
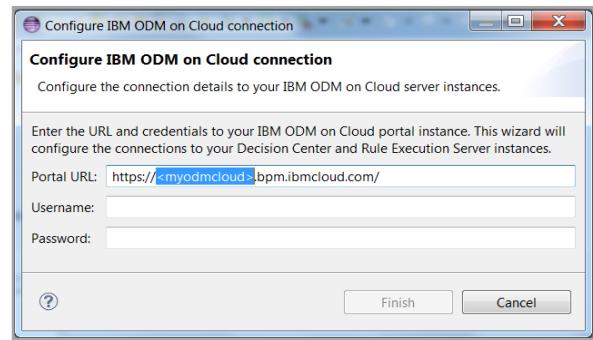
© Copyright IBM Corporation 2019

*Figure B-10. Using Rule Designer*



## Using Rule Designer (2 of 2)

- Configure Rule Designer to use the URL of your ODM on Cloud instance
- Rule Designer interface similar to on-premises version
  - However, some on-premises features (such as certain perspectives, like the Samples console) not available
- Switch to the Rule perspective to work on decision services
  - Create or import decision services
  - Publish decision services to Decision Center on the cloud



Appendix: IBM ODM on Cloud

© Copyright IBM Corporation 2019

Figure B-11. Using Rule Designer (2 of 2)



## Using Decision Center and Rule Execution Server (1 of 3)

- In the user portal, find the environment that you want to use (Development, Test, or Production)
- Click **Launch** for the module that you want to start

A screenshot of the IBM ODM on Cloud user portal. It displays three main modules: "Decision Center Business console", "Decision Center Enterprise console", and "Rule Execution Server console". Each module has a circular icon, a brief description, and "Launch" and "More info" buttons.

|   |   |  |
|---|---|--|
| <b>Decision Center<br/>Business console</b><br><br>Author, test, govern and deploy decision services<br><br> Launch  More info | <b>Decision Center<br/>Enterprise console</b><br><br>Perform occasional administration or advanced management activities.<br><br> Launch  More info | <b>Rule Execution Server<br/>console</b><br><br>Manage and monitor decision services deployed from Rule Designer or Decision Center as part of your development activities.<br><br> Launch  More info |
|---|---|--|

Appendix: IBM ODM on Cloud

© Copyright IBM Corporation 2019

Figure B-12. Using Decision Center and Rule Execution Server (1 of 3)

The IBM ODM on Cloud modules can be started from the user portal.



## Using Decision Center and Rule Execution Server (2 of 2)

- Log in using your ODM on Cloud user name and password
- Module opens in web browser window
  - Interface is the same as on-premises version

A screenshot of the IBM Decision Center and Rule Execution Server interface. The top navigation bar includes 'Decision Center', 'HOME', 'LIBRARY', 'WORK', and a help icon. The main content area has tabs for 'What's New' and 'Stream'. Under 'What's New', there is a section for 'New Rules' listing: 'maximum amount (1)', 'repayment and score (1)', 'minimum income (1)', and 'minimum credit score (1)'. Below this is a section for 'New Updates on Followed Rules' stating 'There have been no new updates on followed rules.' and a section for 'New Comments in Activity Stream' stating 'There have been no new comments in the activity stream.' To the right, a sidebar titled 'Followed Rules' lists four items: 'maximum amount (1)', 'minimum credit score (1)', 'minimum income (1)', and 'repayment and score (1)'. Below this is a section titled 'Rules Recently Worked On' with the message 'You have not worked on any rules yet.'.

Appendix: IBM ODM on Cloud

© Copyright IBM Corporation 2019

*Figure B-13. Using Decision Center and Rule Execution Server (2 of 2)*

The example ODM module that is shown on this slide is the Business console. It has the same interface as the on-premises version.

## Finding help for IBM ODM on Cloud

- IBM Knowledge Center for IBM ODM on Cloud

[http://www.ibm.com/support/knowledgecenter/SS7J8H/welcome/kc\\_welcome\\_cloud.html](http://www.ibm.com/support/knowledgecenter/SS7J8H/welcome/kc_welcome_cloud.html)

- Complete product documentation for IBM ODM on Cloud, including a “Getting Started” tutorial
- IBM ODM on Cloud user portal also has direct links to the documentation

- IBM ODM Support Portal

[https://www.ibm.com/support/entry/portal/product/websphere\\_ibm\\_operational\\_decision\\_manager](https://www.ibm.com/support/entry/portal/product/websphere_ibm_operational_decision_manager)

- Support Portal provides tools and resources for help with IBM Operational Decision Manager
- Open service requests, view fix lists, access community resources, and more

Figure B-14. Finding help for IBM ODM on Cloud

For more information about how to use IBM ODM on Cloud, see the IBM Knowledge Center for IBM ODM on Cloud.

The IBM ODM Support Portal provides general information on IBM Operational Decision Manager and access to other resources that you might find of interest. You can also use the Support Portal to open a service request.

---

# Appendix C. Appendix: ODM on IBM Cloud Private (ICP)

## Estimated time

00:15

## Overview

This unit describes Operational Decision Manager on Cloud.

## IBM Cloud Private

- Cloud platform located behind your firewall
- Enterprise content catalog
  - Open Source and IBM Middleware, DevOps, Data, Analytics, and AI Software
- Core operational services
  - Built-in management, security, and automation
  - Logging, monitoring, metering, security, alerting
- Kubernetes container orchestration platform

Appendix: ODM on IBM Cloud Private (ICP)

© Copyright IBM Corporation 2019

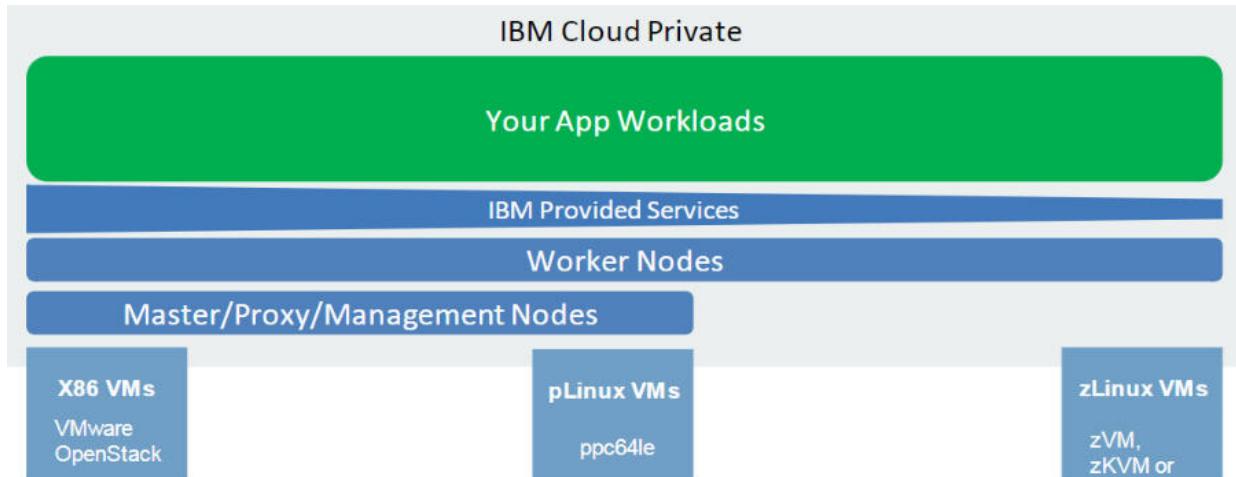
*Figure C-1. IBM Cloud Private*

IBM Cloud Private (ICP) is a cloud platform that is located behind your firewall. It is an integrated platform that consists of the IaaS, PaaS, and developer services necessary to create, run, and manage cloud applications.

With IBM Cloud Private, you use your own infrastructure to host services for a limited number of users behind a firewall. IBM Cloud Private, by virtue of being private, gives you control and flexibility in the way you set up, configure, and run your services.

# IBM Training

## IBM Cloud Private stack



Appendix: ODM on IBM Cloud Private (ICP)

© Copyright IBM Corporation 2019

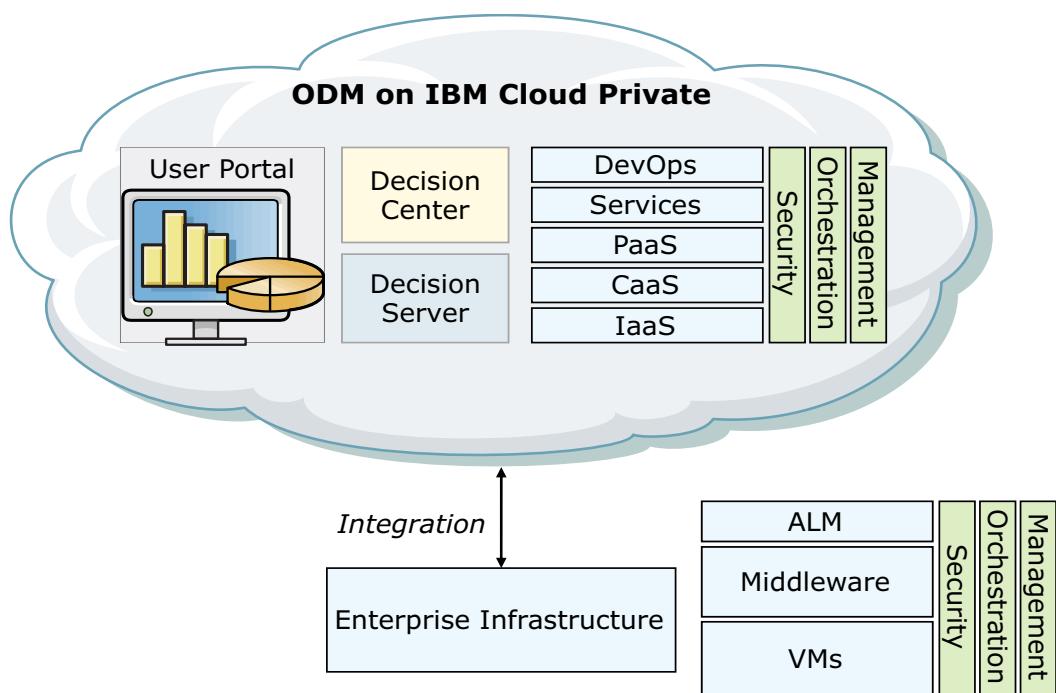
Figure C-2. IBM Cloud Private stack

This slide shows how the IBM Cloud Private stack distributes workloads and services to the servers in a cluster by using different classes of node: boot, master, worker, proxy, and a management node to prevent the master node from becoming overloaded.

IBM Training



## ODM on IBM Cloud Private



Appendix: ODM on IBM Cloud Private (ICP)

© Copyright IBM Corporation 2019

Figure C-3. ODM on IBM Cloud Private

Using ODM on IBM Cloud Private you create and provision multiple machines, change computing resources on-demand, scale applications, and take advantage of a self-service platform.

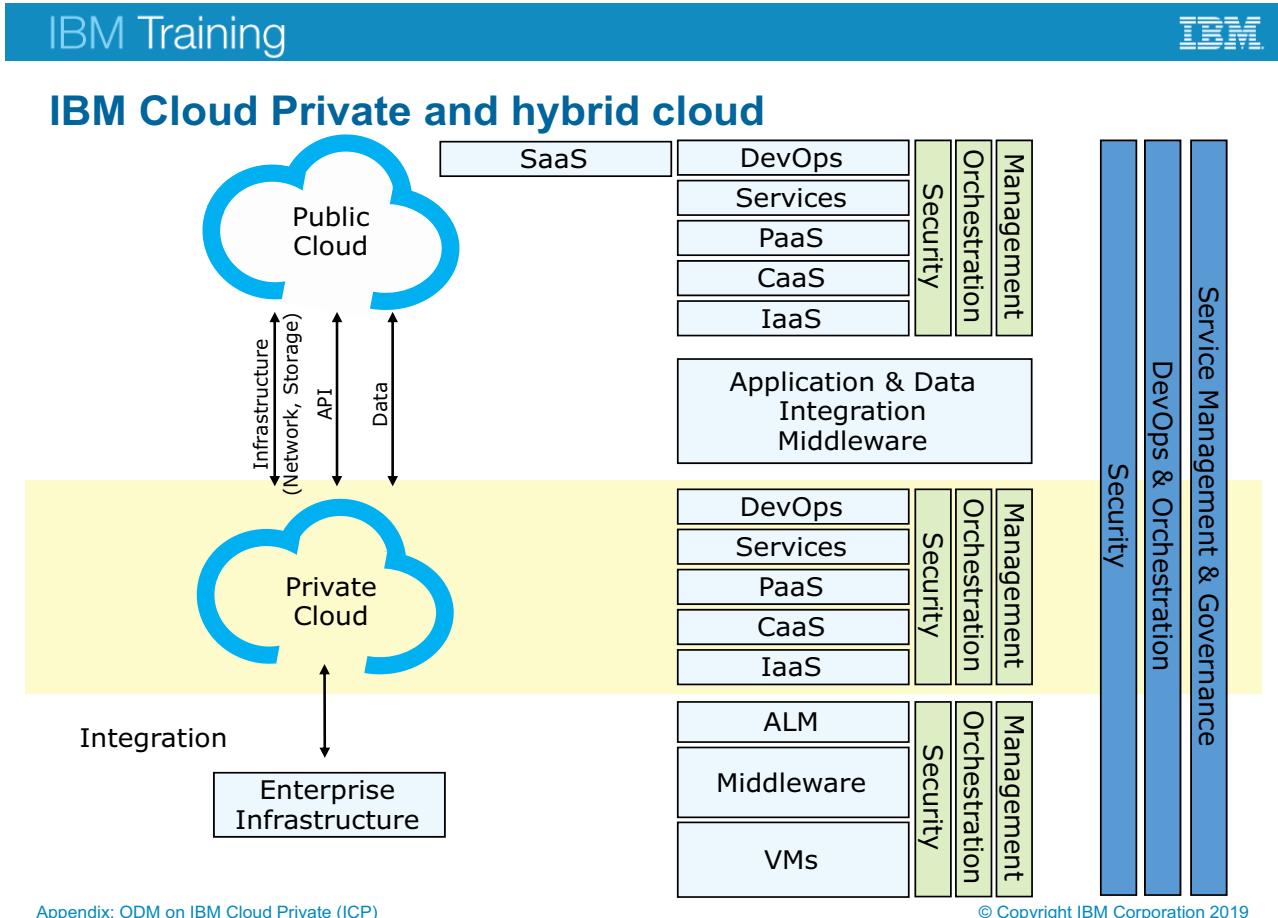


Figure C-4. IBM Cloud Private and hybrid cloud

You can also mix private and public cloud services. With a hybrid approach, you can scale computing requirements beyond the private cloud and into the public cloud. A typical usage scenario for ODM on IBM Cloud Private in a hybrid environment might involve these steps:

1. Develop a decision service project in Rule Designer, and test it in ODM on IBM Cloud Private.
2. When you are ready to scale your services, download and redeploy the executable assets (RuleApp and XOM) to Decision Server running on-premises or in a public cloud, like Operational Decision Manager on Cloud.

## Benefits of ODM on IBM Cloud Private

- Reduced costs
  - Simpler access to more capacity, memory, and CPU
  - Pooled computing resources and virtualization
  - Dedicated hardware provides security, improved up-time, reliability
- Time saving
  - Resources can be allocated as demand changes
  - Use built-in monitoring tools
  - Browse and install packages in your cluster from self-service catalog
- Additional capabilities with ICP
  - Secures your data
  - Opens your data center to work with cloud services
  - Visibility of the running applications
  - Updates workloads and the underlying platform by using continuous delivery techniques without requiring downtime
  - Manages application data for backup, recovery, and disaster failover

Figure C-5. Benefits of ODM on IBM Cloud Private

## Advantages of virtualization

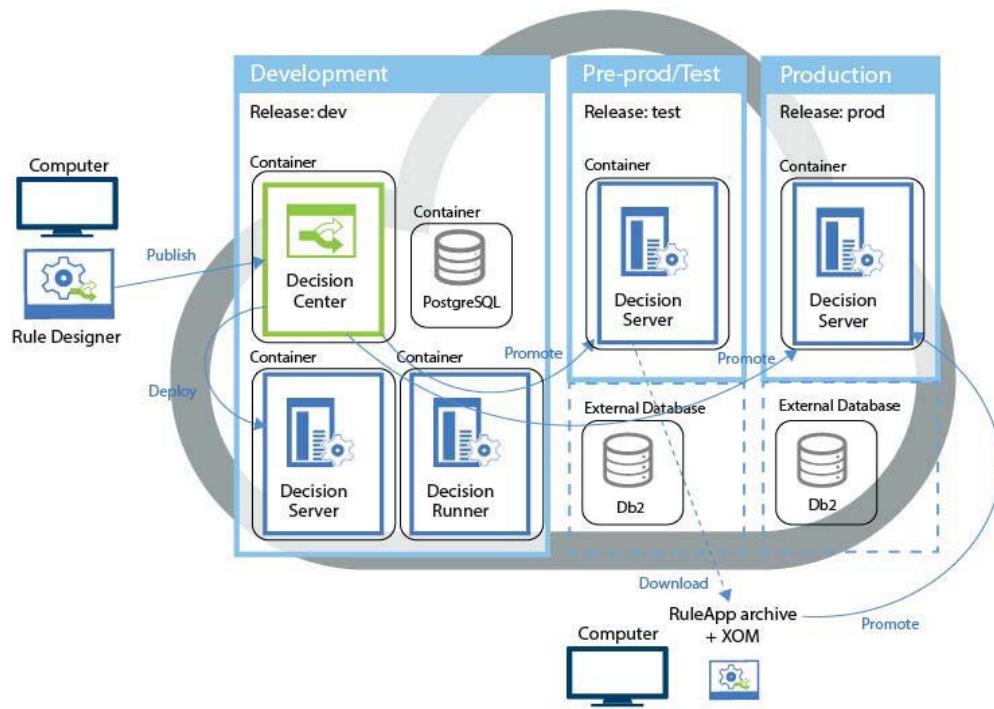
- Decreases rack space, power usage
- Increases value of physical server hardware by sharing resources with virtual servers
- Quickly create copies of servers and have them up and running
- Automatically allocate resources when needed
- Turn off servers during low usage

*Figure C-6. Advantages of virtualization*

Cloud computing removes the complexity of sizing servers for a particular purpose and workload. Allocation of resources, such as disk space, RAM, or CPU takes moments. You do not need to find physical servers with the resources to match your requirements.



## Decision service lifecycle on ICP



Appendix: ODM on IBM Cloud Private (ICP)

© Copyright IBM Corporation 2019

Figure C-7. Decision service lifecycle on ICP

To support the typical ODM workflow on ICP, you can use the **ibm-odm-prod** Helm chart to create separate environments for your development, test, and production needs.

This slide shows the minimal number of ODM instances in order to separate activities. A decision service is first published from Rule Designer to an ODM development instance, then promoted to an ODM test instance before it is promoted to an ODM production instance.

## For more information

- IBM Knowledge Center: **Operational Decision Manager 8.9.2 on IBM Cloud Private**  
[www.ibm.com/support/knowledgecenter/en/SSQP76\\_8.9.2/com.ibm.odm.icp/kc\\_welcome\\_odm\\_icp.html](http://www.ibm.com/support/knowledgecenter/en/SSQP76_8.9.2/com.ibm.odm.icp/kc_welcome_odm_icp.html)
- developerWorks: **Decision service lifecycle on IBM Cloud Private**  
<https://developer.ibm.com/odm/docs/best-practices/odm-lifecycle-icp/>

*Figure C-8. For more information*

Cloud computing removes the complexity of sizing servers for a particular purpose and workload. Allocation of resources, such as disk space, RAM, or CPU takes moments. You do not need to find physical servers with the resources to match your requirements.



IBM Training



© Copyright International Business Machines Corporation 2019.