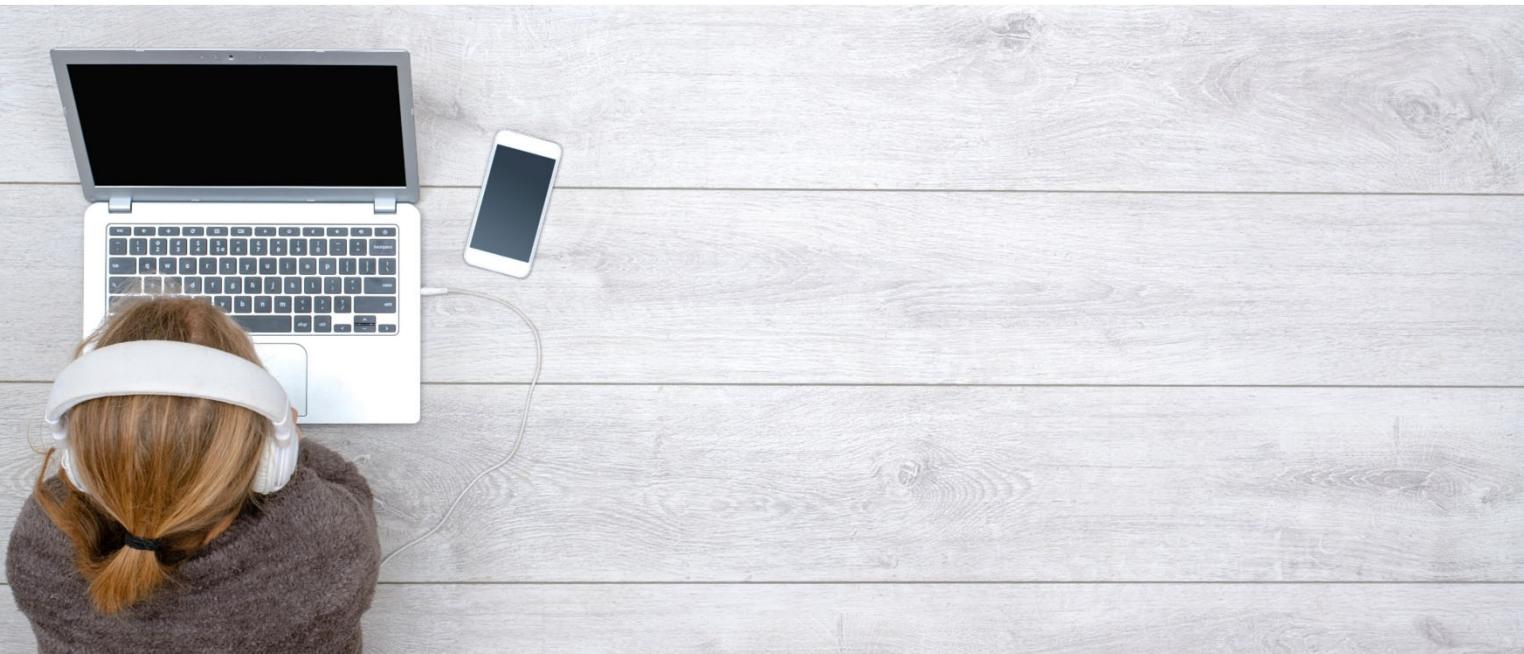


Notebook

Developing Rule Solutions in IBM Operational Decision Manager V8.10.5

Course code WB404 / ZB404 ERC 2.0



IBM Training

IBM

January 2021 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.

© Copyright International Business Machines Corporation 2021.

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

Trademarks	xvii
Course description	xix
Agenda	xxi
Unit 1. Introducing IBM Operational Decision Manager	1-1
Unit objectives	1-2
Topics	1-3
1.1. What is decision management?	1-4
What is decision management?	1-5
Why is there a need for decision management?	1-6
Example: Financial industry	1-7
Examples: FACTA and GDPR regulatory changes	1-8
Challenges to decision automation challenges	1-9
Meeting the challenges	1-10
How does decision management meet the challenge?	1-11
Using operational decision management	1-12
1.2. Digitizing business rules	1-13
Digitizing business rules	1-14
What is a business rule?	1-15
From business policy to business rules	1-16
Synchronized business and IT cycles	1-17
1.3. Introducing IBM Operational Decision Manager	1-18
Introducing IBM Operational Decision Manager	1-19
Operational Decision Manager: Tools	1-20
Operational Decision Manager: Roles and activities	1-22
1.4. ODM portfolio	1-23
ODM portfolio	1-24
Options for using ODM	1-25
IBM ODM on Cloud: Three runtime environments	1-26
Developing decision services in a hybrid cloud environment	1-27
ODM on Certified Kubernetes	1-28
Automation Decision Services	1-29
1.5. ODM roles	1-30
ODM roles	1-31
User roles	1-32
Interaction between roles	1-33
Business analyst	1-34
Policy manager	1-35
Rule author	1-36
Architect	1-37
Developer	1-38
Administrator	1-39
Discussion: Who is in charge of what?	1-40
1.6. Governance and decision management	1-41
Governance and decision management	1-42
Applying governance to decision management	1-43
Unit summary	1-44
Review questions (1 of 2)	1-45

Review questions (2 of 2)	1-46
Review answers (1 of 2)	1-47
Review answers (2 of 2)	1-48
Exercise: Operational Decision Manager in action	1-49
Exercise introduction	1-50
Exercise workflow	1-51
Unit 2. Developing decision services	2-1
Unit objectives	2-2
Topics	2-3
2.1. Using an agile development approach	2-4
Using an agile development approach	2-5
Using an agile development approach	2-6
Agile Business Rule Development (ABRD)	2-7
Discovery and modeling	2-8
Discovery: Identifying decisions in the process (2 of 2)	2-10
Discovery: Identifying the rules	2-11
Discovery: Identifying vocabulary	2-12
Implementing the model	2-13
2.2. Designing the decision service	2-14
Designing the decision service	2-15
Design	2-16
Setting up the infrastructure	2-17
Implementing vocabulary models	2-18
Extracting a ruleset	2-19
2.3. Setting up decision services in Rule Designer	2-20
Setting up decision services in Rule Designer	2-21
Decision service map	2-22
Create a decision service rule project in Rule Designer	2-23
Decision service rule project templates	2-24
Design: XOM and BOM	2-25
Design: XOM	2-26
Design: BOM	2-27
Defining the decision operation	2-28
Using the Operation Map	2-29
Designing the signature	2-30
Designing the signature	2-31
2.4. Project properties	2-32
Project properties	2-33
Properties: Project hierarchy	2-34
Properties: Folders	2-35
Properties: Project references	2-36
Properties: Rule engine type	2-37
2.5. Using modular project organization	2-38
Using modular project organization	2-39
Modular structure (1 of 4)	2-40
Modular structure (2 of 4)	2-41
Modular structure (3 of 4)	2-42
Modular structure (4 of 4)	2-43
2.6. Sharing and synchronizing decision services	2-44
Sharing and synchronizing decision services	2-45
Synchronization between users	2-46
Synchronization tools	2-47
Synchronization architecture	2-48
Connection entries	2-49
Choosing how to synchronize	2-50

Synchronization commands	2-51
Choosing the master source	2-52
Unit summary	2-53
Review questions	2-54
Review answers	2-55
Exercise: Setting up decision services	2-56
Exercise introduction	2-57
Unit 3. Modeling decisions	3-1
Unit objectives	3-2
Topics	3-3
3.1. Introducing decision modeling	3-4
Introducing decision modeling	3-5
Recall: Bottom-up 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
Unit summary	3-20
Review questions	3-21
Review answers	3-22
Exercise: Modeling decisions	3-23
Exercise introduction	3-24
Exercise overview	3-25
Unit 4. Programming with business rules	4-1
Unit objectives	4-2
Topics	4-3
4.1. Introducing ruleset integration and execution	4-4
Introducing ruleset integration and execution	4-5
Integrating business logic	4-6
Execution environments	4-7
4.2. What is a rule engine?	4-8
What is a rule engine?	4-9
Engine	4-10
Rule execution by the rule engine	4-11
Decision engine API	4-12
Compilation and execution	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)	4-22

Execution modes: RetePlus (3 of 3)	4-23
RetePlus: Rule order and selection	4-24
Refraction (1 of 2)	4-25
Refraction (2 of 2)	4-26
Recency (1 of 2)	4-27
Recency (2 of 2)	4-28
Choosing an execution mode	4-29
Choosing execution mode according to application type	4-30
4.4. RetePlus example: Trucks and drivers.	4-31
RetePlus example: Trucks and drivers	4-32
RetePlus in action	4-33
Input objects in working memory	4-34
Evaluating rules with objects	4-35
Rule instances in the agenda for execution (1 of 2)	4-36
Rule instances in the agenda for execution (2 of 2)	4-37
ChangeTire executes	4-38
Side effects: Updated objects create matches	4-39
AssignDriver executes	4-40
Side effects: Updated objects no longer match	4-41
AssignDriver executes again	4-42
Side effects of AssignDriver	4-43
Rule execution completes when agenda is empty	4-44
Unit summary	4-45
Review questions	4-46
Review answers	4-47

Unit 5. Developing object models **5-1**

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	5-48
Refactoring	5-49
Refactoring	5-50
From the XOM to the rules	5-51
BOM and XOM evolution	5-52
XOM changes (1 of 3)	5-53
XOM changes (2 of 3)	5-54
XOM changes (3 of 3)	5-55
BOM changes	5-56
Vocabulary changes	5-57
Unit summary	5-58
Review questions	5-59
Review answers	5-60
Exercise: Working with the BOM	5-61
Exercise introduction	5-62
Exercise: Refactoring	5-63
Exercise introduction	5-64
Unit 6. Orchestrating ruleset execution	6-1
Unit objectives	6-2
Topics	6-3
6.1. Controlling rule execution	6-4
Controlling rule execution	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
Rule tasks	6-11
Initial and final actions	6-12
Transitions	6-13
Forks and joins	6-14
Expression of initial or final actions, and conditions	6-15
Main ruleflow	6-16
6.3. Controlling rule selection for execution	6-17
Controlling rule selection for execution	6-18
Rule selection pipe	6-19
Ruleflow scope selection	6-20
Runtime rule selection	6-21
Rule hierarchies	6-22

Rule overriding	6-23
Rule condition evaluation	6-24
6.4. Controlling rule order during rule execution	6-25
Controlling rule order during rule execution	6-26
Controlling rule order	6-27
Rule priority	6-28
Rule task execution order (1 of 2)	6-29
Rule task execution order (2 of 2)	6-30
Execution modes	6-31
Unit summary	6-32
Review questions	6-33
Review answers	6-34
Exercise: Working with ruleflows	6-35
Exercise introduction	6-36
Unit 7. Authoring rules.....	7-1
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	7-6
Operational Decision Manager roles and activities	7-7
7.2. Authoring action rules with BAL	7-8
Authoring action rules with BAL	7-9
Business Action Language (BAL)	7-10
Rule structure	7-11
Example of action rule	7-12
Rule definitions: Overview	7-13
Rule definitions: Values	7-14
Rule definitions: Constraints	7-15
Multiple variables in BAL	7-16
Rule conditions: Introduction	7-17
Rule conditions: Comparison test	7-18
Rule conditions: Membership test	7-19
Rule conditions: Existence test	7-20
Rule conditions: Count test	7-21
Multiple conditions	7-22
Avoiding ambiguity with parentheses (1 of 2)	7-23
Avoiding ambiguity with parentheses (2 of 2)	7-24
Use of commas (,) in rule conditions	7-25
Rule actions	7-26
Examples of rule actions	7-28
BAL number operators	7-29
BAL arithmetic operators	7-30
7.3. Authoring decision tables	7-31
Authoring decision tables	7-32
About decision tables	7-33
Example: Symmetrical rules	7-34
Example: Symmetrical rules in a decision table	7-35
Structure of a decision table	7-36
Preconditions	7-37
Error checking in the editors	7-38
Locking facilities	7-39
Decision table size	7-40
7.4. Advanced Rule Language (ARL)	7-41
Advanced Rule Language (ARL)	7-42

Translation from BAL to ARL	7-43
Viewing ARL	7-44
7.5. Technical rules	7-45
Technical rules	7-46
Writing technical rules in IRL	7-47
IRL syntax	7-48
Technical rule structure	7-50
Multiple variables in IRL	7-51
Create a technical rule	7-52
7.6. Defining objects that are used in rules	7-53
Defining objects that are used in rules	7-54
Introduction	7-55
Parameters	7-56
Ruleset variables	7-57
Objects in working memory	7-58
Working with objects in rule artifacts	7-59
Using IRL	7-60
Using verbalized elements in BAL	7-61
Rule variables (1 of 2)	7-62
Rule variables (2 of 2)	7-63
Using automatic variables	7-64
Unit summary	7-65
Review questions	7-66
Review answers	7-67
Exercise: Exploring action rules	7-68
Exercise introduction	7-69
Exercise: Authoring action rules	7-70
Exercise introduction	7-71
Exercise: Authoring decision tables	7-72
Exercise introduction	7-73
Unit 8. Customizing rule vocabulary with categories and domains	8-1
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
Accessing the domain file	8-32
Updating dynamic domain files	8-33
Viewing domain changes	8-34
Unit summary	8-35
Review questions	8-36
Review answers	8-37
Exercise: Working with static domains	8-38
Exercise introduction	8-39
Exercise: Working with dynamic domains	8-40
Exercise introduction	8-41
Unit 9. Working with queries	9-1
Unit objectives	9-2
Topics	9-3
9.1. Searching for rule artifacts	9-4
Searching for rule artifacts	9-5
Search rule artifacts	9-6
Search for rule dependencies	9-7
9.2. Querying rules	9-8
Querying rules	9-9
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
9.3. Extracting rulesets	9-20
Extracting rulesets	9-21
Queries and ruleset extractor	9-22
Ruleset archive	9-23
Unit summary	9-24
Review questions	9-25
Review answers	9-26
Exercise: Working with searches and queries	9-27
Exercise introduction	9-28
Unit 10. Debugging rulesets	10-1
Unit objectives	10-2
Topics	10-3
10.1. Working with launch configurations	10-4
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
10.2. Automatic exception handling.	10-12
Automatic exception handling	10-13
Handling exceptions	10-14
Enabling automatic exception handling	10-15
Capturing trace logs in Rule Designer	10-16
10.3. Using Rule Designer debugging tools	10-17
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
Exercise: Debugging a ruleset	10-27
Exercise introduction	10-28
Unit 11. Enabling tests and simulations.	11-1
Unit objectives	11-2
Topics	11-3
11.1. Overview of testing and simulation.	11-4
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	11-32
Exercise: Enabling rule validation	11-33
Exercise introduction	11-34
Unit 12. Managing deployment	12-1
Unit objectives	12-2
Topics	12-3

12.1. Deploying rules.....	12-4
Deploying rules	12-5
Preparing for deployment	12-6
RuleApp and RuleApp archive	12-7
RuleApp and ruleset properties	12-8
Deployed RuleApp and ruleset names	12-10
Ruleset path	12-11
Version policy (1 of 2)	12-12
Version policy (2 of 2)	12-13
Deployment configurations (1 of 3)	12-14
Deployment configurations (2 of 3)	12-15
Deployment configurations (3 of 3)	12-16
Deployment from Decision Center	12-17
Deployment from Rule Execution Server console	12-18
12.2. Managing XOMs.....	12-19
Managing XOMs	12-20
XOM deployment with decision services	12-21
Modifying the XOM	12-22
Managed Java XOM artifacts	12-23
Link from a ruleset to a managed Java XOM element	12-24
Java XOM deployment	12-25
XOM deployment from Decision Center	12-26
Managed XOM and Decision Center	12-27
Managed XOMs in Rule Designer and Decision Center	12-28
Embedded managed Java XOMs	12-29
Managed XOM storage	12-30
12.3. Building and deploying with Ant tasks	12-31
Building and deploying with Ant tasks	12-32
Deployment and management with Ant	12-33
Setting up the Ant environment	12-34
12.4. Building and deploying with the REST API.....	12-35
Building and deploying with the REST API	12-36
REST API tool in Rule Execution Server console	12-37
Decision Center API console	12-38
12.5. Building projects with the Build Command.....	12-39
Building projects with the Build Command	12-40
Building RuleApps with Build Command	12-41
Build Command Line	12-42
Build Command Maven plugin	12-43
Project Object Model (POM) files	12-44
Example decision service POM file (1 of 2)	12-45
Example decision service POM file (2 of 2)	12-46
Build results	12-47
Unit summary	12-48
Review questions	12-49
Review answers	12-50
Exercise: Managing deployment	12-51
Exercise introduction	12-52
Unit 13. Executing rules with Rule Execution Server	13-1
Unit objectives	13-2
Topics	13-3
13.1. Introducing managed execution	13-4
Introducing managed execution	13-5
Introduction to Rule Execution Server	13-6
Key functions of Rule Execution Server	13-7

Setting up a Rule Execution Server	13-8
13.2. Managed execution with Rule Execution Server	13-10
Managed execution with Rule Execution Server	13-11
Rule Execution Server in a production enterprise application	13-12
Introducing managed execution with Rule Execution Server	13-13
Example of possible platforms	13-14
13.3. Rule Execution Server modular architecture	13-15
Rule Execution Server modular architecture	13-16
Rule Execution Server architecture	13-17
Execution stack: Execution components	13-19
Execution Unit (XU)	13-20
JMX management and execution model	13-21
Management and monitoring stack: Console	13-22
Persistence layer	13-23
Ant tasks	13-24
13.4. Managed execution in action	13-25
Managed execution in action	13-26
Elements of a rule-based solution	13-27
Ruleset parsing	13-28
Ruleset execution	13-29
Ruleset update at run time	13-30
Runtime class loading	13-31
13.5. Platforms for Rule Execution Server	13-32
Platforms for Rule Execution Server	13-33
Architecture questions	13-34
Possible choices: Introduction	13-35
Java SE	13-37
Java EE	13-38
Java SE and Java EE: A summary	13-40
Transparent decision services	13-41
13.6. Rule Execution Server API	13-42
Rule Execution Server API	13-43
Introduction	13-44
Execution patterns: Synchronous	13-45
Execution patterns: Asynchronous	13-46
Rule session API overview	13-47
Rule session factories	13-48
Rule sessions	13-49
Stateless rule sessions	13-50
Stateful rule sessions	13-51
Rule session requests	13-52
Rule session requests: Decision ID	13-53
Rule session responses	13-54
Ruleset path for execution (1 of 2)	13-55
Ruleset path for execution (2 of 2)	13-56
Traces	13-57
Decision traces: How to	13-58
Decision traces: Complement	13-59
13.7. Executing rules in Java SE	13-60
Executing rules in Java SE	13-61
Rule execution in Java SE	13-62
Deployed Rule Execution Server artifacts in Java SE	13-63
Required Rule Execution Server API in Java SE	13-64
Example with a stateless rule session in Java SE	13-65
13.8. Executing rules in Java EE	13-67
Executing rules in Java EE	13-68

Rule execution in Java EE	13-69
Deployed Rule Execution Server artifacts in Java EE	13-70
Required API in Java EE	13-71
Java EE POJO rule session	13-72
Java EE EJB rule session	13-73
Java EE message-driven bean (MDB)	13-74
13.9. Executing rules as transparent decision services.....	13-75
Executing rules as transparent decision services	13-76
Rules as transparent decision services	13-77
Unit summary	13-78
Review questions	13-79
Review answers	13-80
Exercise: Exploring the Rule Execution Server console	13-81
Exercise introduction	13-82
Unit 14. Working with transparent decision services.....	14-1
Unit objectives	14-2
Topics	14-3
14.1. Calling decision services	14-4
Calling decision services	14-5
Working with HTDS description files	14-6
Calling a decision service as a SOAP web service	14-7
Calling a decision service by using REST API	14-8
14.2. Using the REST service for ruleset execution	14-9
Using the REST service for ruleset execution	14-10
REST service for ruleset execution	14-11
Endpoint URIs	14-12
HTTP methods and content types	14-13
Request and response schema	14-14
WADL representation	14-15
Exposing decision services as APIs	14-16
14.3. Testing ruleset execution with the REST API.....	14-17
Testing ruleset execution with the REST API	14-18
Steps for using the REST service for ruleset execution	14-19
Retrieving the HTDS description (1 of 2)	14-20
Retrieving the HTDS description (2 of 2)	14-21
Generating WADL execution request	14-22
Example: OpenAPI JSON request	14-23
Execution: OpenAPI JSON response	14-24
Monitoring transparent decision services	14-25
Unit summary	14-26
Review questions	14-27
Review answers	14-28
Exercise: Executing rules as a hosted transparent decision service (HTDS)	14-29
Exercise introduction	14-30
Unit 15. Auditing and monitoring ruleset execution	15-1
Unit objectives	15-2
Topics	15-3
15.1. Auditing ruleset execution.....	15-4
Auditing ruleset execution	15-5
Auditing ruleset execution	15-6
Decision Warehouse	15-7
Default use of Decision Warehouse	15-8
Step 1: Install and create database resource	15-9
Step 2: Enable rule execution monitoring	15-10

Step 3: Execute the ruleset	15-11
Step 4-a: View stored decision traces	15-12
Step 4-b: Filter stored decision traces	15-13
Step 4-c: View decision details and fired rules	15-14
15.2. Monitoring ruleset execution	15-15
Monitoring ruleset execution	15-16
Monitoring ruleset execution	15-17
Debug mode	15-18
Ruleset monitoring options (1 of 5)	15-19
Ruleset monitoring options (2 of 5)	15-20
Ruleset monitoring options (3 of 5)	15-21
Ruleset monitoring options (4 of 5)	15-22
Ruleset monitoring options (5 of 5)	15-23
Ruleset statistics (1 of 3)	15-24
Ruleset statistics (2 of 3)	15-25
Ruleset statistics (3 of 3)	15-26
Test of ruleset execution	15-27
Logged events on Execution Units	15-28
Unit summary	15-29
Review questions	15-30
Review answers	15-31
Exercise: Auditing ruleset execution through Decision Warehouse	15-32
Exercise introduction	15-33
Unit 16. Applying decision governance	16-1
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. Decision governance framework	16-17
Decision governance framework	16-18
Decision governance framework overview	16-19
Publishing decision services from Rule Designer	16-20
Decision Center: Governance in Business console	16-21
States and user roles	16-22
Releases	16-23
Release governance	16-24
Release deployment	16-25
16.3. Operational Decision Manager support for governance	16-26
Operational Decision Manager support for governance	16-27
Decision lifecycle in Operational Decision Manager (1 of 2)	16-28
Decision lifecycle in Operational Decision Manager (2 of 2)	16-29
How can you apply governance?	16-30
Unit summary	16-31
Review questions	16-32
Review answers (1 of 2)	16-33
Review answers (2 of 2)	16-34

Unit 17. Course summary	17-1
Unit objectives	17-2
Course objectives	17-3
Course objectives	17-4
IBM credentials: Badges and certifications	17-5
Learn more about this product	17-6
Additional resources (1 of 5)	17-7
Additional resources (2 of 5)	17-8
Additional resources (3 of 5)	17-9
Additional resources (4 of 5)	17-10
Additional resources (5 of 5)	17-11
Unit summary	17-12
Course completion	17-13
Appendix A. List of abbreviations	A-1
Appendix B. ODM on Cloud.....	B-1

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:

CICS®
IBM Cloud™
SPSS®

Express®
ILOG®
Tivoli®

IBM API Connect®
Redbooks®
WebSphere®

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 the on-premises version of IBM Operational Decision Manager V8.10.5, 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 V8.10.5

Duration: 5 days

Purpose

This course introduces developers to IBM Operational Decision Manager V8.10.5. It teaches you how 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 Rules, which is the primary working environment for developers. You learn how to support your business users by setting up the rule authoring environment and object models that are required to execute rule artifacts. You also work with rule authoring so that you can support business users to set up and customize the rule authoring and validation environments. You learn how to manage ruleset deployment and execution, and work extensively with Rule Execution Server. And, you learn how Operational Decision Manager features support decision governance so that it can be implemented in your organization.

The lab environment for this course uses Windows Server 2016 Standard.

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 ruleset 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
- Apply governance principles to decision management

Agenda



Note

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

Day 1

- (00:30) Course introduction
- (01:15) Unit 1. Introducing IBM Operational Decision Manager
- (02:00) Exercise 1. Operational Decision Manager in action
- (01:30) Unit 2. Developing decision services
- (01:45) Exercise 2. Setting up decision services
- (00:30) Unit 3. Modeling decisions
- (00:30) Exercise 3. Modeling decisions

Day 2

- (00:45) Unit 4. Programming with business rules
- (01:00) Unit 5. Developing object models
- (00:45) Exercise 4. Working with the BOM
- (00:45) Exercise 5. Refactoring
- (00:45) Unit 6. Orchestrating ruleset execution
- (00:45) Exercise 6. Working with ruleflows
- (01:30) 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
- (00:30) Exercise 12. Working with searches and queries
- (00:45) Unit 10. Debugging rulesets
- (01:00) Exercise 13. Debugging a ruleset (start)

Day 4

- (00:30) Exercise 13. Debugging a ruleset (finish)
- (01:00) Unit 11. Enabling tests and simulations
- (01:15) Exercise 14. Enabling rule validation
- (01:00) Unit 12. Managing deployment
- (01:00) Exercise 15. Managing deployment
- (02:00) Unit 13. Executing rules with Rule Execution Server

Day 5

- (00:45) Exercise 16. Exploring the Rule Execution Server console
- (00:45) Unit 14. Working with transparent decision services
- (00:30) Exercise 17. Executing rules using the REST service
- (01:00) Unit 15. Auditing and monitoring ruleset execution
- (01:00) Exercise 18. Auditing ruleset execution through Decision Warehouse
- (01:00) Unit 16. Applying decision governance
- (00:30) Unit 17. Course summary

Unit 1. Introducing IBM Operational Decision Manager

Estimated time

01:15

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 purpose of decision management
- Describe how the Operational Decision Manager architecture supports business and technical user roles and tasks
- Map the various roles that are involved in a decision management solution to roles in your organization
- Identify the need for governance

© Copyright IBM Corporation 2021

Figure 1-1. Unit objectives

Topics

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

© Copyright IBM Corporation 2021

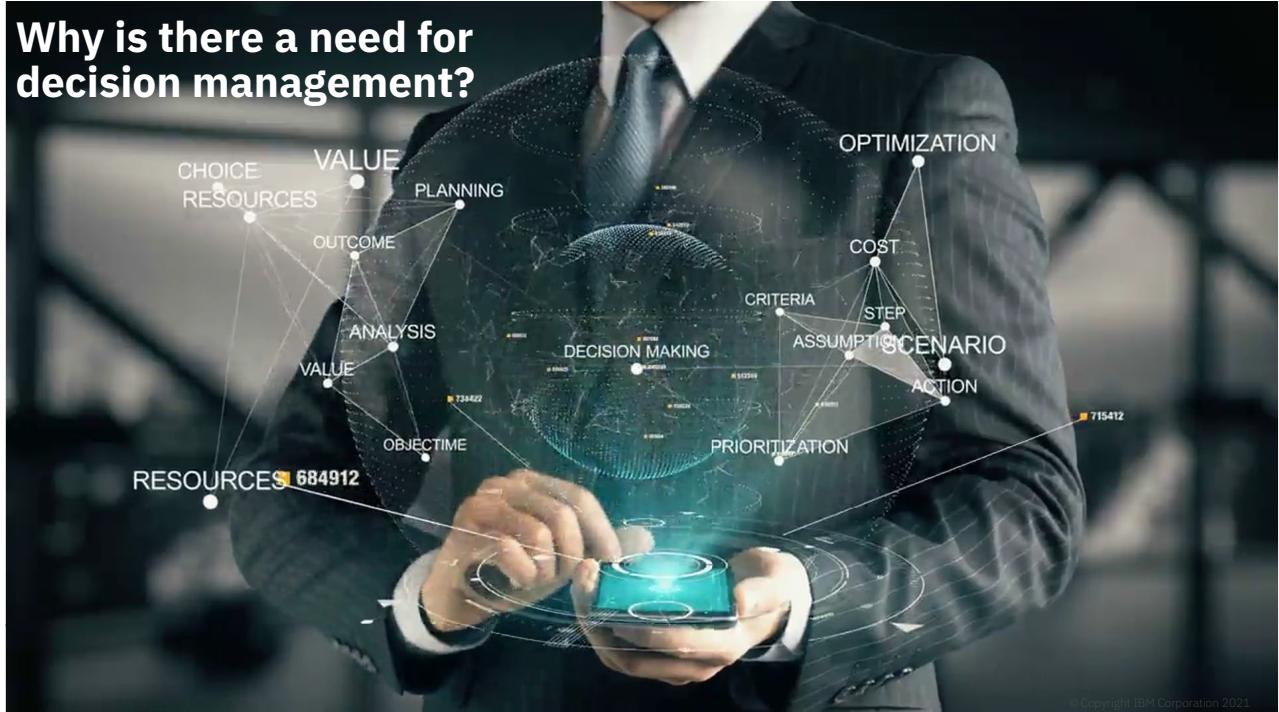
Figure 1-2. Topics

1.1. What is decision management?



What is decision management?

Figure 1-3. What is decision management?



© Copyright IBM Corporation 2021

Figure 1-4. Why is there a need for decision management?

Business rules are everywhere. Every business, government, and industry is challenged with making thousands—even millions—of decisions each day.

In many organizations, the **same** decision is made by **different** people, in different locations. How do **you ensure** that those decisions are consistent? How quickly can you **adapt** to meet market demands? or **comply** with new regulations?



Introducing IBM Operational Decision Manager

© Copyright IBM Corporation 2021

Figure 1-5. Example: Financial industry

To illustrate, consider the financial industry, which is a typical example of the current business environment for many different domains. It includes a broad network of interactions between employees, customers, suppliers, and partners, along with internal and external processes and systems, and government regulations that vary according to geography.

Whether the decision is to underwrite a mortgage, provide a loan, or extend credit, these types of decisions are repeated thousands of times per day. So they need to be correct and consistent.

Examples: FACTA and GDPR regulatory changes



US Foreign Account Tax Compliance Act (FACTA)



European General Data Protection Regulation (GDPR)

Figure 1-6. Examples: FACTA and GDPR regulatory changes

The business also needs to be able to respond quickly to dynamic nature of this environment, such as regulatory or economic changes. As was highlighted with the 2010 US Foreign Account Tax Compliance Act (FACTA) or the 2016 European General Data Protection Regulation (GDPR), regulatory changes in one country can impact an organization's business rules globally.



Challenges to decision management

- Decisions are **hardcoded** and **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

Figure 1-7. 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 or locked in processes. 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 respond to change.



How decision management meets the challenges

- Focus on **high-volume, repeatable** business decisions
- **Reduce load** on people so they are free to focus on subjective decisions
- Maximize **straight-through processing**
- **Consistent, predictable, traceable** decision making

Introducing IBM Operational Decision Manager

© Copyright IBM Corporation 2021

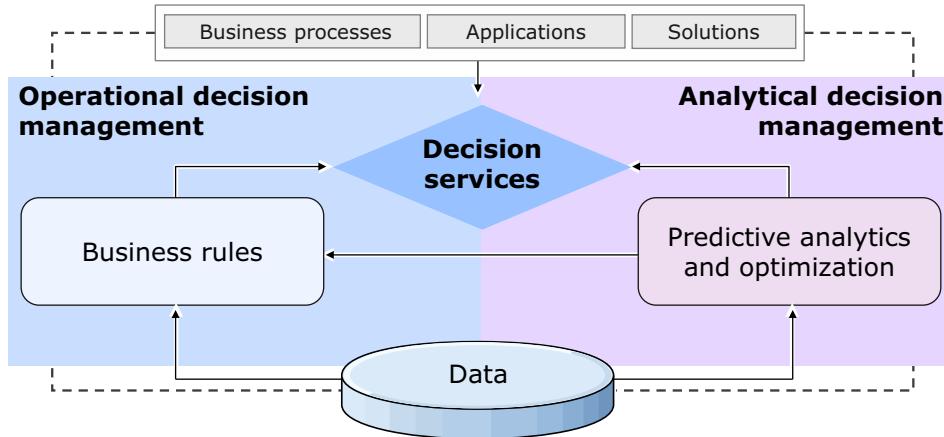
Figure 1-8. Meeting the challenges

Decision management focuses on high-volume, repeatable business decisions. By combining business expertise with technology, decision automation becomes an extension of the decisions that business people would make if they had unlimited time to consider those decisions.

Obviously, not every business decision can be automated. But by automating repetitive decisions, the business users are free to focus on decisions that require their skill and discernment.

What is decision management?

- Decision logic is captured, changed, and governed in a controlled and scalable way
- Business policies are deployed as ***decision services***
- Decision services can be called in real time by processes, applications, and other business solutions



Introducing IBM Operational Decision Manager

© Copyright IBM Corporation 2021

Figure 1-9. How does decision management meet the challenge?

With IBM Operational Decision Manager, your business policies are digitized and packaged as a *decision service* that 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.

By incorporating analytics and machine learning to analyze your decision outcomes and other business data, you can measure effectiveness of operational decisions and continue to improve them.



Note

Analytical decision management is not the focus of this course.

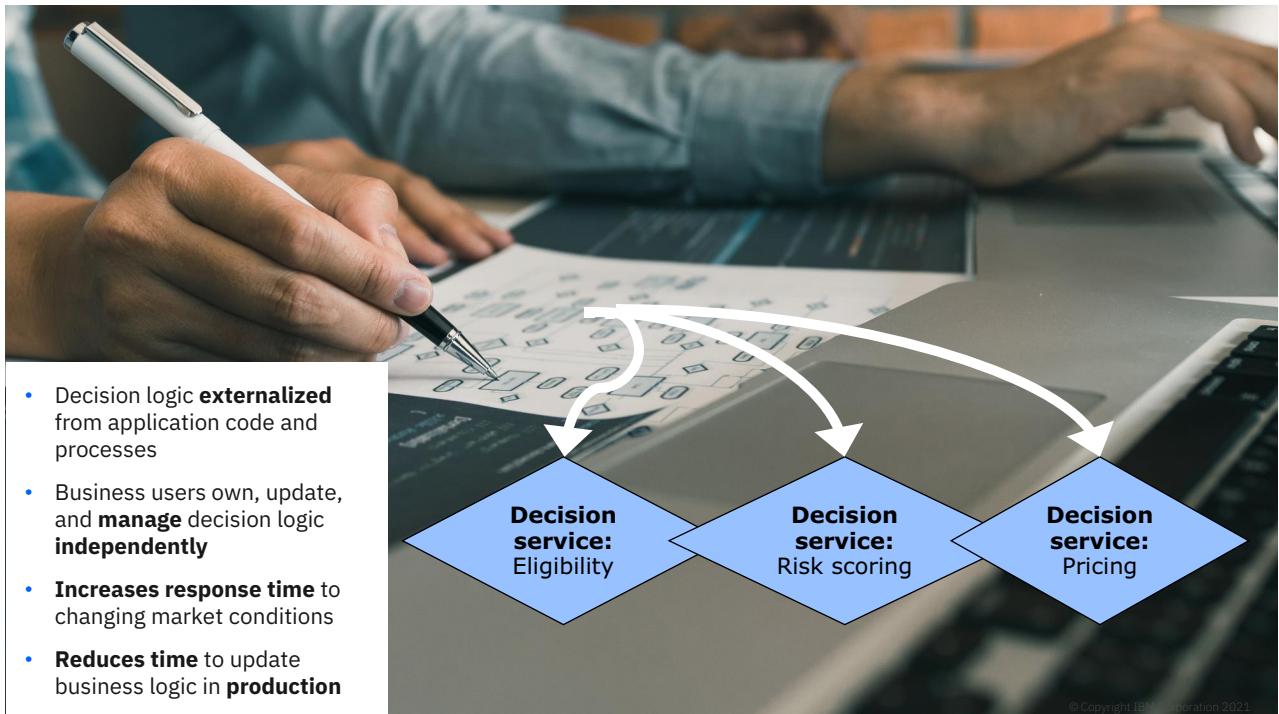


Figure 1-10. Using operational decision management

Because the decision logic is externalized from application code, it can be managed independently.

The business rules are coded in *natural language* and made available through a user interface where business experts can easily author, review, and manage them with minimal dependence on IT.

Changes to business policy do not affect application and process code. This reduces the time and effort to update that business logic in production systems and increases the organization's ability to respond to changes in the business environment.

Regardless of the industry, decision automation can improve **repeatable** decisions by maximizing straight-through processing and providing faster, consistent, predictable, and traceable decision making.

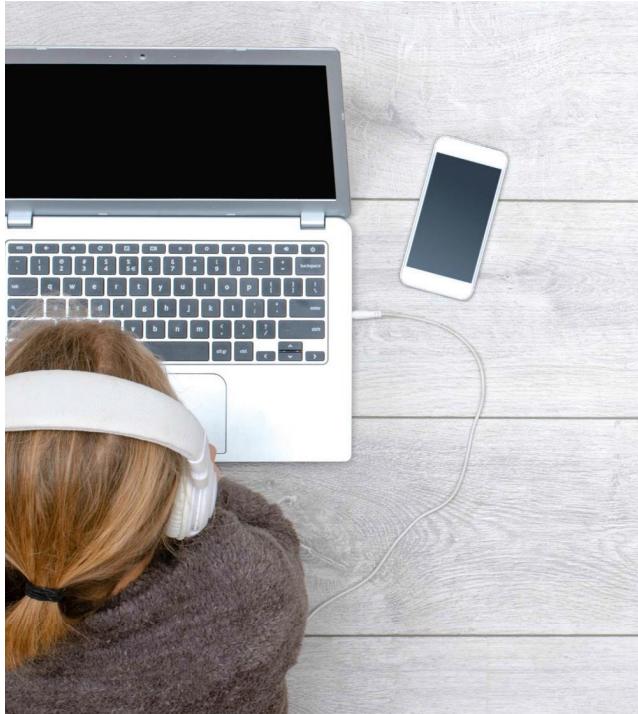


Information

Should rules be managed 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 service and improve business performance.

1.2. Digitizing business rules



Digitizing business rules

Figure 1-11. Digitizing business rules

What is a business rule?

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 1-12. 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, a business rule is a precise statement that describes, constrains, or controls some aspect of your business.
- From the development or IT perspective, business rules are technical assets that relate to the implementation of a decision system, and integration into an existing infrastructure. You see business rules as a package of executable business policy statements that can be invoked from an application.

From business policy to business rules

Rules formalize business policy as “if-then” statements

Business policy



Customers who spend a lot of money in a single transaction need an upgrade

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 1-13. From business policy to business rules

Every transaction, customer interaction, or process is dependent on business policies. Business rules formalize a business policy into a series of “if-then” statements that can be understood by the application.

For example, a business policy that reads as:

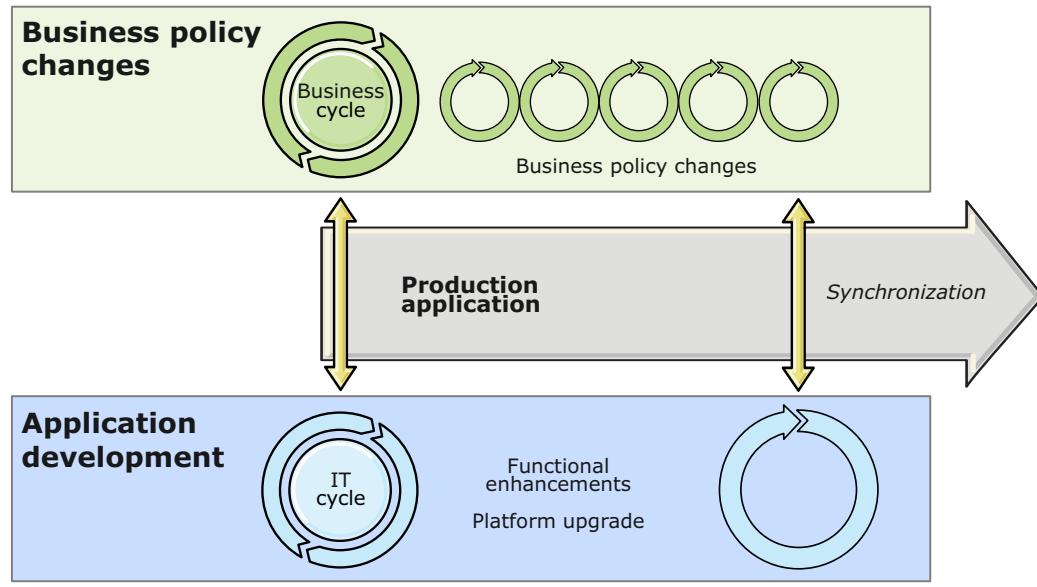
“Customers who spend a lot of money in a single transaction need an upgrade.”

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

These business rules codify organizational knowledge, such as corporate policies and best practices. A single business decision can require hundreds of rules to implement it.

Synchronized business and IT cycles



Introducing IBM Operational Decision Manager

© Copyright IBM Corporation 2021

Figure 1-14. Synchronized business and IT cycles

Because the business cycle is much shorter than the IT cycle, these cycles evolve in parallel and are managed asynchronously.

For example, 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 due to marketing strategies or regulation changes. Since the business rule changes do not affect code, policy managers can review and 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.

1.3. Introducing IBM Operational Decision Manager



Introducing IBM Operational Decision Manager

Figure 1-15. Introducing IBM Operational Decision Manager

Operational Decision Manager: Tools

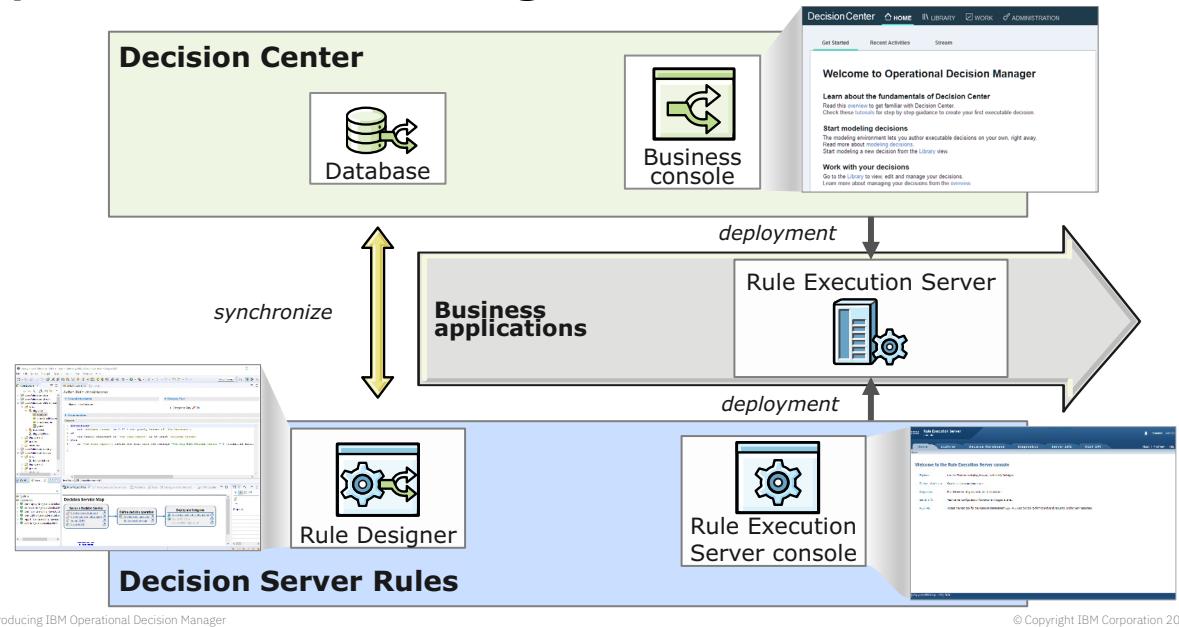


Figure 1-16. Operational Decision Manager: Tools

Because business and IT view business rules from distinct perspectives, this dual perspective is reflected in the Operational Decision Manager tooling.

As shown in this diagram, Operational Decision Manager consists of a set of modules that operate in two main environments:

- **Decision Center** for business users who develop and maintain the decision logic
- **Decision Server** for technical users who develop and maintain the rule application

Decision Center provides a collaborative environment for business users to define, validate, deploy, and govern decision logic.

- Business users work in **Business console**, which is a web-based interface for rule authoring, testing, deployment, and release management.
- Decision artifacts are stored in a centralized **database** with version control, release management, and secure access. The artifacts in this repository are accessible to both the Decision Center and Decision Server environments. Changes that are made in either environment can be **synchronized**. Both environments can also deploy.

Decision Server contains runtime components and Eclipse-based development tools.

- **Decision Server Rules** includes rules-based development tools and runtime environment.
 - Rule Designer is an Eclipse-based development environment in which you design, author, test, and deploy decision services
 - Rule Execution Server provides the runtime environment for running and monitoring decision services
 - Rule Execution Server is a web-based console interface for monitoring and managing Rule Execution Server activity

**Note**

Operational Decision Management includes Decision Server Insights for event-based decision making. Decision Service Insights is not covered in this course.

Operational Decision Manager: Roles and activities

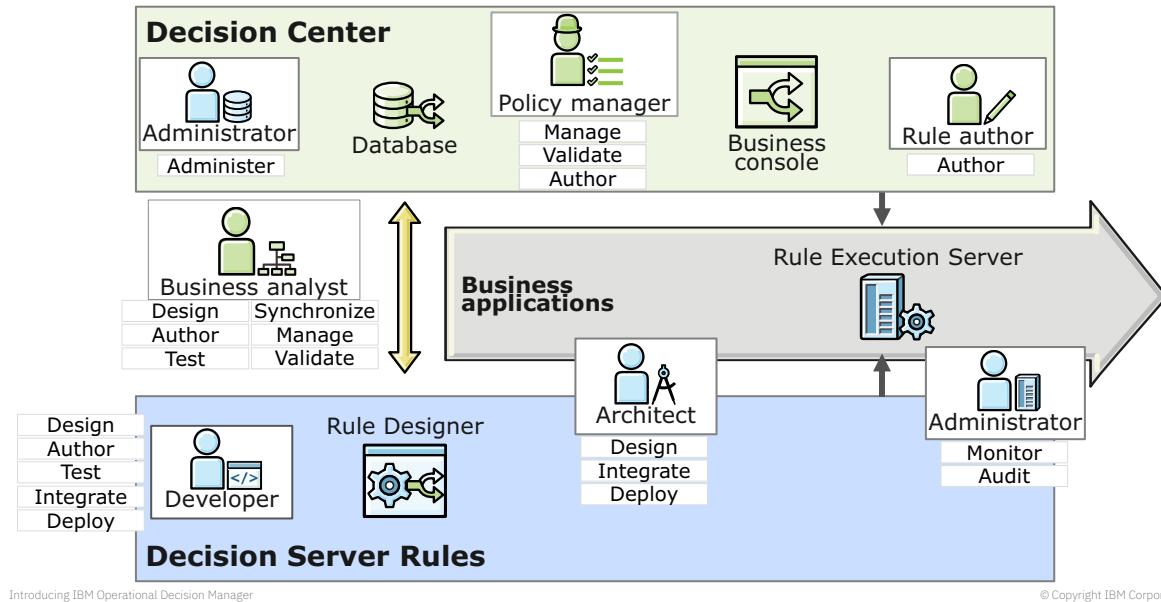


Figure 1-17. Operational Decision Manager: Roles and activities

The ODM modules are aimed at specific user roles, which are based on their varied skill sets.

This graphic shows an overview of the various activities carried out by these users and in which environment.

The rules are written in natural language, which makes it possible for business users can update and manage them directly. However, for these rules to be deployed and executed, developers need to set up the infrastructure for business authors to design and develop the decision logic. Decision Server is the technical environment where developers set up and maintain this infrastructure.

After the infrastructure is set up, distributed business teams can start working in Decision Center to create, maintain, and deploy decision services.

1.4. ODM portfolio



ODM portfolio

Figure 1-18. ODM portfolio

On-premises	On cloud	IBM Automation platform	System Z
<ul style="list-style-type: none"> Full-feature installation Continuous delivery model Long term release support support for V8.10.5 Windows, Linux, AIX, MacOS 	<ul style="list-style-type: none"> ODM on Cloud, IBM managed SAAS Fully hosted and managed on IBM public cloud ODM on Certified Kubernetes Enables private or hybrid cloud services ODM for Developers Decision Center and Decision Server deployed to a single Docker container 	<ul style="list-style-type: none"> Automation Decision Services Delivered as part of a global project that includes workflow, robotic process automation, content or data capture automation Available on-premises, on-cloud, and on multi-cloud 	<ul style="list-style-type: none"> z/OS, zLinux Native z/OS Execution (zRES) CICS/IMS Invocat

Introducing IBM Operational Decision Manager

© Copyright IBM Corporation 2021

Figure 1-19. Options for using ODM

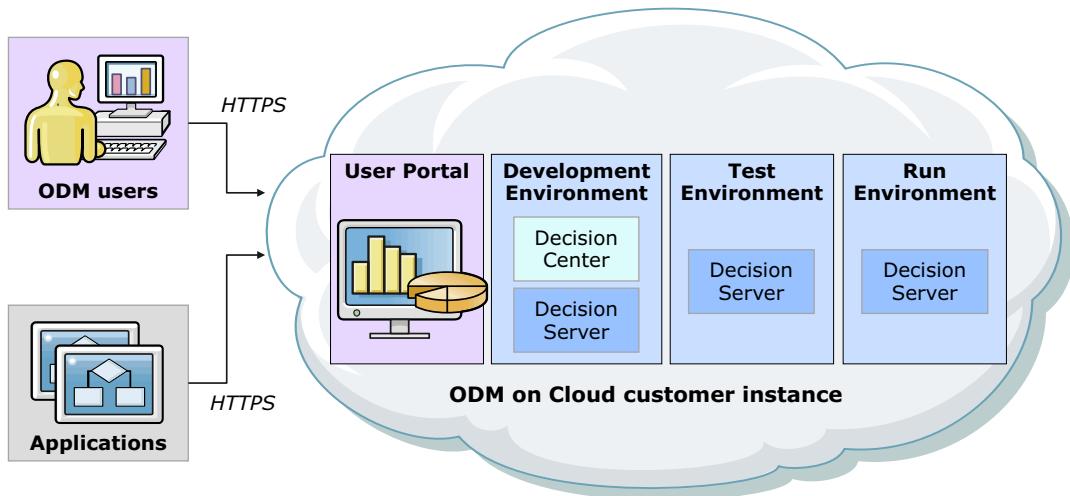
This slide outlines the various installation options for working with Operational Decision Manager.



Note

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

IBM ODM on Cloud: Three runtime environments



Introducing IBM Operational Decision Manager

© Copyright IBM Corporation 2021

Figure 1-20. IBM ODM on Cloud: Three runtime environments

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

Developing decision services in a hybrid cloud environment

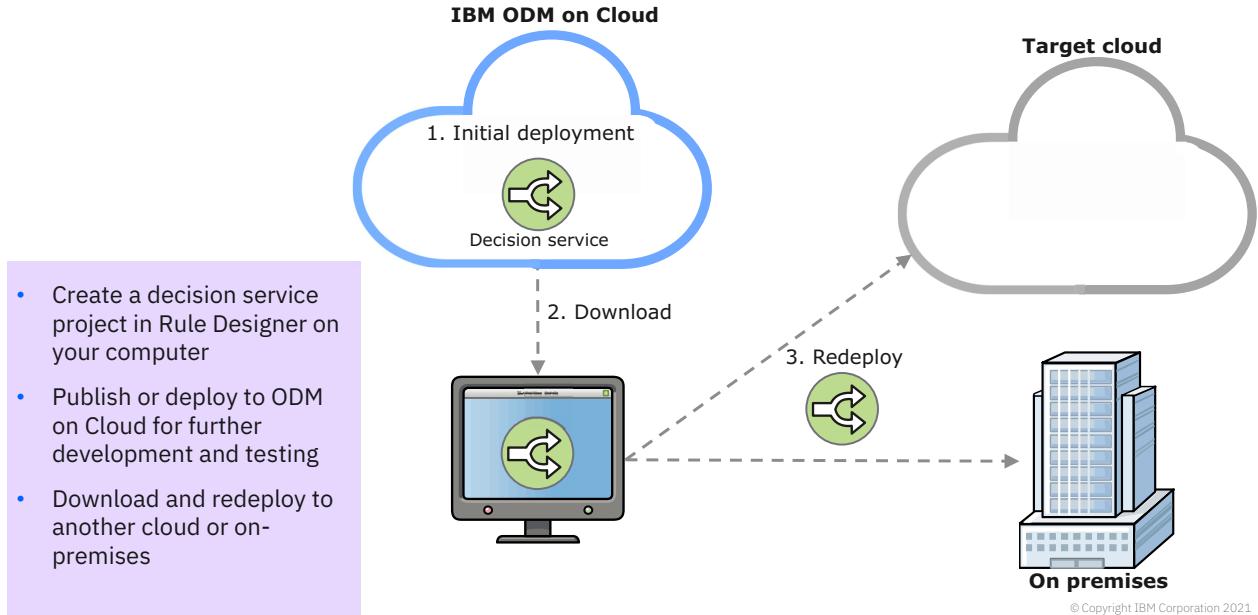


Figure 1-21. Developing decision services in a hybrid cloud environment

ODM on Cloud can also be part of a hybrid cloud solution. This diagram outlines a typical hybrid scenario where a decision service is created locally and deployed to ODM on Cloud for collaborative development and testing. Then, the executable assets are downloaded and redeployed to another instance of Decision Server that is running on premises or in another target cloud.

For more information about IBM ODM on Cloud, see [Appendix B, "ODM on Cloud"](#).

See also: www.ibm.com/support/knowledgecenter/SS7J8H/welcome/kc_welcome_cloud.html

ODM on Certified Kubernetes

- Use your infrastructure to host private cloud services
 - Create and provision multiple machines
 - Change computing resources on-demand
 - Scale applications
 - Self-service platform
- Fully integrates with IBM public cloud for hybrid cloud solution
- Includes Decision Center and Decision Server
 - Decision Server Insights is not included
- ODM for Developers includes Decision Center and Decision Server on a single container
 - See hub.docker.com/r/ibmcom/odm and <https://odmdev.github.io/odm-ondocker>

Introducing IBM Operational Decision Manager

© Copyright IBM Corporation 2021

Figure 1-22. ODM on Certified Kubernetes

You can work with an on a private or hybrid cloud solution by using ODM on Certified Kubernetes.

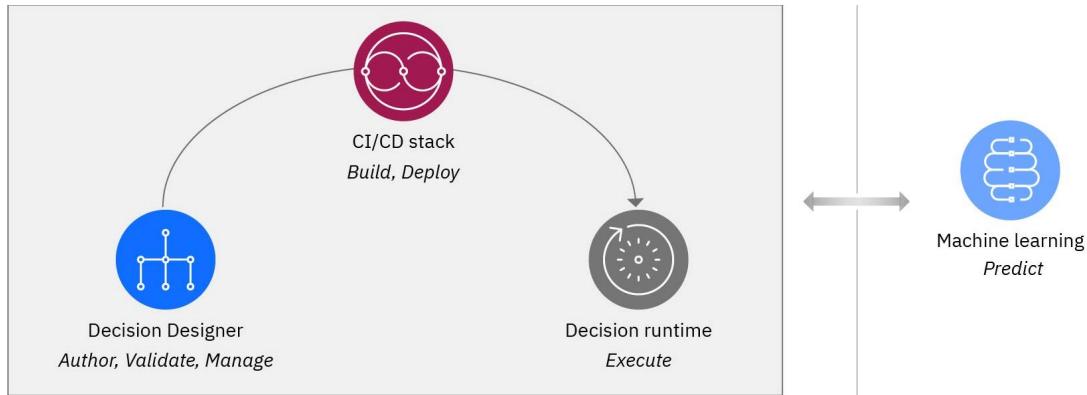
To get started developing rule applications, you can use an evaluation version of ODM, ODM for Developers, on a single Docker image.

For more information, see the Knowledge Center:

www.ibm.com/support/knowledgecenter/en/SSQP76_8.10.x/com.ibm.odm.kube/kc_welcome_odm_kube.html

Automation Decision Services

- Comprehensive environment for authoring, managing, and running decision services
- Delivered as part of the IBM Automation platform
 - Integration with Business Automation Studio for modeling, authoring, and validation tasks
 - Includes Decision Designer and Decision runtime



Introducing IBM Operational Decision Manager

© Copyright IBM Corporation 2021

Figure 1-23. Automation Decision Services

Automation Decision Services, which is delivered as part of the IBM Automation platform, provides a comprehensive environment for authoring, managing and running decision services. For more information, see the Knowledge Center:

www.ibm.com/support/knowledgecenter/en/SSYHZ8_20.0.x/com.ibm.dba.aid/topics/con_intro.html

1.5. ODM roles



ODM roles

Figure 1-24. ODM roles

This topic reviews the roles that are involved in an ODM solution.

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 2021

Figure 1-25. 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**



Introducing IBM Operational Decision Manager

Figure 1-26. 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: Rule Designer and Decision Center Business console



© Copyright IBM Corporation 2021

Figure 1-27. 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 console



© Copyright IBM Corporation 2021

Figure 1-28. Policy manager

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. They also work with rule authors in Decision Center to author, validate, and manage rules in the console.

Rule author

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



© Copyright IBM Corporation 2021

Figure 1-29. 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: Decision Server

© Copyright IBM Corporation 2021

Figure 1-30. 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.

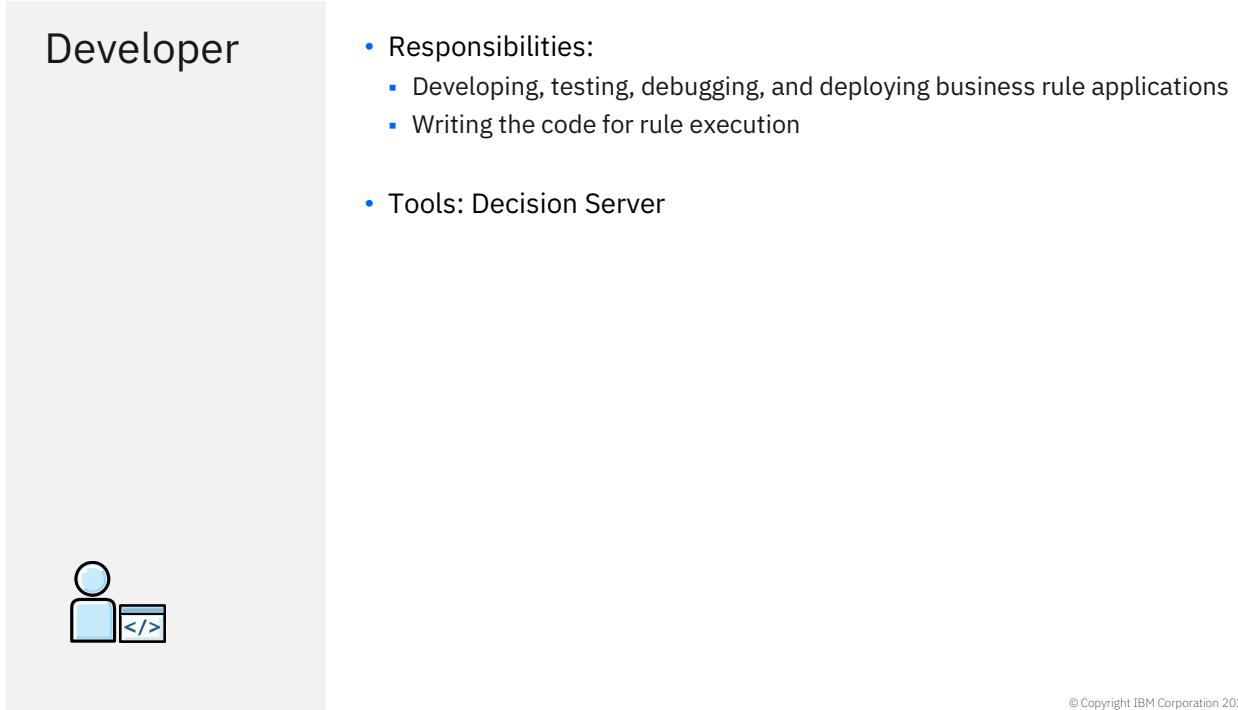


Figure 1-31. Developer

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



The slide features a light gray header bar with the word "Administrator" in bold black font. Below this is a white content area containing a bulleted list of responsibilities and tools. At the bottom of the slide is a decorative icon consisting of four blue icons: two stylized human figures, a cylinder representing storage or databases, and a server tower.

© Copyright IBM Corporation 2021

Figure 1-32. 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?

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 ODM roles?



Figure 1-33. Discussion: Who is in charge of what?

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.

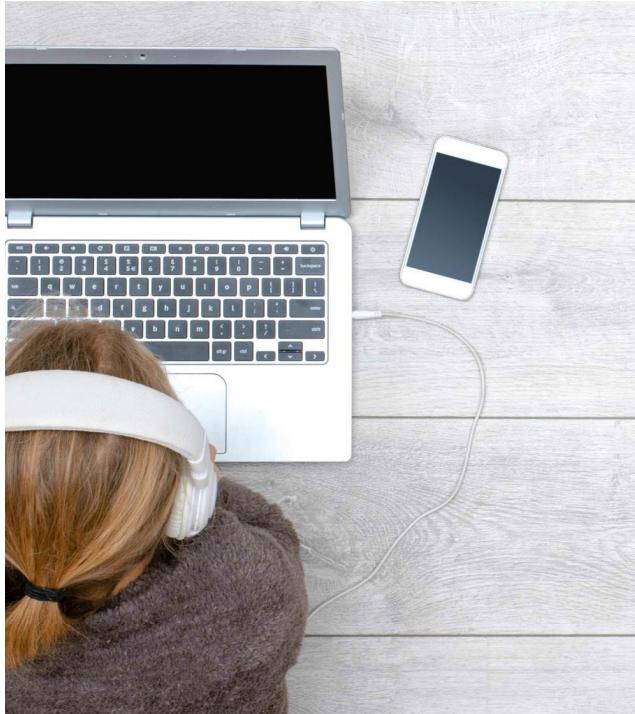
1. Who is in charge of capturing rules from the business policies?

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 ODM roles?

1.6. Governance and decision management



Governance and decision management

Figure 1-34. 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



Figure 1-35. 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.

In Operational Decision Manager, you can use the ready-to-use **decision governance 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 and the decision governance framework are specially designed to support rule authors who work in the Decision Center Business console.

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

Unit summary

- Explain the purpose of decision management
- Describe how the Operational Decision Manager architecture supports business and technical user roles and tasks
- Map the various roles that are involved in a decision management solution to roles in your organization
- Identify the need for governance

© Copyright IBM Corporation 2021

Figure 1-36. 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:** A financial institution wants to modify mortgage eligibility requirements. This task must be handled by the development team.
- 3. True or False:** An airline wants to offer a seasonal promotion on international flights, but such a change can only happen when the application infrastructure is updated, so it will take several months to implement.
- 4. Drag the words.** Put the rule vocabulary in the correct sequence to transform this policy into an if-then statement:

The minimum annual income for a borrower is \$10,000



Introducing IBM Operational Decision Manager

© Copyright IBM Corporation 2021

Figure 1-37. Review questions (1 of 2)

Write your answers here:

- 1.
- 2.



Review questions (2 of 2)

4. Select all that apply. Operational Decision Manager includes which components:

- a. Decision Server Rules
- b. Decision Server Insights
- c. Decision Center
- d. Business Automation Workflow

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

- a. Policy manager
- b. Business analyst
- c. Architect
- d. Developer

Figure 1-38. Review questions (2 of 2)

Write your answers here:

- 1.
- 2.

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 : A financial institution wants to modify mortgage eligibility requirements. This task must be handled by the development team.
The answer is False. With ODM, policy managers can modify the rules themselves with minimal dependence on you.
3. True or False: An airline wants to offer a seasonal promotion on international flights, but such a change can only happen when the application infrastructure is updated, so it will take several months to implement.
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 1-39. Review answers (1 of 2)



Review answers (2 of 2)

3. Drag the words. Put the rule vocabulary in the correct sequence to transform this policy into an if-then statement:

The minimum annual income for a borrower is \$10,000



4. Operational Decision Manager includes which components:

The answer is A, B, and C.

5. Which of the following roles are involved during the early stages of a business rule application development project?

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

Figure 1-40. Review answers (2 of 2)

Exercise: Operational Decision Manager in action



Figure 1-41. 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

© Copyright IBM Corporation 2021

Figure 1-42. Exercise introduction

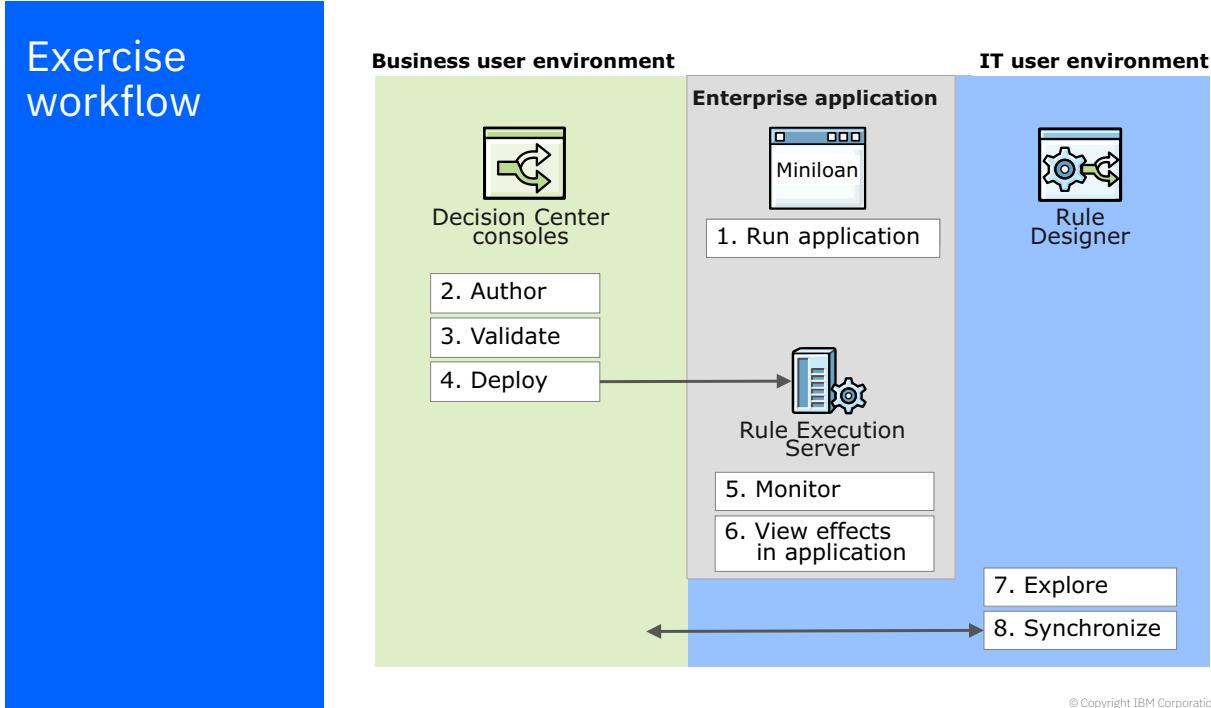


Figure 1-43. Exercise workflow

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

© Copyright IBM Corporation 2020

Figure 2-1. Unit objectives

Topics

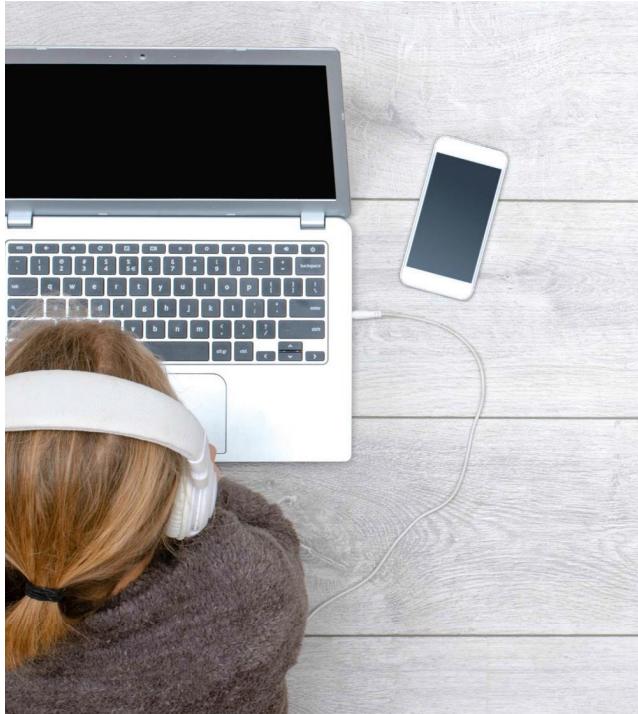
- Using an agile development approach
- Designing the business rule application
- Setting up decision services
- Project properties
- Using modular project organization
- Sharing and synchronizing decision services

© Copyright IBM Corporation 2020

Figure 2-2. Topics

This unit covers how to define a business rule, where to look for rules, and how rules are implemented with an agile approach by using Operational Decision Manager.

2.1. Using an agile development approach

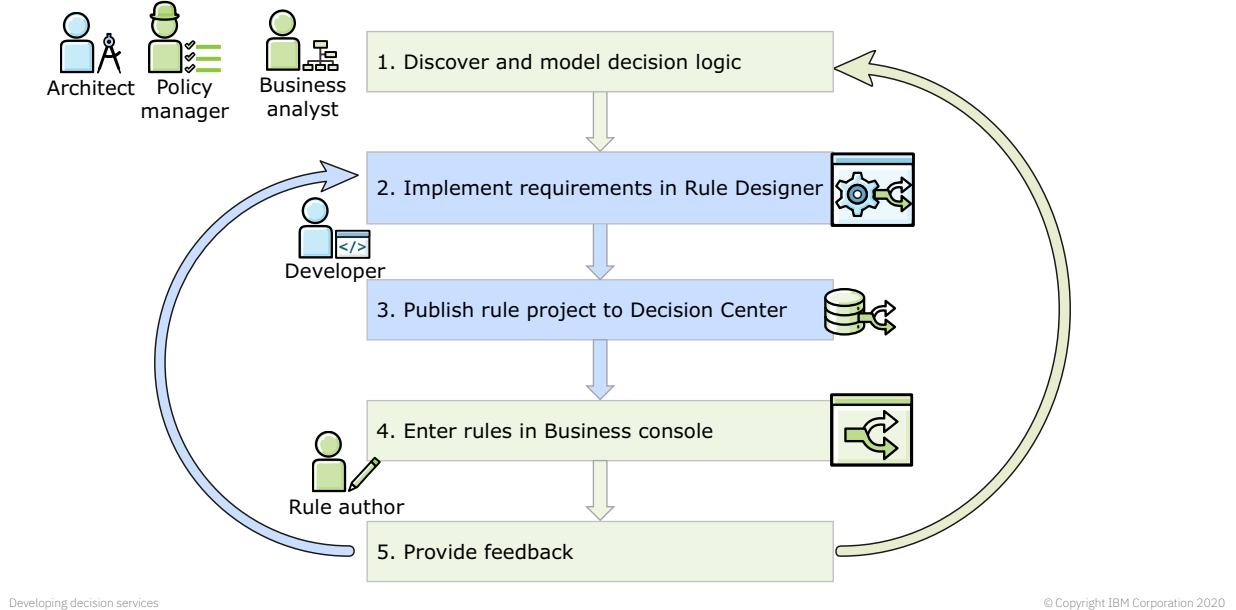


Using an agile development approach

© Copyright IBM Corporation 2020

Figure 2-3. Using an agile development approach

Using an agile development approach



Developing decision services

© Copyright IBM Corporation 2020

Figure 2-4. Using an agile development approach

Decision management projects require extensive collaboration between business and technical stakeholders in the project. Using an agile, or iterative, approach you can implement decision management incrementally.

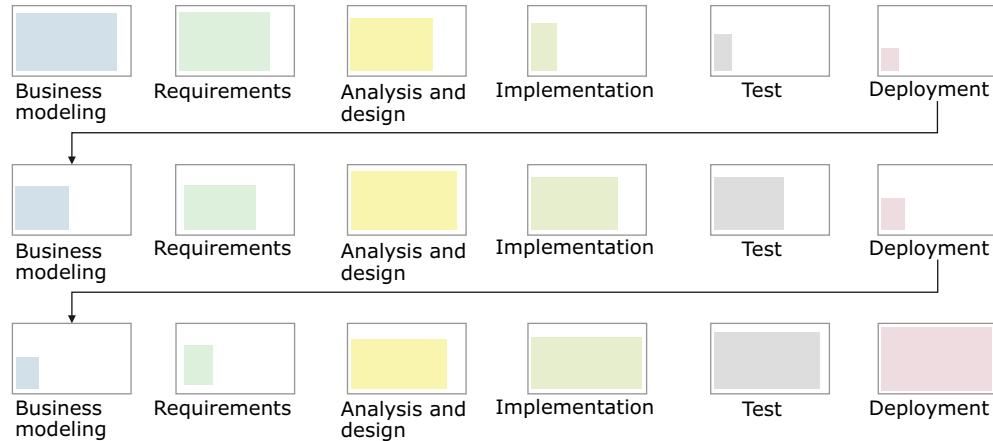
As shown on this slide, the overall process includes these steps:

1. Architects work with business users to determine or **discover** which decisions in the business process to implement and model the decision logic. The goal of this phase is to ensure that the logic is expressed correctly “on paper” before moving to the tools. The results from this phase become the application requirements that get passed to the development team.
2. Based on the models, initial rules, and discussion with the business analyst, you can start work in Rule Designer to design and set up the rule authoring environment.
3. When that environment is ready, it is published to Decision Center for the business users to begin testing. You can continue to work simultaneously with the business users on the same project, and periodically synchronize updates between Decision Center and Decision Server.
4. To ensure that the rules were discovered and analyzed correctly, the rule authors start entering rules into the Decision Center Business console as soon as possible.
5. As rule author work with the rules in the tools, they can also provide feedback on possible design or analysis issues. The rules might look good “on paper,” but trying to implement them in the tool can uncover missing vocabulary or new requirements.

By using an agile or iterative approach, the business users can provide **early** feedback on the project implementation. Early feedback allows the development team 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.

Agile Business Rule Development (ABRD)

- Agile Business Rule Development (ABRD) includes a set of phases and activities



Developing decision services

© Copyright IBM Corporation 2020

Figure 2-5. Agile Business Rule Development (ABRD)

The Agile Business Rule Development methodology, or ABRD, includes a sequence of phases that support an iterative and agile decision management implementation. Each phase includes iterations on a set of activities.

While there is some participation in each of the activities during each of the phases, the amount of effort on these activities shifts during the project.

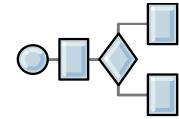
Initially, efforts are concentrated on modeling, requirements, analysis and design activities.

As the project evolves and requirements stabilize, more effort can be spent on implementation, testing, and deployment activities.

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

Discovery and modeling

- 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 reduce back- and-forth validation between development and business users



Developing decision services

© Copyright IBM Corporation 2020

Figure 2-6. Discovery and modeling

Before implementing decisions, you first must find, or *discover*, them from the various rule sources.

During the discovery phase, architects work with business analysts, policy managers, and other business experts to identify where decisions are used within your business process and which decisions need to be automated. 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 also 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 and IT can be an issue when implementing policy requirements in core business applications.



Information

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 sub-decisions, 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	Manual
Give personal information	-
System verifies information	Automated: <i>Is the information valid?</i>
System determines eligibility	Automated: <i>Is the client eligible?</i>
System determines price	Automated: <i>What price applies based on criteria?</i>
System returns quotation	-
Accept or reject quotation	Manual

Developing decision services

© Copyright IBM Corporation 2020

Figure 2-7. 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

Developing decision services

© Copyright IBM Corporation 2020

Figure 2-8. 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.

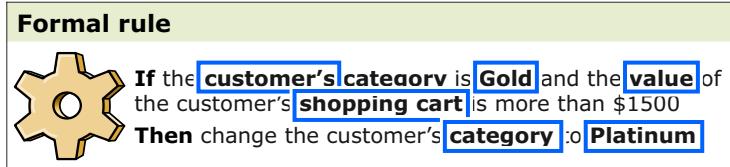
For example, when processing a loan request, the main business decision is whether to grant a loan to an applicant. When you break down that decision, you discover a series of other smaller decisions, such as:

- 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



- Ensure that all rule vocabulary is included in the object model

Developing decision services

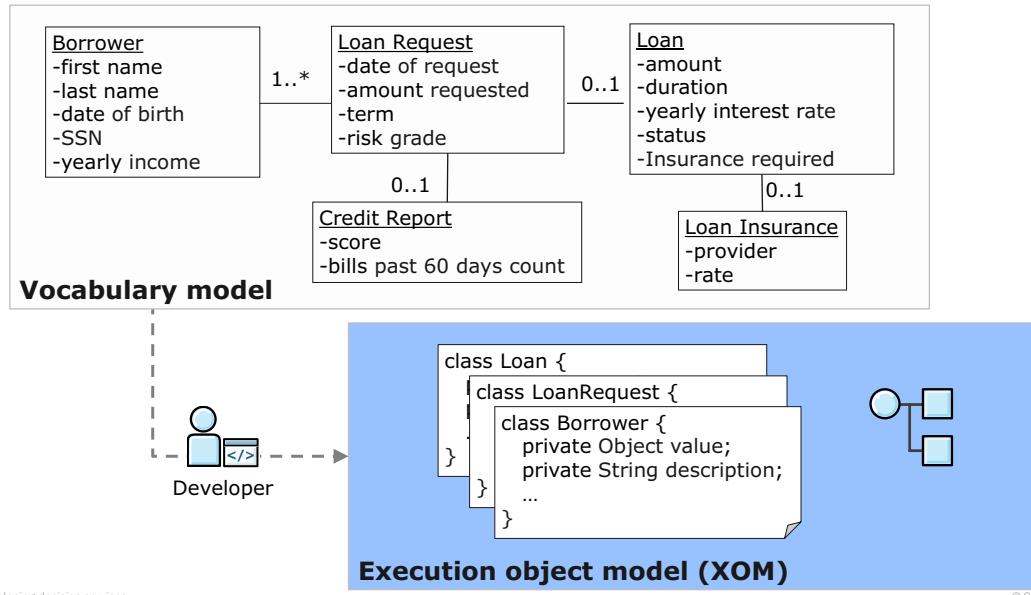
© Copyright IBM Corporation 2020

Figure 2-9. Discovery: Identifying vocabulary

Discovery also involves 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.

Implementing the model



Developing decision services

© Copyright IBM Corporation 2020

Figure 2-10. Implementing the model

Based on the object models from the business users, you can implement the models 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.

Rule discovery is intertwined with vocabulary and implementation of the object model. During initial discovery and analysis phases, you can already start working on the design of the application, which includes writing code to implement business object models. You set up the rule authoring environment so that early results from rule discovery activities can be implemented and tested.

Gaps in vocabulary models will show up when rule authors try to enter the rules in Decision Center. As developers, you can expect several iterations with business users to finalize the implementation of the object model.

2.2. Designing the decision service

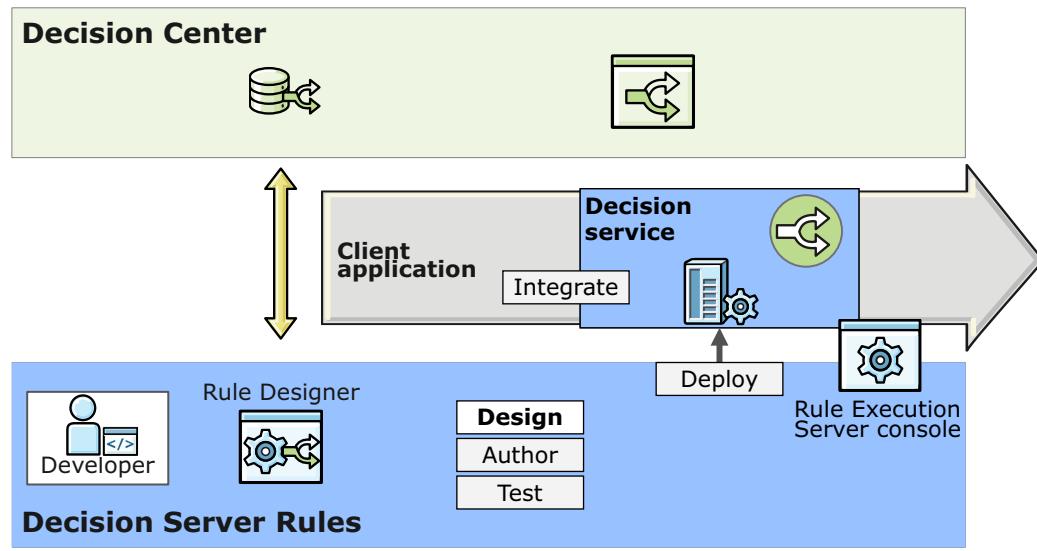


Designing the decision service

© Copyright IBM Corporation 2020

Figure 2-11. Designing the decision service

Design



Developing decision services

© Copyright IBM Corporation 2020

Figure 2-12. Design

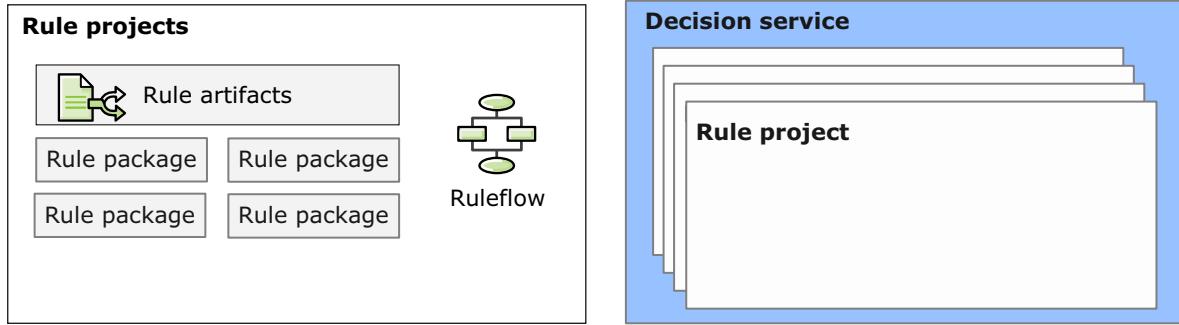
As a developer, your first task in a decision management project is put in place the necessary infrastructure for editing rules and producing an executable decision service.

This involves:

- Designing how the rules are organized and ensuring sequence in which they execute produces the correct decision
- Designing the execution models to implement the rule vocabulary in Java or XML
- Designing the application to integrate the business logic

Setting up the infrastructure

- Start with a rule project
 - Eclipse container for decision artifacts
- A decision service is a collection of rule projects
 - Incorporates governance features
 - All related rule projects are managed and deployed as a single unit



Developing decision services

© Copyright IBM Corporation 2020

Figure 2-13. Setting up the infrastructure

As a first step, you design the decision service structure. It all starts with a *rule project*.

A rule project is a type of Eclipse project that you use as a container for decision artifacts. Within the rule project, rules are further organized into rule packages.

As you see in the slide, in addition to the business rules, the rule project also contains the ruleflow, which outlines the general sequence in which business logic must be evaluated.

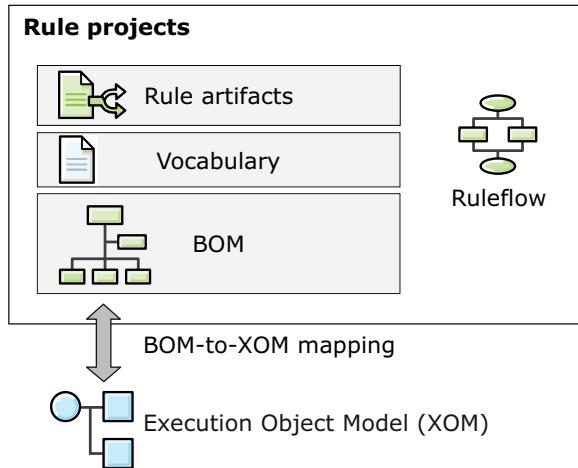
A *decision service* is a collection of rule projects. You design the project hierarchy so that the main rule project is the top-level project. Decision services incorporate governance features so that all related rule projects are managed and deployed as a single unit by means of the main rule project.

Implementing vocabulary models

Execution Object Model (XOM) is code implementation of vocabulary models

Business object model (BOM) is the business view of code

Vocabulary enables rule authoring



Developing decision services

© Copyright IBM Corporation 2020

Figure 2-14. Implementing vocabulary models

You implement the vocabulary models from the business users as code in an Execution Object Model (XOM). The business object model (BOM) is the business view of the code implementation. The vocabulary layer on the BOM makes rule authoring possible in the rule editors. To create the vocabulary, you can either generate a BOM directly from the XOM in Rule Designer, or you can manually map the BOM to the XOM.



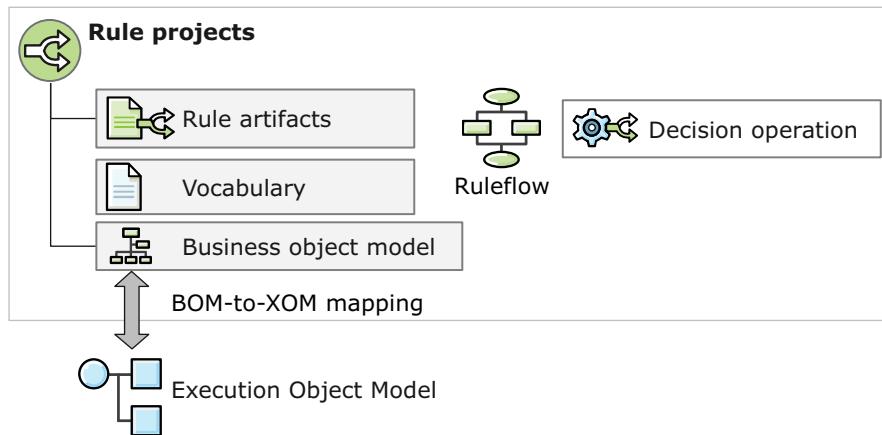
Note

You work with the BOM, XOM, and BOM-to-XOM mapping later in the course.

Extracting a ruleset

A ruleset is extracted from the finalized rule project

A decision operation defines ruleset content and signature



Developing decision services

© Copyright IBM Corporation 2020

Figure 2-15. Extracting a ruleset

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 single ruleset from multiple rule projects.

The ruleset uses input and output *parameters* to pass data to and from the client application. You define each ruleset with a unique *signature* of input and output parameters.

In decision services, you create a **decision operation** that defines the content of the ruleset and the ruleset signature.

You can create different decision operations that use the same rule projects but with different signatures. This facilitates reuse if new decision points are added.

To use a ruleset, a contract must exist between the application and the ruleset. The main elements of the contract are:

- Parameters—When the calling application sends a request for a decision service, it passes objects through the input parameters as defined by the signature. The decision result is returned to the calling application by the output parameters.
- BOM-to-XOM mapping—By means of the BOM-to-XOM mapping, the rules in the ruleset can manipulate the objects passed by the input parameters from the application.

2.3. Setting up decision services in Rule Designer



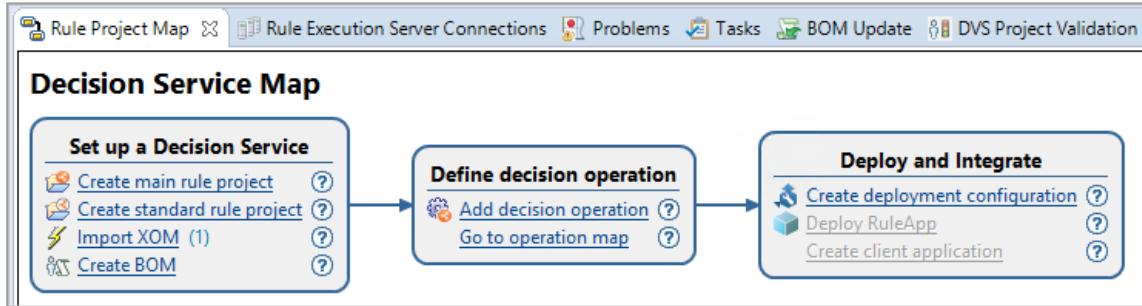
Setting up decision services in Rule Designer

© Copyright IBM Corporation 2020

Figure 2-16. Setting up decision services in Rule Designer

Decision service map

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



Developing decision services

© Copyright IBM Corporation 2020

Figure 2-17. 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:

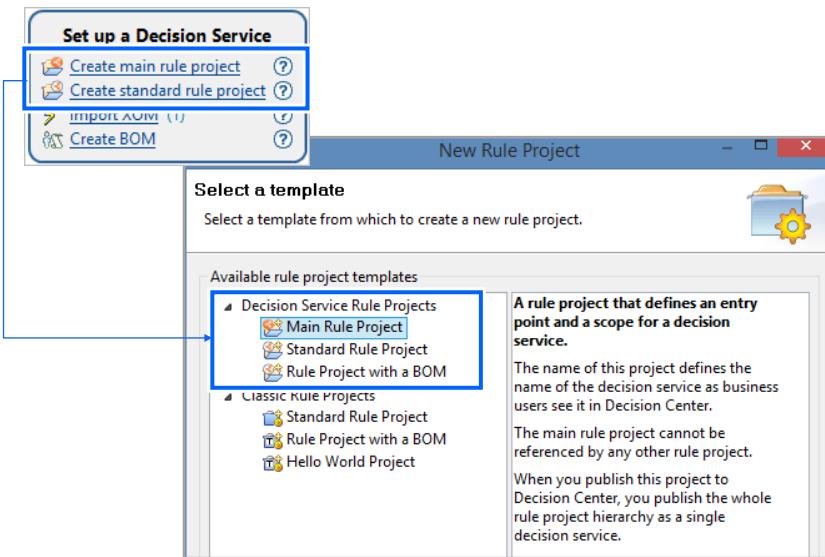
- Goals are represented as rounded *parts*.
- Tasks are represented as *links* in the goals of the map, and correspond to the steps that you saw previously.

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

Choose to create a **main** or **standard** rule project from the templates for Decision Service Rule Projects



Developing decision services

© Copyright IBM Corporation 2020

Figure 2-18. 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
rules	Stores rule artifacts, including action rules, decision tables, decision trees, and ruleflows
bom	Stores the business object model and the vocabulary that is used in the rules
deployment	Stores the decision operations and deployment configurations
queries	Stores queries that can be run to find certain rules and apply actions on those rules
resources	Stores any type of file that is not part of the rule model
templates	Stores templates (partially filled rules) that can serve as a starting point when creating rules

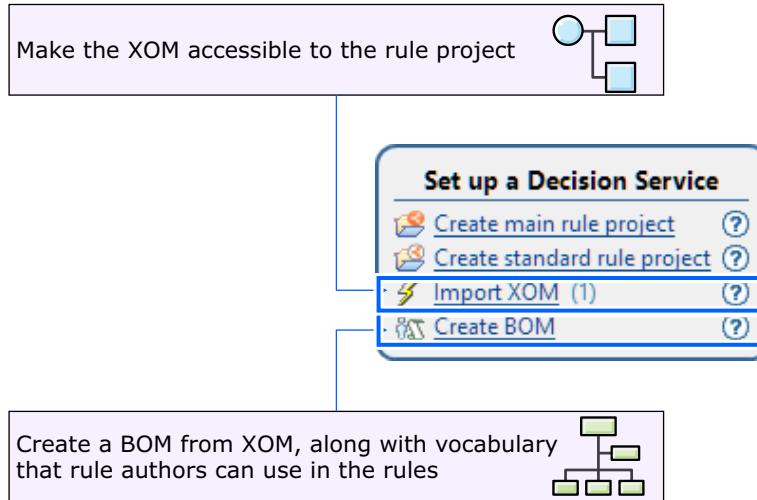
Developing decision services

© Copyright IBM Corporation 2020

Figure 2-19. 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



Developing decision services

© Copyright IBM Corporation 2020

Figure 2-20. 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.

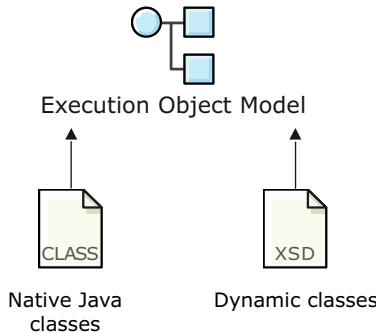
To accurately reflect the business perspective, developers write the code implementation of the object model to create the execution object model (XOM). You can then generate the BOM from the XOM.

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 2020

Figure 2-21. Design: XOM

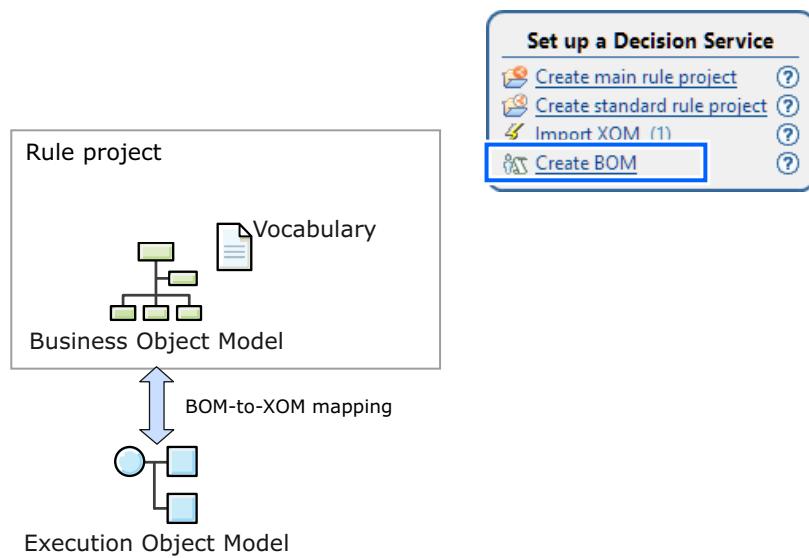
The business logic can be implemented by using either Java objects or an XML schema (XSD). As mentioned earlier, the implementation code is called the execution object model (XOM). The XOM makes rule execution possible. While business rules use the vocabulary from the BOM, each business element in a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

Creates a default BOM-to-XOM mapping and vocabulary



Developing decision services

© Copyright IBM Corporation 2020

Figure 2-22. 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

Decision Operation Overview - loan-operation

General

Name: loan-operation
Description:

Signature

Define the input and output parameters for the ruleset.

Input: borrower
Input - output: loan
Output: report

Ruleset Name

Enter the name of the ruleset part of the ruleset path that is called by Rule Execution Server.

Ruleset name: loan_ruleset

Ruleflow

Define the main ruleflow that is used as the entry point for execution.

Use main ruleflow: loanvalidation
 Do not use a ruleflow.

Source Rule Project

The operation can reference elements from the selected project and its dependencies.

Project: loanvalidation-rules

Business Rule Content

Select a subset of rules for the ruleset. By default, all the rules within the scope of the source rule project are selected. You can extract rules with a query or by using a validator.

Query: Browse...
Validator: Default Validator

Overview | Signature | Source |

© Copyright IBM Corporation 2020

Figure 2-23. Defining the decision operation

After the BOM is created, you create the decision operation to define the ruleset content.

Using the Operation Map

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

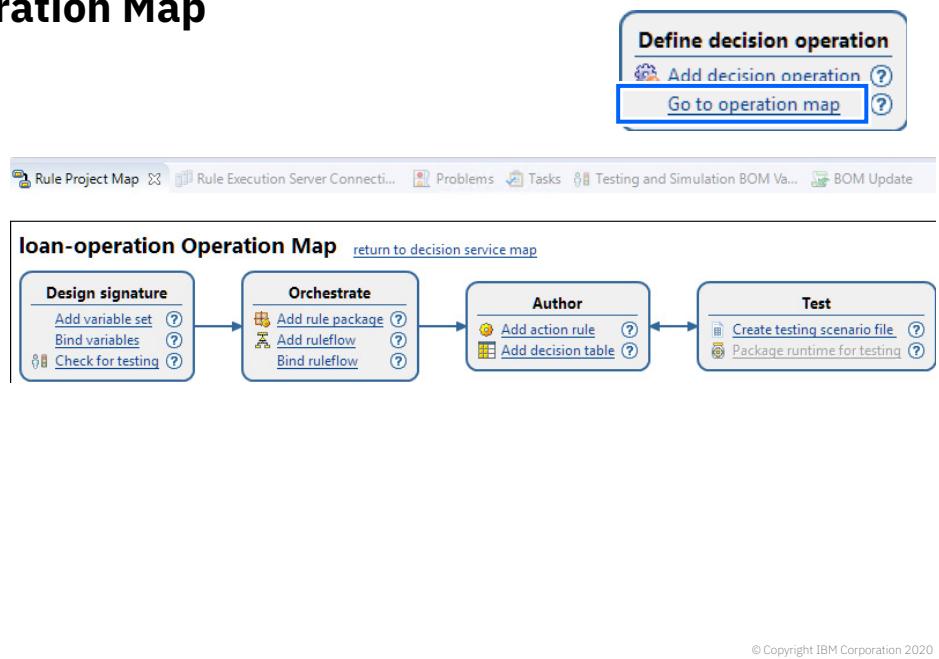


Figure 2-24. Using the Operation Map

From the Operation map, you can define all the artifacts that are required to make a decision, including the rules, ruleflow, rule variables, and parameters for the signature.

Designing the signature

Create a variable set to contains all your rule variables and parameters for the decision operation

Name	Type	Verbalization	Initial Value
borrower	loan.Borrower	the borrower	
loan	loan.Loan	the loan	
report	loan.Report	the loan report	

Developing decision services

© Copyright IBM Corporation 2020

Figure 2-25. Designing the signature

To define the ruleset signature, you first create the variables in a *variable set*.

The ruleset signature defines the parameters to pass objects between the calling application to the rule engine. These objects must be defined in both the BOM and XOM.

The rules do not handle the objects directly. During rule execution, the values of the parameters are manipulated through 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.

Designing the signature

Bind the variables to the ruleset parameters

Decision Operation Signature - loan-operation

Eligible variables
Select the ruleset variables that you want to use as parameters for the decision operation. Ruleset variables are defined in variable sets.

Input Parameters
Define the parameters required to call the execution.

Parameter name	Verbalization	Type	Initial Value
borrower	the borrower	loan.Borrower	

Input - Output Parameters
Define the parameters that are required, modified, and then returned by the execution.

Parameter name	Verbalization	Type	Initial Value
loan	the loan	loan.Loan	

Output Parameters
Define the parameters that are initialized and returned by the execution.

Parameter name	Verbalization	Type	Initial Value
report	the loan report	loan.Report	

Overview | Signature | Source

Developing decision services

© Copyright IBM Corporation 2020

Figure 2-26. 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

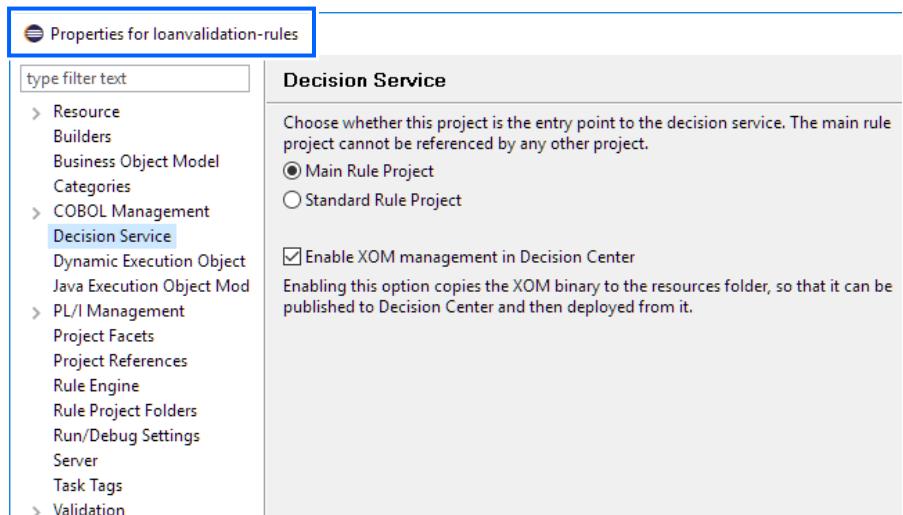
© Copyright IBM Corporation 2020

Figure 2-27. 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 2020

Figure 2-28. Properties: Project hierarchy

Project properties are defined in the Properties dialog box.

Properties: Folders

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

The screenshot shows the 'Properties for loanvalidation-rules' dialog box. On the left, there is a 'type filter text' input field and a list of project categories. In the center, under 'Rule Project Folders', there is a tree view of the project structure:

- BOM Folder
 - bom - /loanvalidation-rules
- Deployment Folder
 - deployment - /loanvalidation-rules
- Query Folder
 - queries - /loanvalidation-rules
- Resource Folder
 - resources - /loanvalidation-rules
- Source Folder
 - rules - /loanvalidation-rules
- Template Folder
 - path: none

Developing decision services

© Copyright IBM Corporation 2020

Figure 2-29. Properties: Folders

Properties: Project references

Specify project references to define dependencies between projects

The screenshot shows the 'Properties for loanvalidation-rules' dialog. On the left, a sidebar lists various project management categories: Resource, Builders, Business Object Model, Categories, COBOL Management, Decision Service, Dynamic Execution Object, Java Execution Object Mod, PL/I Management, Project Facets, and Project References. The 'Project References' item is highlighted with a blue border. The main panel contains a heading 'Project References' and a note: 'Projects may refer to other projects in the workspace. Use this page to specify what other projects are referenced by the project.' Below this, a section titled 'Project references for 'loanvalidation-rules'' shows a list with one item: 'loanvalidation-xom' preceded by a checked checkbox and a small icon.

Developing decision services

© Copyright IBM Corporation 2020

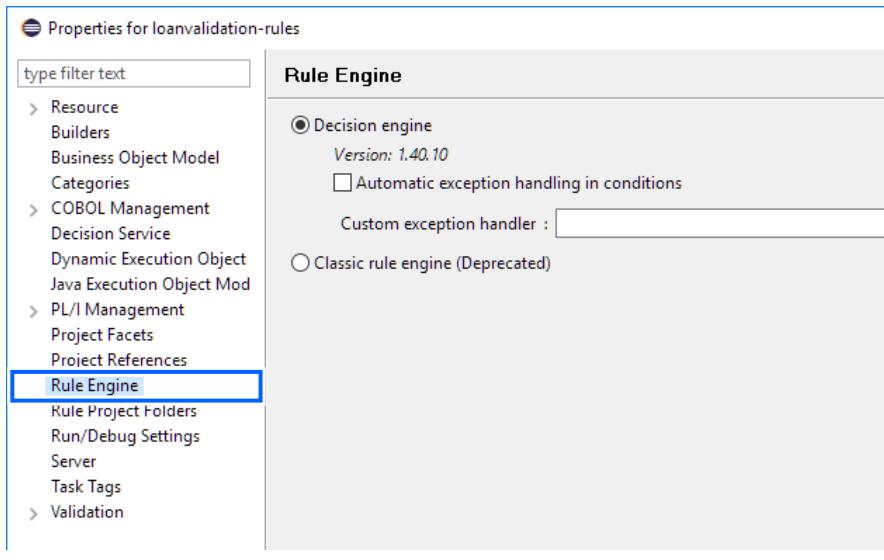
Figure 2-30. Properties: Project references

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 2020

Figure 2-31. 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



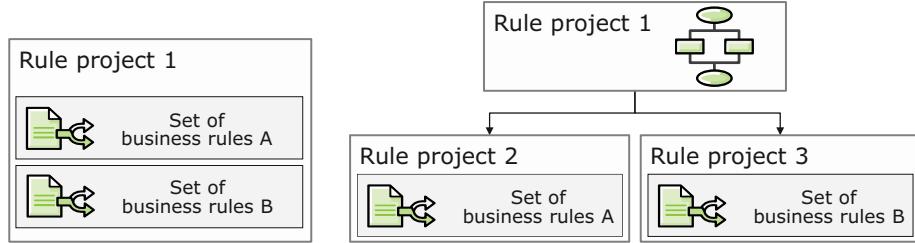
Using modular project organization

© Copyright IBM Corporation 2020

Figure 2-32. 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



Developing decision services

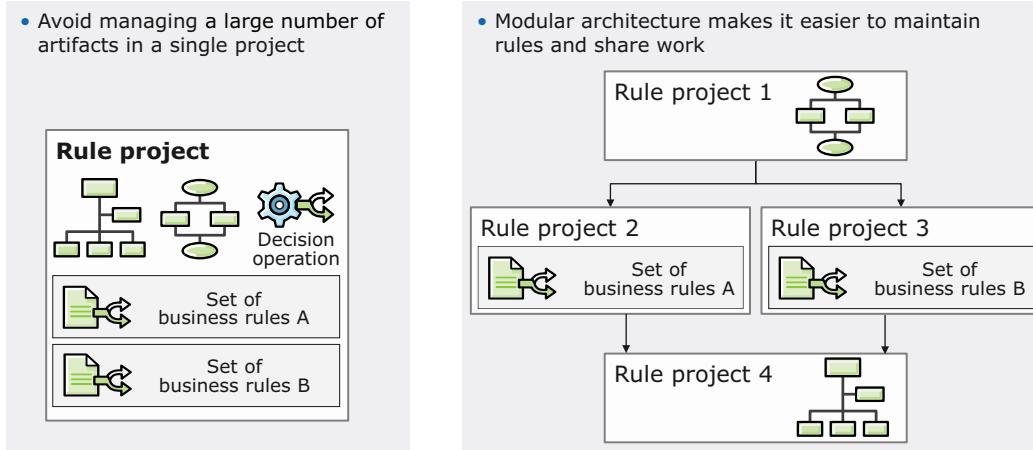
© Copyright IBM Corporation 2020

Figure 2-33. Modular structure (1 of 4)

Rule projects can reference each other, like Java projects. Rules in one project can be dependent upon rules or other artifacts in a separate project.

Modular structure (2 of 4)

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



Developing decision services

© Copyright IBM Corporation 2020

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

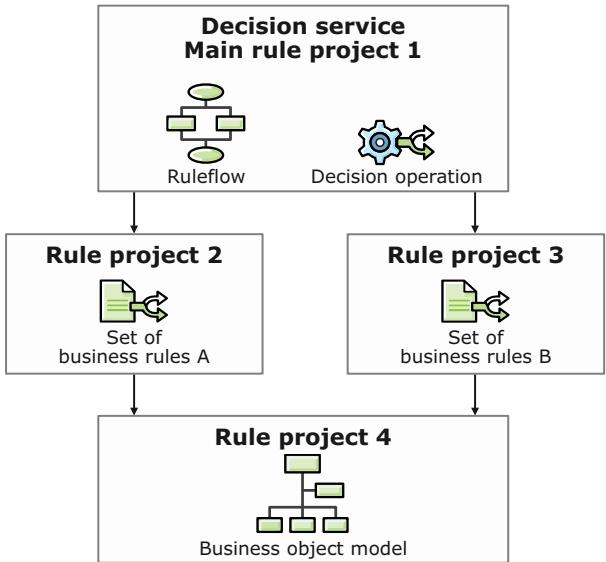
As the business rule application evolves, developers can set up multiple rule projects to handle large numbers of rules.

Dividing the business logic into logical segments makes it easier to maintain rules and share work among multiple users. For example, referencing other projects is a good way to share a common model between several projects that implement different decisions or to manage commonly used rules in a project.

A ruleset can be extracted from a single project or from a top-level rule project that references other rule projects.

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 2020

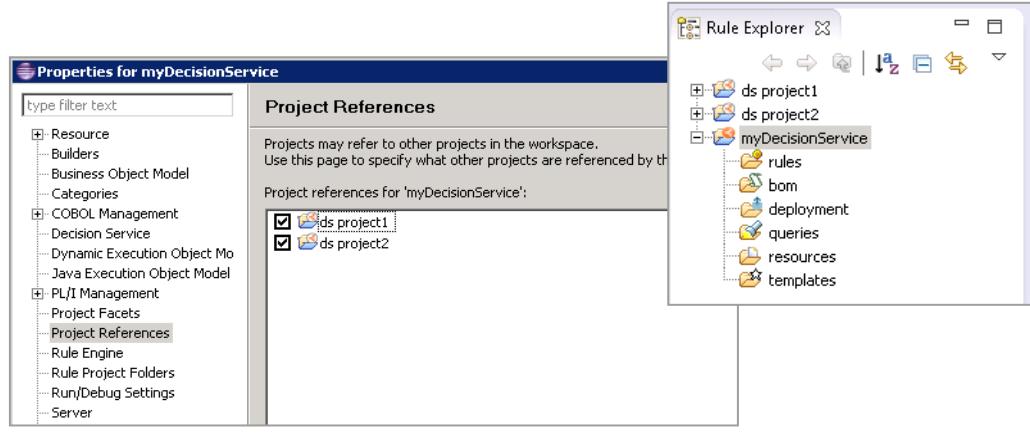
Figure 2-35. Modular structure (3 of 4)

The main decision service project defines the project hierarchy.

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 2020

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

If other rule projects are included in the decision service, they are referenced by the main project.

2.6. Sharing and synchronizing decision services



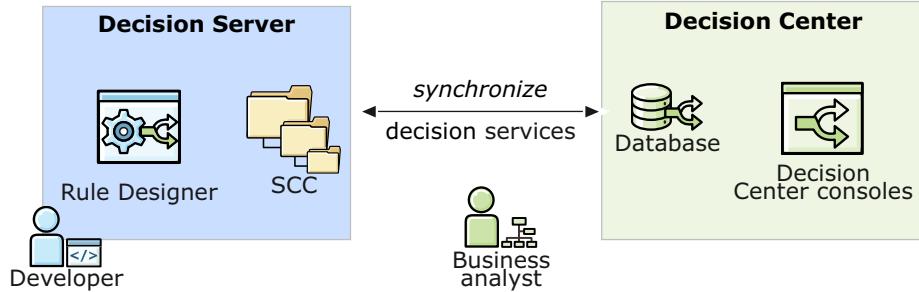
Sharing and synchronizing decision services

© Copyright IBM Corporation 2020

Figure 2-37. 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 2020

Figure 2-38. Synchronization between users

After a rule project is set up, it can be published from Rule Designer to Decision Center so that business users can also start working on the rules in Decision Center or Rule Solutions for Office.

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

- Business users store rule projects in the Decision Center repository, and access them through the Decision Center Business console or Enterprise console. They can also edit rules through Rule Solutions for Office. The Decision Center repository handles multi-user concurrency and version control.
- Developers store rules in a file system, and access them with Rule Designer and Event Designer. Developers work on copies of a rule project or event project in their Eclipse workspace, and keep a master copy in a source code control (SCC) system that handles file sharing, conflict resolution, and version management.

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

Developing decision services

© Copyright IBM Corporation 2020

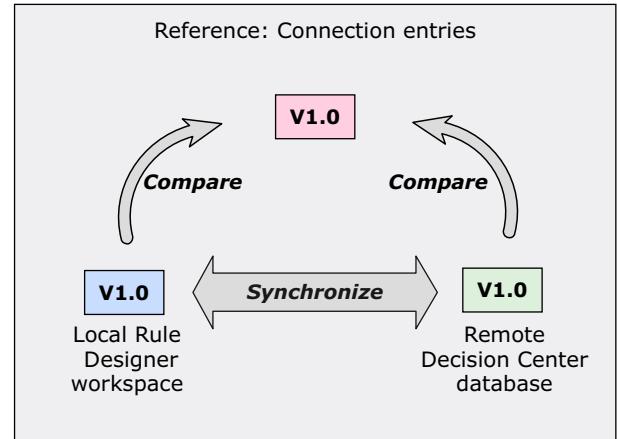
Figure 2-39. Synchronization tools

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 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 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
 - Compares them to the reference state



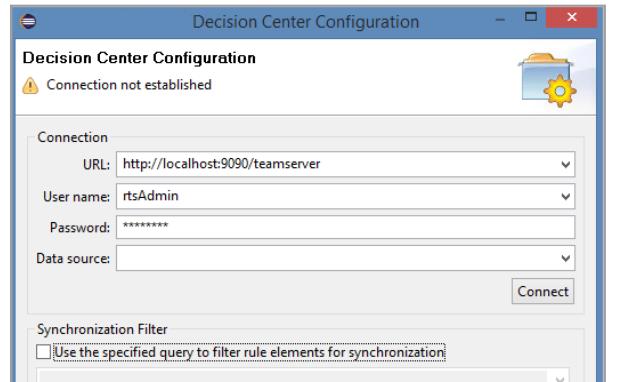
Developing decision services

© Copyright IBM Corporation 2020

Figure 2-40. Synchronization architecture

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

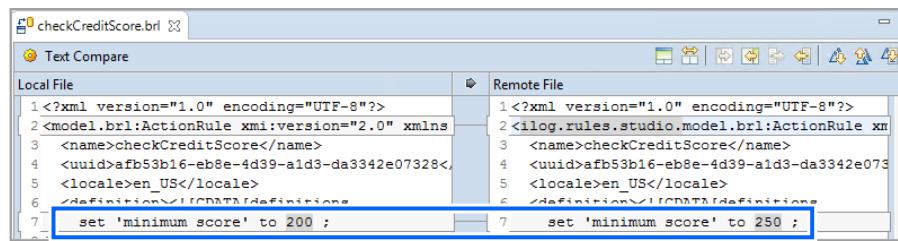
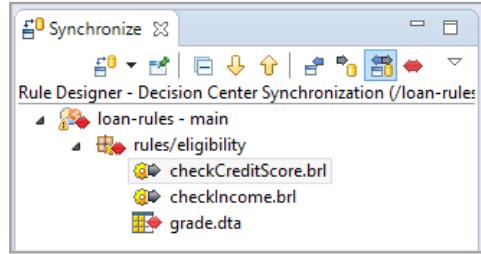
Developing decision services

© Copyright IBM Corporation 2020

Figure 2-41. Connection entries

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



Developing decision services

© Copyright IBM Corporation 2020

Figure 2-42. Choosing how to synchronize

Synchronization commands

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

Developing decision services

© Copyright IBM Corporation 2020

Figure 2-43. Synchronization commands

Choosing the master source

Business-user-centric approach

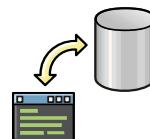
- Decision Center repository is considered master
- Repository provides rule management features, such as version control, baselines, and access control
- Copies that are stored through Designer are considered temporary
- Decision Center repository does not contain the artifacts that are required for rule execution, such as XOMs, .jar files, and libraries



Developing decision services

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



© Copyright IBM Corporation 2020

Figure 2-44. Choosing the master source

You need to establish which source to use as the master, either:

- Decision Center repository
- Source code control (SCC)

To avoid conflicts when rule projects are shared between business users and developers, you must establish clearly which source is the master: Decision Center or source code control (SCC).

Technical users synchronize their rule projects through SCC, and business users use Decision Center to synchronize their work. One dedicated technical user synchronizes 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

© Copyright IBM Corporation 2020

Figure 2-45. 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.

Developing decision services

© Copyright IBM Corporation 2020

Figure 2-46. 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.

Figure 2-47. Review answers

Exercise: Setting up decision services

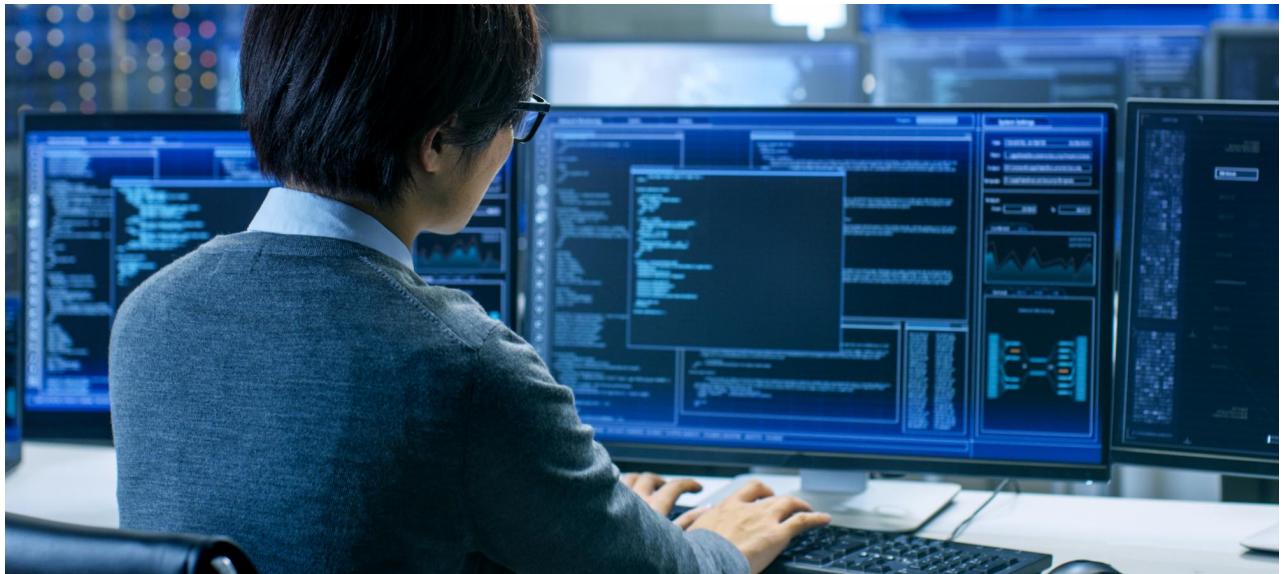


Figure 2-48. Exercise: Setting up decision services

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

© Copyright IBM Corporation 2020

Figure 2-49. Exercise introduction

Unit 3. Modeling decisions

Estimated time

00:30

Overview

This unit introduces decision modeling in Decision Center.

How you will check your progress

- Review
- Exercise

Unit objectives

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

© Copyright IBM Corporation 2020

Figure 3-1. Unit objectives

Topics

- Introducing decision modeling
- Creating decision models

© Copyright IBM Corporation 2020

Figure 3-2. Topics

3.1. Introducing decision modeling



Introducing decision modeling

© Copyright IBM Corporation 2020

Figure 3-3. Introducing decision modeling

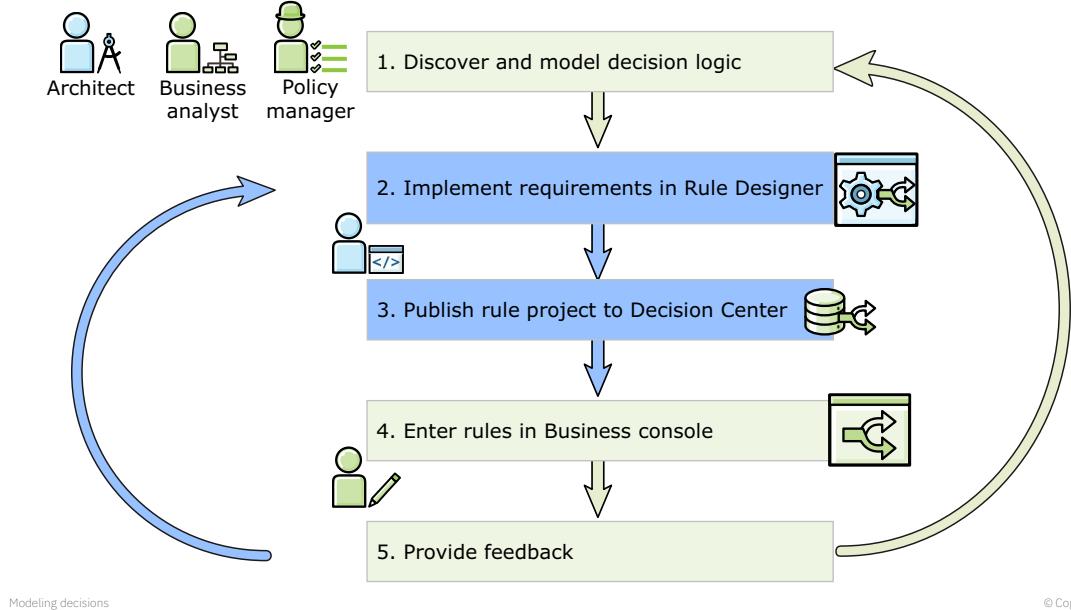


Note

Decision modeling in Decision Center is deprecated in this release. Modeling capabilities are migrated to Automation Decision Services in IBM Cloud Pak for Automation. For more information, see:

www.ibm.com/support/knowledgecenter/en/SSYHZ8_20.0.x/com.ibm.dba.aid/topics/con_aid_intro.html

Recall: Bottom-up approach



Modeling decisions

© Copyright IBM Corporation 2020

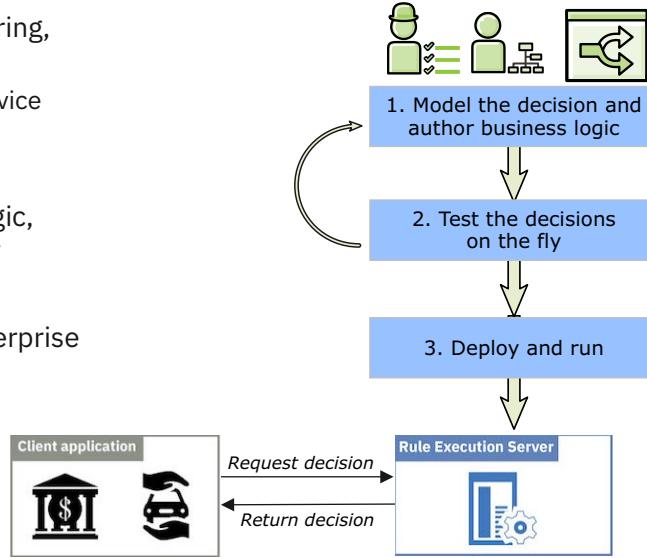
Figure 3-4. Recall: Bottom-up approach

In the previous unit, you learned how to support business users by setting up the rule authoring environment and creating the decision service for them and publishing it to Decision Center.

Decision Center also supports a top-down approach that allows business users to bypass IT (steps 2 and 3) by using decision model services.

Modeling decisions in Business console

- Enable business users to start discovering, modeling, and running decisions
 - Business users can create a decision service from scratch
- 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 2020

Figure 3-5. Modeling decisions in Business console

Business users can build a decision model that captures the required data and business logic and immediately test and run that model.

The model can also be deployed as a decision service and run against real data in a production environment.

Choosing the top-down decision model approach

- Business users
 - 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
- Developers
 - Useful for simple decisions
 - Use as a template to capture the requirements of the underlying decision service model in preparation for larger projects

Modeling decisions

© Copyright IBM Corporation 2020

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

The main reason for choosing a decision model service is to make the business expert completely autonomous in 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.

The decision model service is a simplified version of a decision service, and should be used for simple decisions.

As a developer, you can use 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. Use it as a **template** to build a regular decision service in Rule Designer, and 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

Decision models don't use a BOM, which means they don't require a Java or XML XOM to run. Your model structure also defines the order of rule execution so no ruleflow is required.

3.2. Creating decision models



Creating decision models

© Copyright IBM Corporation 2020

Figure 3-8. Creating decision models

Modeling in Business console

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

The screenshot shows the Decision Center interface with the following elements:

- Header:** DecisionCenter, HOME, LIBRARY (highlighted with a blue box), WORK.
- Navigation:** Get Started, Recent Activities, Stream.
- Welcome Section:** 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. (This section is highlighted with a blue box).
- Work with your decisions:** Go to the Library to view, edit and manage your decisions. Learn more about managing your decisions from the overview.

Modeling decisions

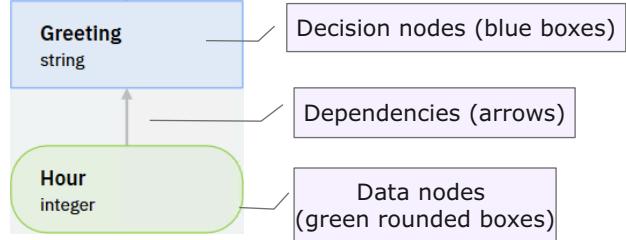
© Copyright IBM Corporation 2020

Figure 3-9. Modeling in Business console

For this course, you work with Decision Composer in Business console.

Creating the decision model diagram (1 of 2)

- To model a decision:
 - 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)



Modeling decisions

© Copyright IBM Corporation 2020

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

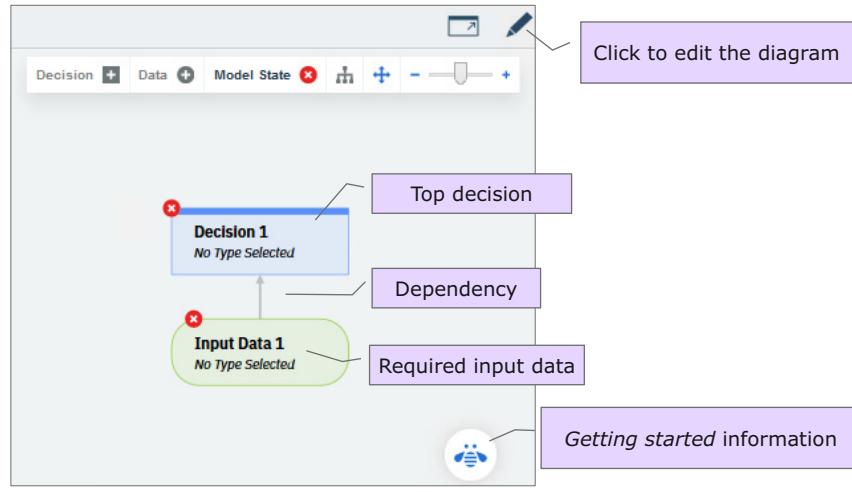
To model a decision, you model the node structure and author the decision logic.

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).

Creating the decision model diagram (2 of 2)

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



Modeling decisions

© Copyright IBM Corporation 2020

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

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

As a rule, the diagram 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

- Decision 1
- Details
- Description (optional): Describe the node (optional)
- Output variable name: Decision 1
- Same as decision node
- Output type: Make a selection
- A data type is required
- No tables and rules added yet

Data node definition

- Input Data 1
- Details
- Description (optional): Describe the node (optional)
- Output variable name: Input Data 1
- Same as data node
- Output type: Make a selection
- A data type is required
- Please select a type first.

- You can choose default data types or create custom types

Modeling decisions

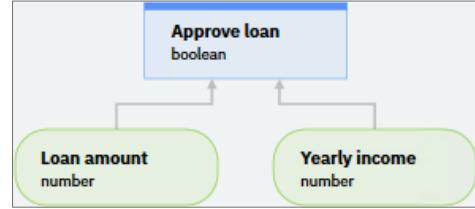
© Copyright IBM Corporation 2020

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

When you create decision nodes and data nodes, you must define a name and data type.

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



Modeling decisions

© Copyright IBM Corporation 2020

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

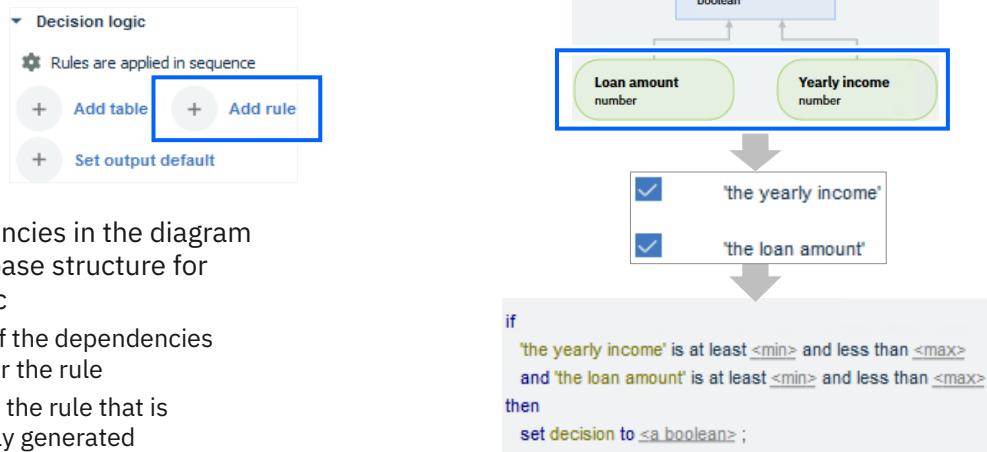
As you begin modeling, you start from the final or main decision at the top and work down to capture the input required to produce a decision.

Consider an example decision model for loan approval. You could start the model with the main **Approve loan** decision at the top. At a minimum, this main decision is dependent on the loan amount requested and the borrower's yearly income. 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, or what terms apply to the loan for repayment. Each sub-decision 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 2020

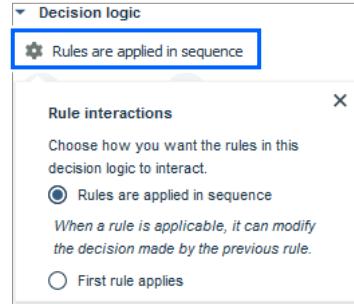
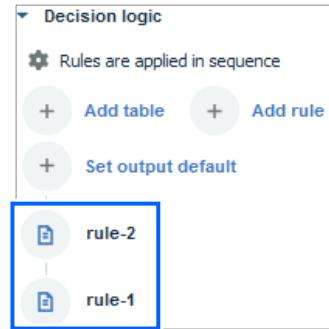
Figure 3-14. Authoring decision logic (1 of 2)

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.

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 2020

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

Defining custom types

- Define custom types on the **Types** tab
 - Create types to define objects

The screenshot shows the 'Types' tab in the IBM Decision Modeler interface. A 'person' type is defined with attributes name, age, and address. A 'category' enumeration type is also shown with values beginner, intermediate, and advanced.

Name	Type	List
name	string	<input type="checkbox"/> Delete
age	number	<input type="checkbox"/> Delete
address	string	<input type="checkbox"/> Delete

Name
category

Values

Value	Delete
beginner	<input type="checkbox"/>
intermediate	<input type="checkbox"/>
advanced	<input type="checkbox"/>

Modeling decisions

© Copyright IBM Corporation 2020

Figure 3-16. Defining custom types

With custom types, you can create objects that can be used as output types for decision nodes and data nodes.

Unit summary

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

© Copyright IBM Corporation 2020

Figure 3-17. Unit summary

Review questions



1. True or False: Business users can use decision models to create decision services in Decision Center Business console.
2. True or False: Decision model services cannot be used in a production environment with real data.
3. True or False: Developers do not work with decision models.

Modeling decisions

© Copyright IBM Corporation 2020

Figure 3-18. Review questions

Write your answers here:

- 1.
- 2.
- 3.

Review answers



- True or False: Business users can use decision models to create decision services in Decision Center Business console.

The answer is True.

- True or False: Decision model services cannot be used in a production environment with real data.

The answer is False. Decision model services are simplified decision services but they are fully consumable by enterprise applications.

- True or False: Developers do not work with decision models.

The answer is False. Developers can use decision models as a template to understand and capture requirements for building a more comprehensive decision service in Rule Designer.

Figure 3-19. Review answers

Exercise: Modeling decisions



Figure 3-20. Exercise: Modeling decisions

Exercise introduction

- Create a model diagram
- Define the decision and data node structure
- Create custom data types
- Author business logic in decision modeling language
- Test the model

© Copyright IBM Corporation 2020

Figure 3-21. 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



Modeling decisions

© Copyright IBM Corporation 2020

Figure 3-22. Exercise overview

Unit 4. Programming with business rules

Estimated time

00:45

Overview

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

How you will check your progress

- Review

Unit objectives

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

© Copyright IBM Corporation 2020

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

© Copyright IBM Corporation 2020

Figure 4-2. Topics

4.1. Introducing ruleset integration and execution

In this topic, you are introduced to rule execution and associated concepts.



Introducing ruleset integration and execution

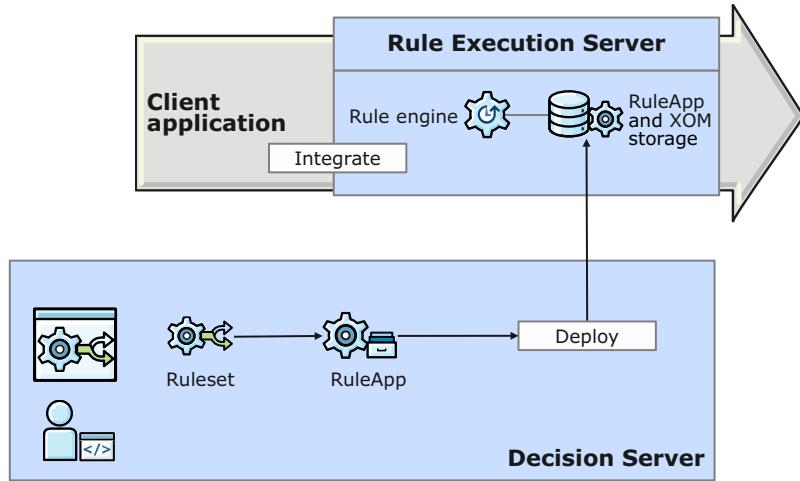
Figure 4-3. Introducing ruleset integration and execution

Integrating business logic

Package rulesets in a deployable management unit called a **RuleApp**

Deploy the RuleApp to Rule Execution server to make rulesets available to rule engine

Integrate by writing code in the client application to send ruleset execution requests to the rule engine



Programming with business rules

© Copyright IBM Corporation 2020

Figure 4-4. Integrating business logic

To make the rules available to the rule engine, you first package them as a ruleset in a deployable management unit called a **RuleApp**. A RuleApp can contain several rulesets.

To use decision logic in your application, you must **deploy and integrate** the ruleset for execution by the rule engine in your enterprise environment.

You **deploy** the RuleApp to Rule Execution Server to make the ruleset available to the rule engine. Deployment can be done in various ways, including from Rule Designer or from Decision Center.

You **integrate** the execution of your rule by writing the code in your client application that sends decision service requests to the rule engine. The rule engine executes the ruleset and returns the decision.

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

Programming with business rules

© Copyright IBM Corporation 2020

Figure 4-5. Execution environments

The rule engine can either be embedded directly in your client application or managed within the Rule Execution Server.

A common usage of an embedded rule engine is for conducting local tests. For example, you use this option when you debug your rules in 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 rule engine?

In this topic, you are introduced to rule execution and associated concepts.

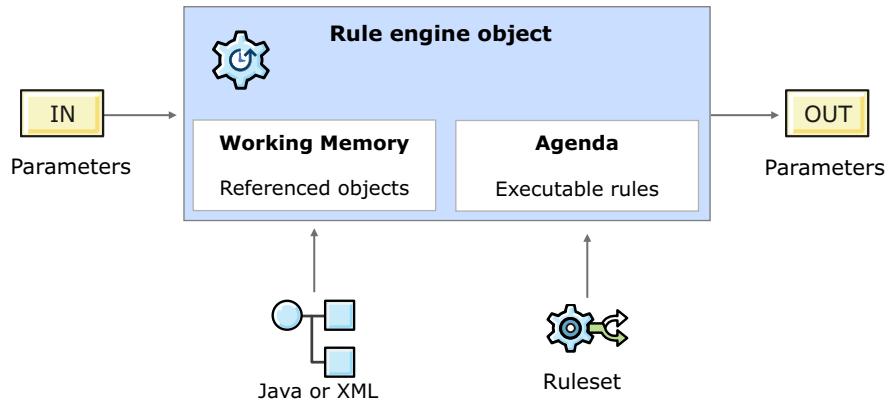


What is a rule engine?

Figure 4-6. What is a rule engine?

Engine

The engine is the module that executes the rules



Programming with business rules

© Copyright IBM Corporation 2020

Figure 4-7. 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:

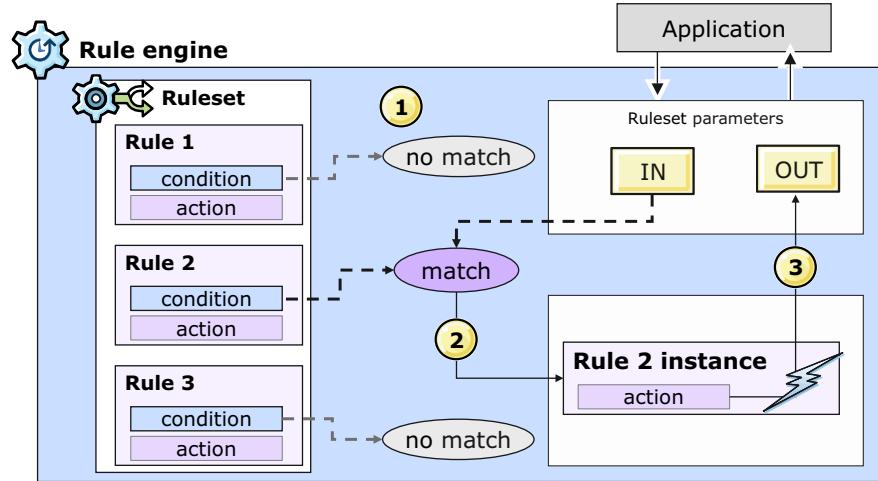
- References to the application objects are added to the rule engine. Through the XOM, the rule engine accesses the application objects. The rule engine uses these references to monitor the application objects.
- The rule engine processes the rules that you provide. It evaluates the rules against the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

Rule execution by the rule engine

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



Programming with business rules

© Copyright IBM Corporation 2020

Figure 4-8. 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:

1. 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*.
2. The rule engine creates a *rule instance* for each match between a business rule and an object or group of objects.
3. 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

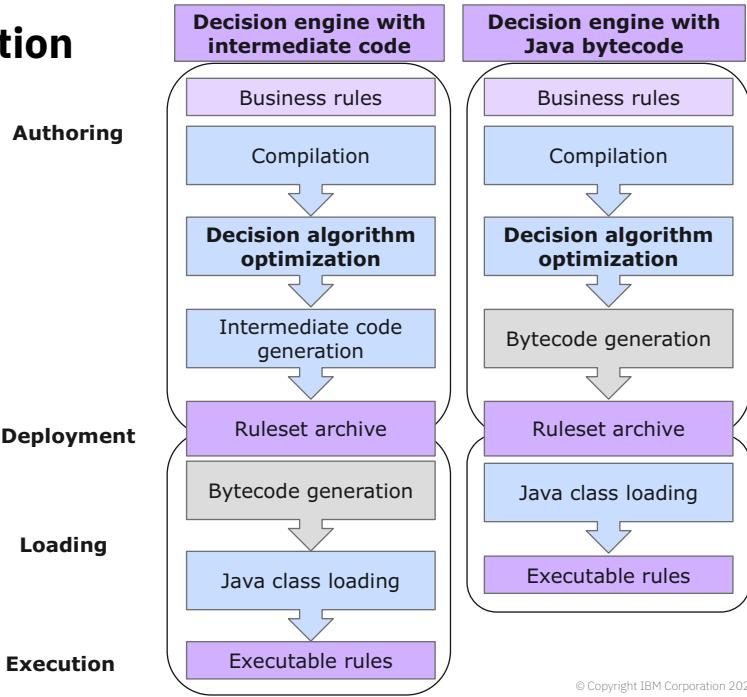
- A decision engine is an instance of the Engine interface
 - An application can contain several rule engine objects
- Use the decision engine API to define the loading of ruleset archives into an engine loader, which represents the result of compilation of a ruleset
- Decision Server class library includes 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 4-9. Decision engine API

Compilation and execution

In Rule Designer and Decision Center, the bytecode generation option is selected by default

In Decision Center, you must have enabled the XOM management to be able to activate the bytecode generation



Programming with business rules

© Copyright IBM Corporation 2020

Figure 4-10. Compilation and execution

The decision engine compiles rule artifacts into an archive that contains compiled and optimized code that becomes executable when translated to Java bytecode.

Ruleset loading in the decision engine is fast because no code is parsed or interpreted at run time. All the code is already compiled (to intermediate code or Java bytecode) and fully optimized for rule execution.

https://www.ibm.com/support/knowledgecenter/en/SSQP76_8.10.x/com.ibm.odm.dserver.rules.designer.run/executing_decision_topics/con_decision_engine.html

4.3. Understanding rule execution modes



Understanding rule execution modes

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

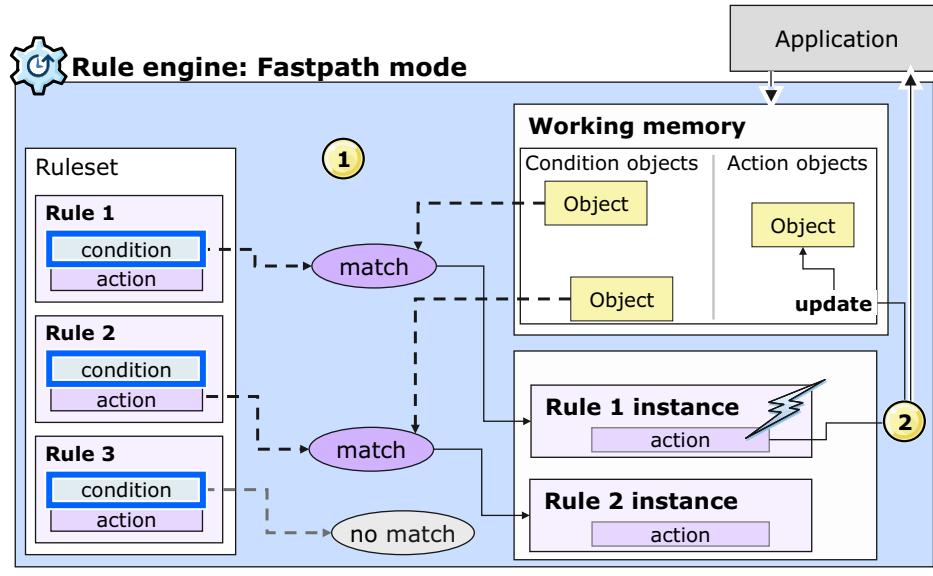
Programming with business rules

© Copyright IBM Corporation 2020

Figure 4-12. Execution modes

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

Execution modes: Fastpath (1 of 2)



Programming with business rules

© Copyright IBM Corporation 2020

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

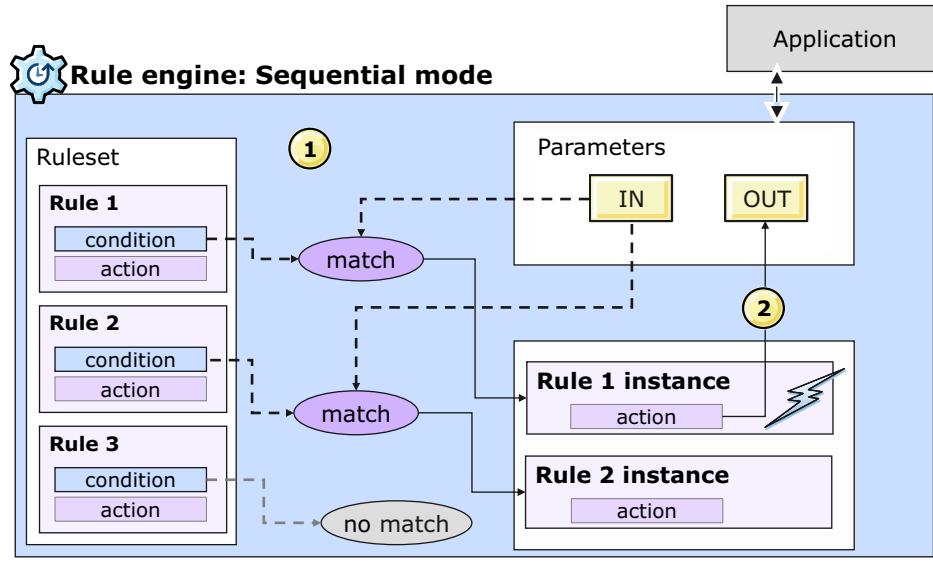
Programming with business rules

© Copyright IBM Corporation 2020

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 2020

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

Programming with business rules

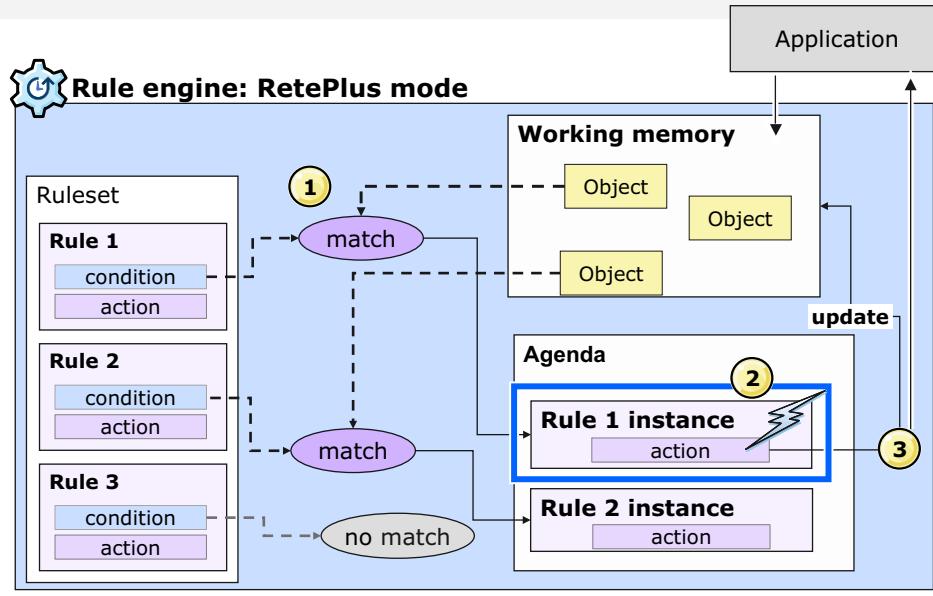
© Copyright IBM Corporation 2020

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 2020

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)

Programming with business rules

© Copyright IBM Corporation 2020

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

Programming with business rules

© Copyright IBM Corporation 2020

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.

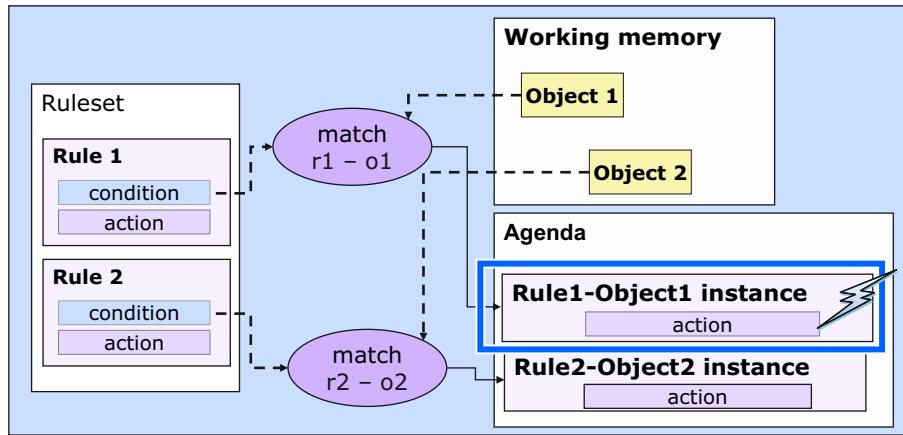
The agenda then orders rule instances and selects the one to fire first, based on the *refraction* and *recency* principles, in this order. The principle of refraction can be used to prevent loops. Recency is used to resolve conflicts when several rule instances are candidates for firing at the same 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 2020

Figure 4-21. Refraction (1 of 2)

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

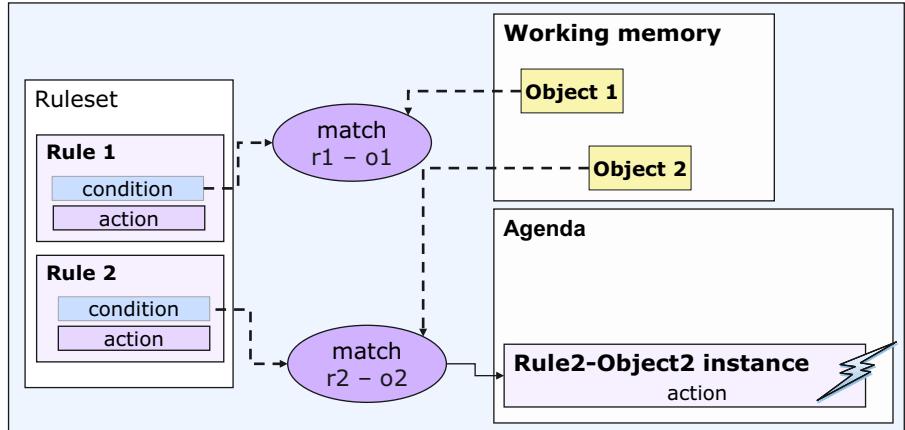
- Matched objects are not modified
- Matched objects are modified in such a way that the rule condition continues to hold true

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 2020

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.

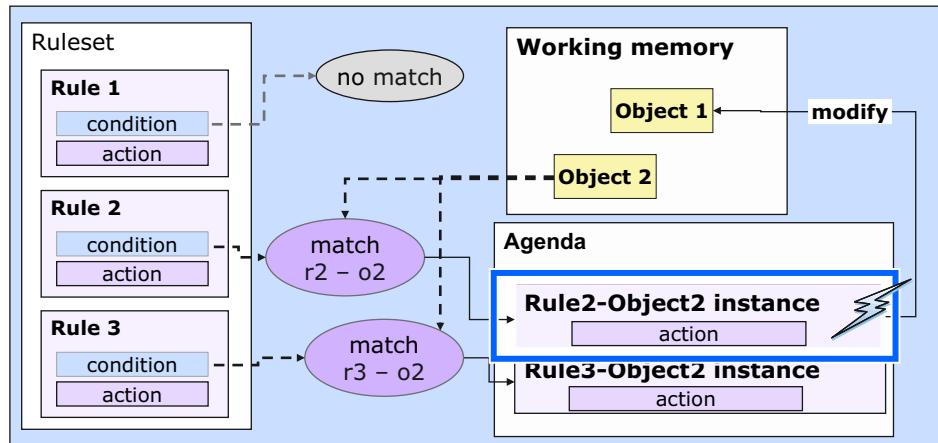


Note

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 2020

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

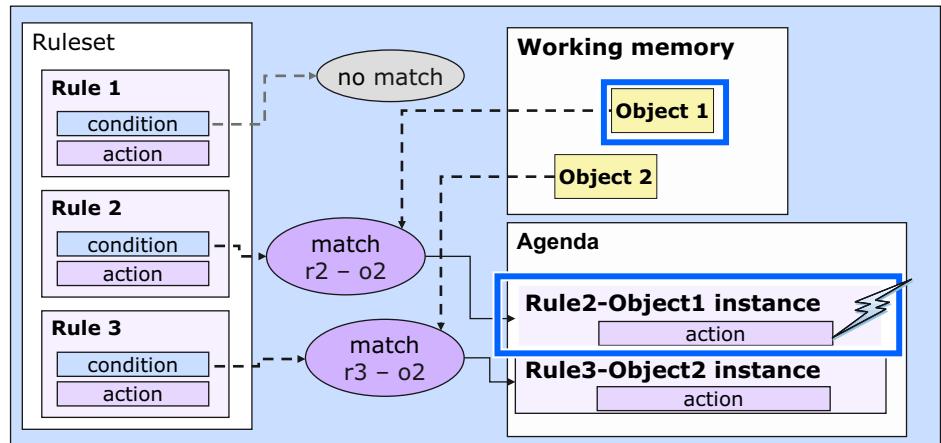


Figure 4-24. Recency (2 of 2)

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

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

Programming with business rules

© Copyright IBM Corporation 2020

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

Programming with business rules

© Copyright IBM Corporation 2020

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

4.4. RetePlus example: Trucks and drivers



RetePlus example: Trucks and drivers

Figure 4-27. RetePlus example: Trucks and drivers

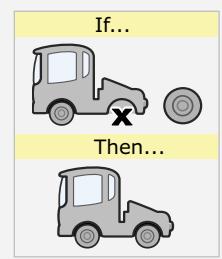
RetePlus in action

The ruleset contains these rules:

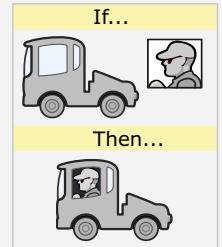
ChangeTire

AssignDriver

```
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;
```



Programming with business rules

© Copyright IBM Corporation 2020

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.

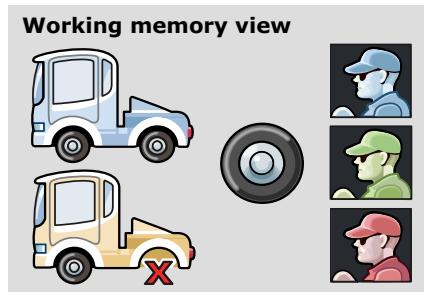
Every time an object is modified, its new state might result in a new match to a rule. The rule engine must be notified when objects are modified so that it can repeat the pattern matching process. To ensure this notification, you must select the appropriate **Update object state** options in the `Truck`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



Programming with business rules

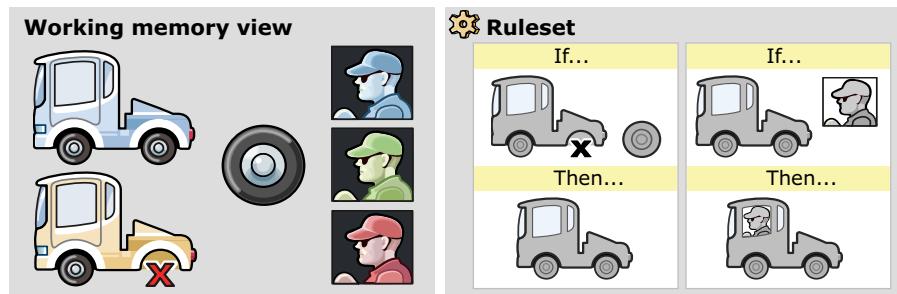
© Copyright IBM Corporation 2020

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



Programming with business rules

© Copyright IBM Corporation 2020

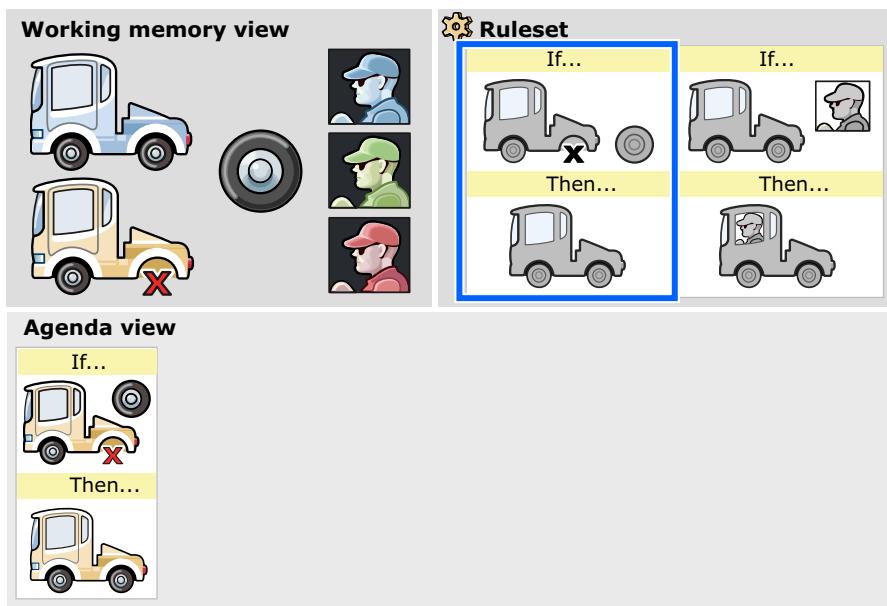
Figure 4-30. Evaluating rules with objects

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)

The **ChangeTire** rule conditions match a truck and an available tire in the working memory

Because conditions test true, a **ChangeTire** rule instance is put into the agenda



Programming with business rules

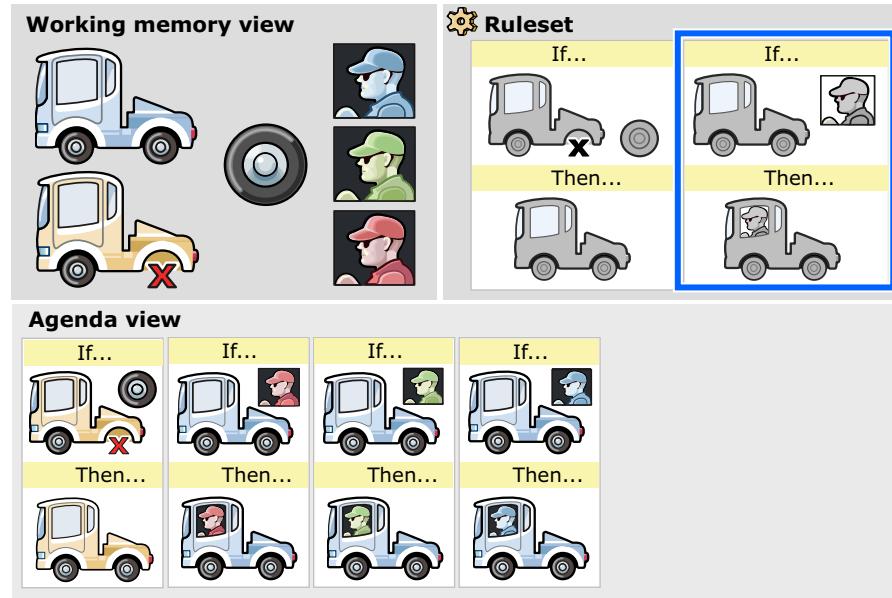
© Copyright IBM Corporation 2020

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)

AssignDriver rule
tests true 3 more times so 3 more instances added to agenda



Programming with business rules

© Copyright IBM Corporation 2020

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

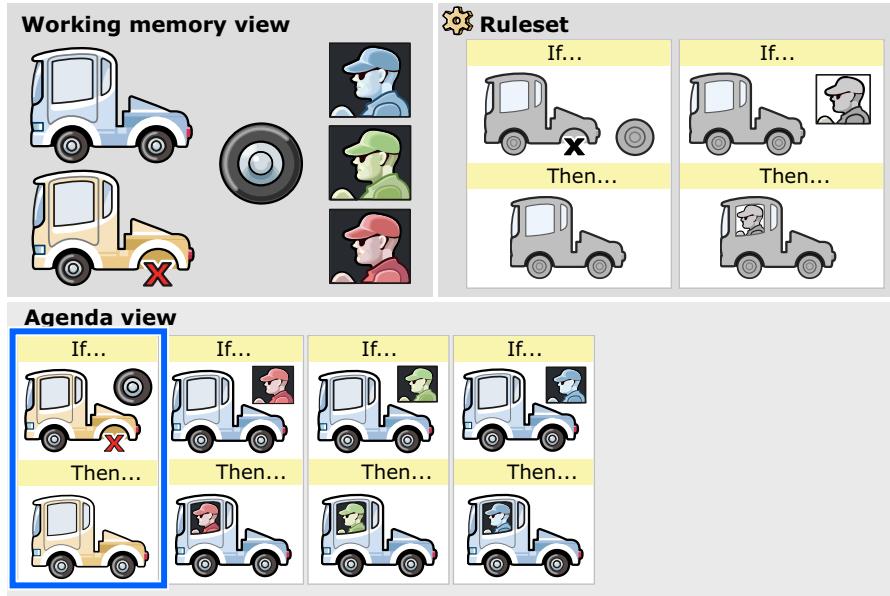
The `AssignDriver` rule evaluates to `true` three separate times:

- 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

The **ChangeTire** rule has the higher priority, so the **ChangeTire** rule instance executes first



Programming with business rules

© Copyright IBM Corporation 2020

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

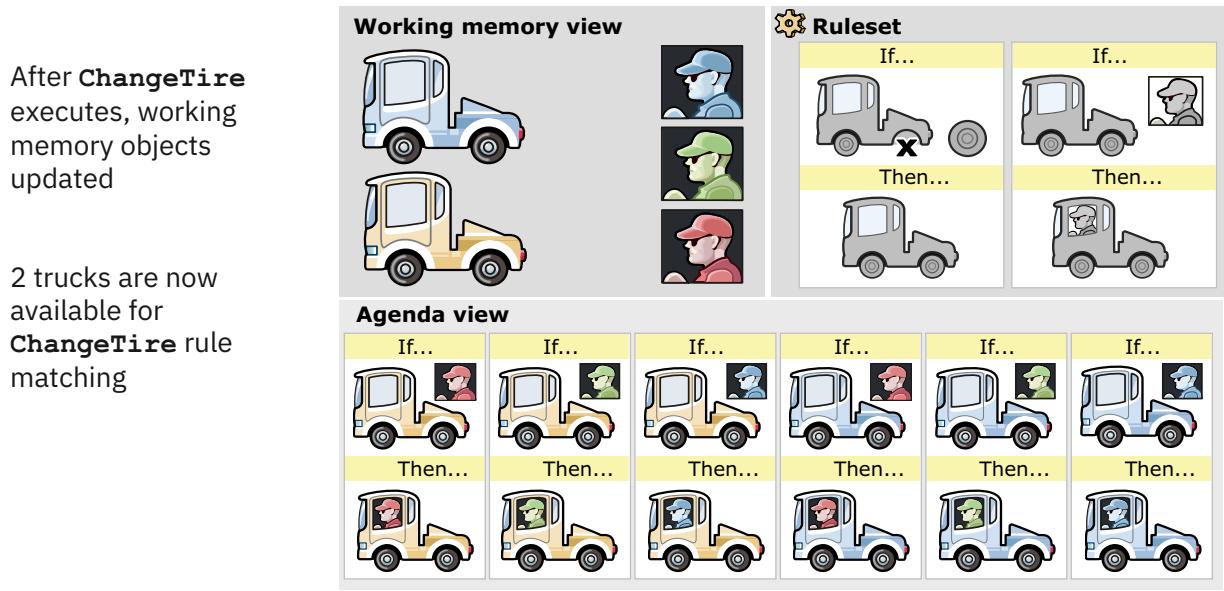


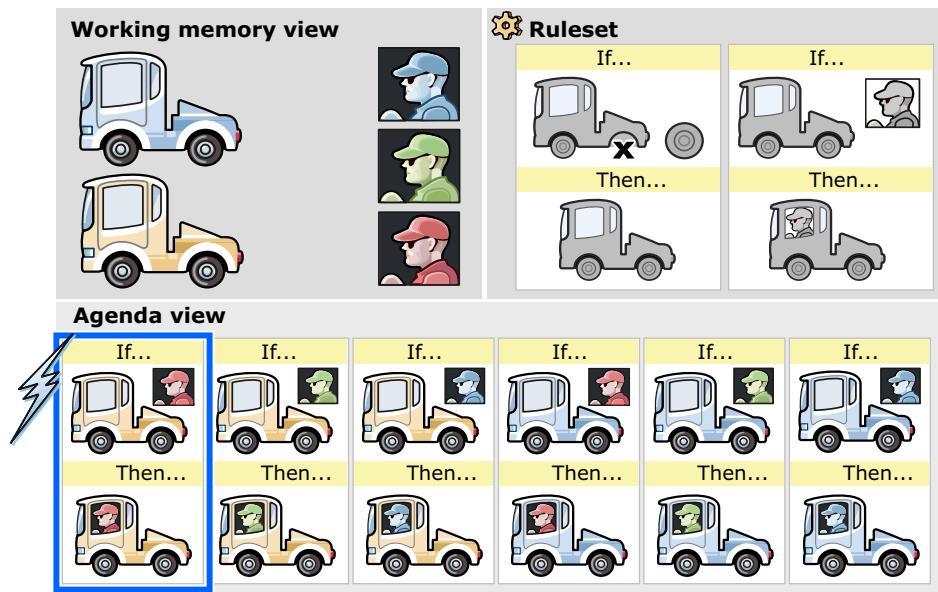
Figure 4-34. Side effects: Updated objects create matches

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

Last match fires first



Programming with business rules

© Copyright IBM Corporation 2020

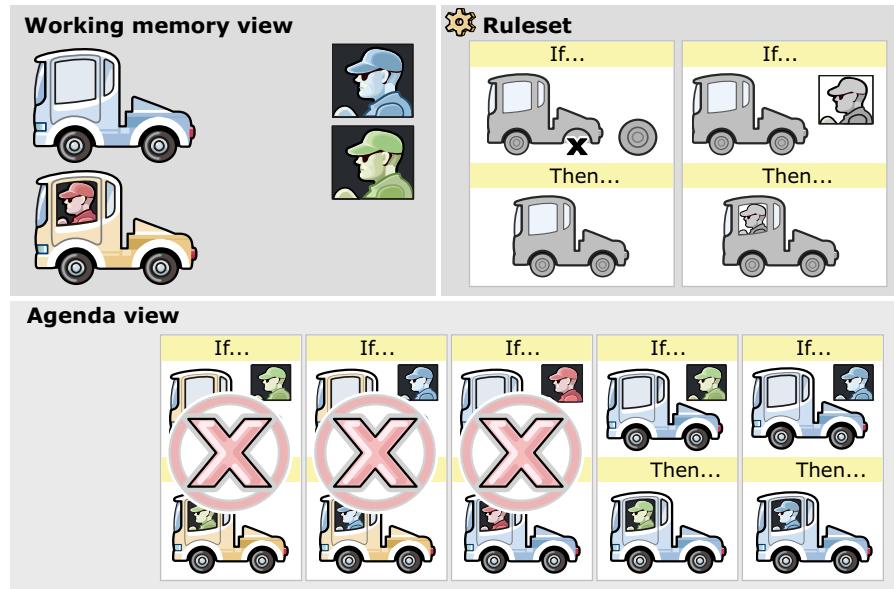
Figure 4-35. `AssignDriver` executes

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

AssignDriver no longer tests true with remaining objects in working memory

Rule instances removed from agenda



Programming with business rules

© Copyright IBM Corporation 2020

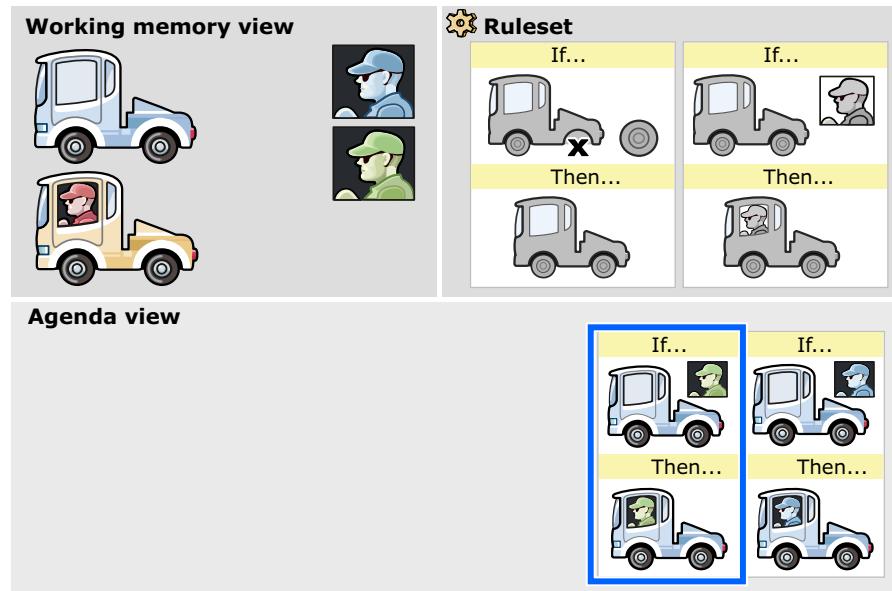
Figure 4-36. Side effects: Updated objects no longer match

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

The next most recent rule instance fires:

AssignDriver with the **blue truck** and the driver with the **green hat**



Programming with business rules

© Copyright IBM Corporation 2020

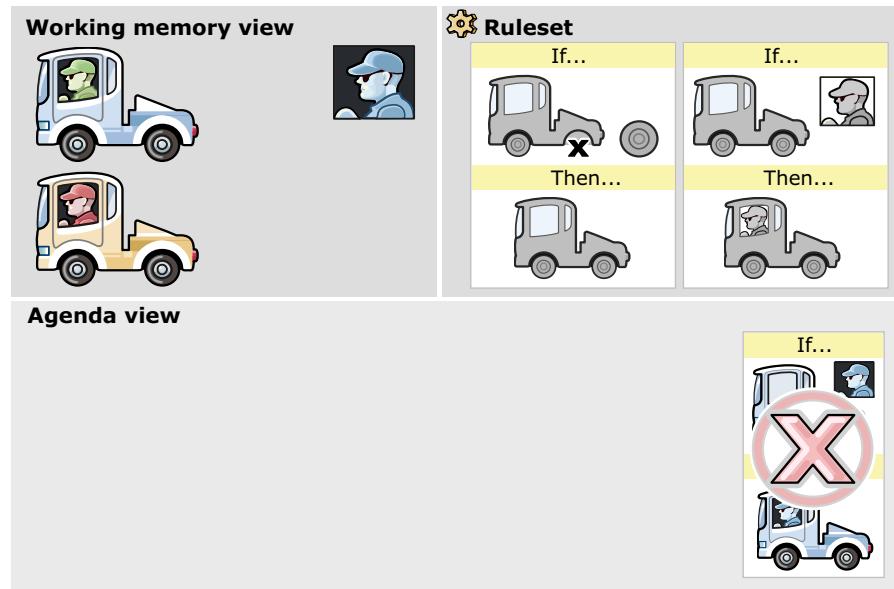
Figure 4-37. AssignDriver executes again

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

The working memory is updated and the rules retested but no matches are found

Rule instance is deleted from agenda



Programming with business rules

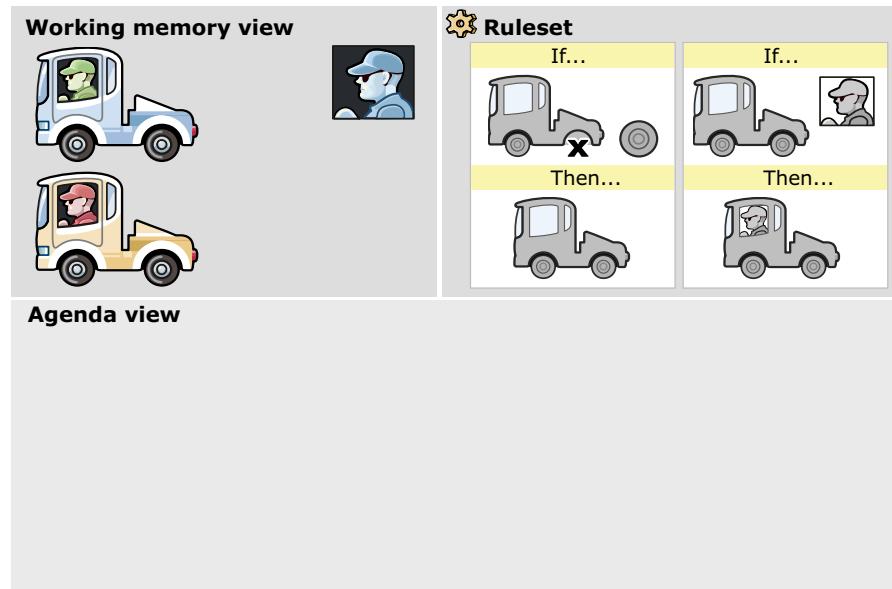
© Copyright IBM Corporation 2020

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

The agenda is empty and rule execution is complete



Programming with business rules

© Copyright IBM Corporation 2020

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 rule execution
- Explain rule execution modes and execution principles

© Copyright IBM Corporation 2020

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.

Programming with business rules

© Copyright IBM Corporation 2020

Figure 4-41. Review questions

Write your answers here:

- 1.
- 2.
- 3.

Review answers



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

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

© Copyright IBM Corporation 2020

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

© Copyright IBM Corporation 2020

Figure 5-2. Topics

5.1. Designing the models

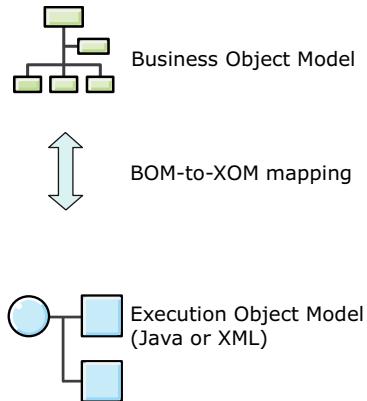


Designing the models

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



Developing object models

© Copyright IBM Corporation 2020

Figure 5-4. Introduction to the models

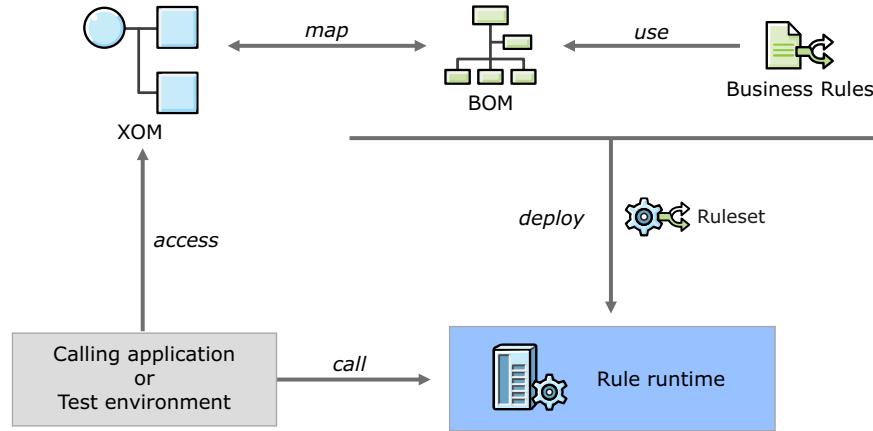
The business object model (BOM) and the execution object model (XOM) are related models.

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 2020

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

Developing object models

© Copyright IBM Corporation 2020

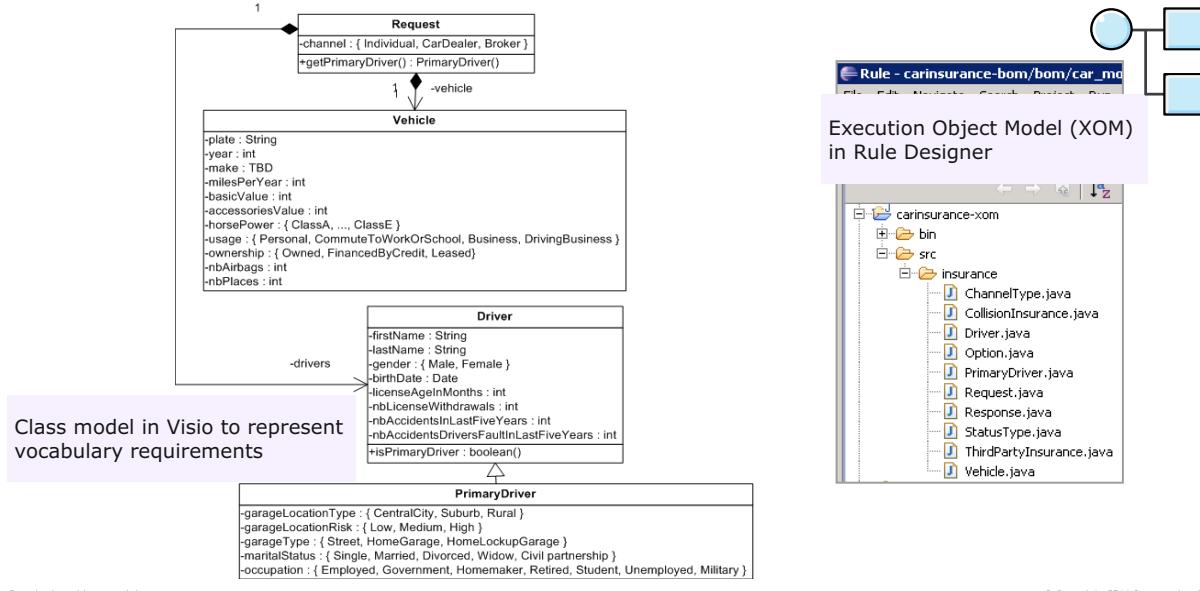
Figure 5-6. Designing the BOM and the XOM

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

The starting point is based on the models that the business analysts provide as requirements. For the bottom-up approach, developers use those requirements to implement the code XOM first and then generate the BOM automatically.

The BOM and the XOM evolve in parallel. Changes to the vocabulary require updates to the BOM and possibly the XOM, so expect several iterations before the BOM and XOM are stable.

Example: Car insurance class diagram



Developing object models

© Copyright IBM Corporation 2020

Figure 5-7. Example: Car insurance class diagram

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

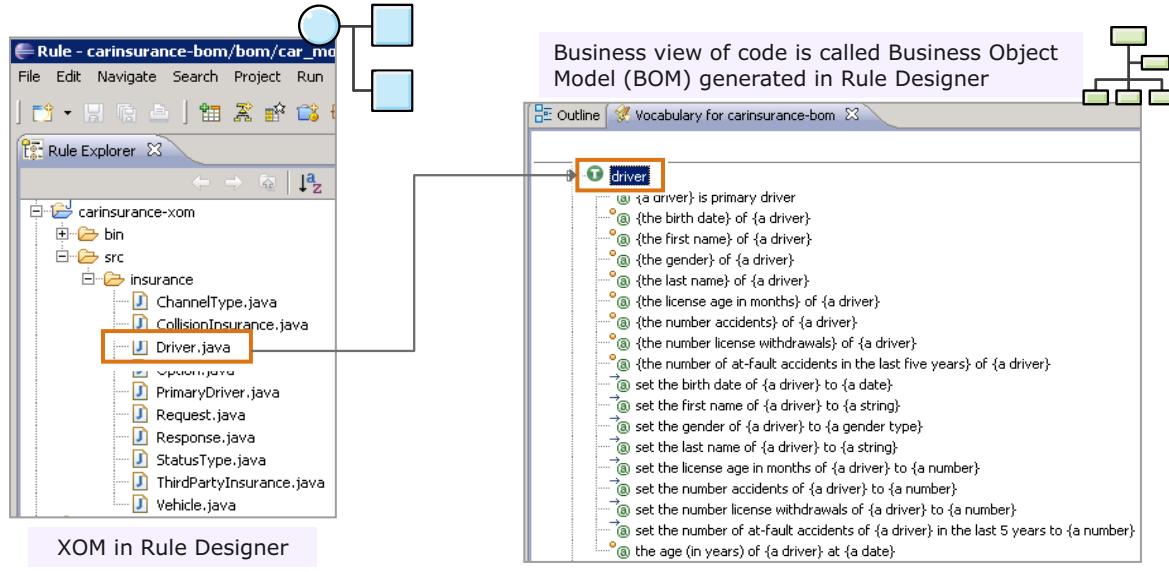
The developer can easily read these requirements and implement them in either Java or XML.

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.

Example: Implementation of insurance model



Developing object models

© Copyright IBM Corporation 2020

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.

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 2020

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

In this topic, you learn what the business execution model (BOM) is. You also learn what the execution object model (XOM) is. You also learn how to design the BOM and the XOM, and how these two models relate one to the other.



Defining business and execution object models

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*
 - Each entry defines a set of business elements
- BOM entries are ordered with a *BOM path*
- Attach natural-language terms to BOM elements to create *vocabulary*

Developing object models

© Copyright IBM Corporation 2020

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.

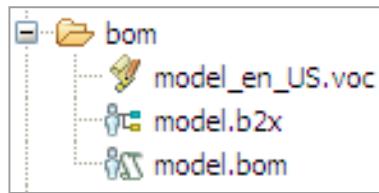
A single BOM can include one or several BOM *entries*, and each entry defines a set of business elements.

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
- Created manually or generated automatically from the XOM
 - A BOM entry is attached to a single XOM source to simplify its refactoring on XOM changes
- Composed of several files, with extensions `bom`, `voc`, and `b2x`



Developing object models

© Copyright IBM Corporation 2020

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

Developing object models

© Copyright IBM Corporation 2020

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

Developing object models

© Copyright IBM Corporation 2020

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

Developing object models

© Copyright IBM Corporation 2020

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

Developing object models

© Copyright IBM Corporation 2020

Figure 5-16. Rule project and XOM

5.3. Editing the business object model

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



Editing the business object model

Figure 5-17. Editing the business object model

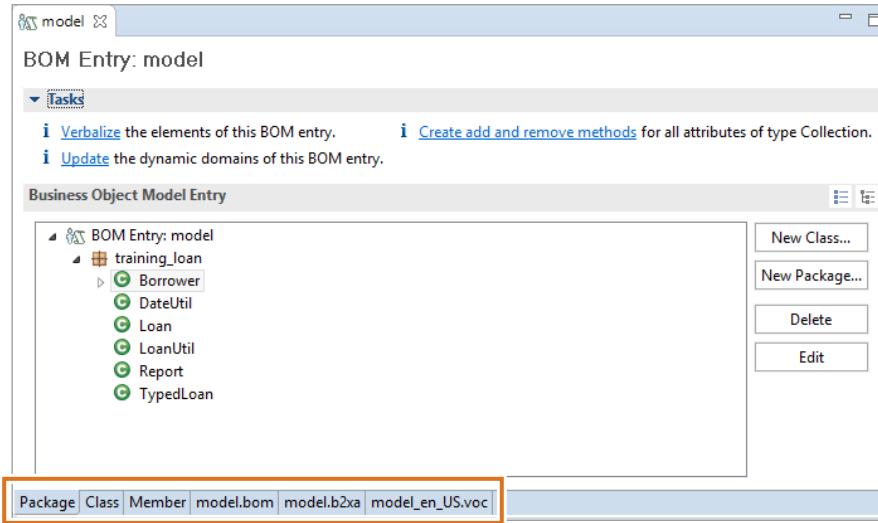
Introduction

- 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 that the tests and simulations in Decision Validation Services require

Figure 5-18. Introduction

BOM editor (1 of 2)

Edit BOM entries and the vocabulary with the BOM editor



Developing object models

© Copyright IBM Corporation 2020

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.

BOM editor (2 of 2)

Define BOM properties

Verbalize classes

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

Define constructors for testing and simulation

Use to generate scenario templates

The screenshot shows the 'Member Borrower (class: training_loan.Borrower)' dialog box. In the 'General Information' section, there are fields for Name (Borrower), Type (with a 'Browse...' button), Class (training_loan.Borrower), and two checkboxes: 'Deprecated' and 'Testing and simulation constructor'. The 'Testing and simulation constructor' checkbox is highlighted with a red border. In the 'Arguments' section, there is a table with four rows of arguments: arg1 (String), arg2 (String), arg3 (Date), and arg4 (String). To the right of the table are buttons for 'Add...', 'Remove...', 'Up', 'Down', and 'Edit...'. Below the table, it says 'Edit the arguments of this member.'

Name	Type	Domain	
arg1	java.lang.String		Add...
arg2	java.lang.String		Remove...
arg3	java.util.Date		Up
arg4	java.lang.String		Down
			Edit...

Developing object models

© Copyright IBM Corporation 2020

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.

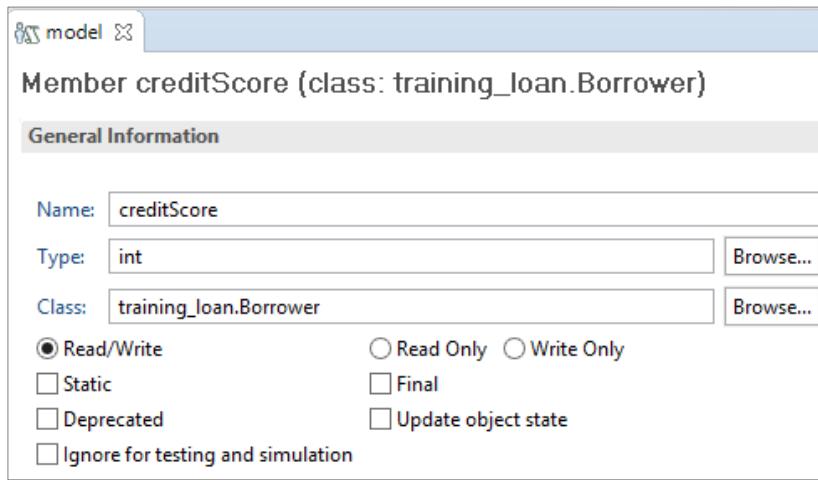
In the General Information section, you can edit the definition of the BOM element itself.

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

General information for attributes

Define object attributes



Developing object models

© Copyright IBM Corporation 2020

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 [<cross-ref>]Unit 10, "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.

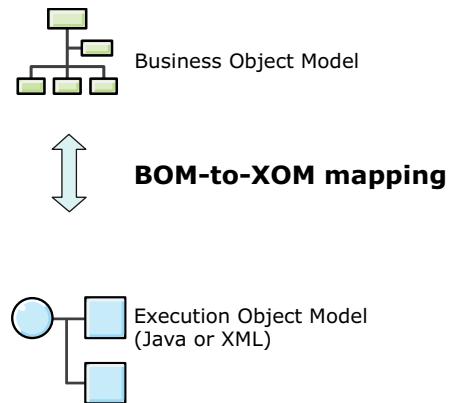


Mapping the BOM to the XOM

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



Developing object models

© Copyright IBM Corporation 2020

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

Developing object models

© Copyright IBM Corporation 2020

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.



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)

Developing object models

© Copyright IBM Corporation 2020

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

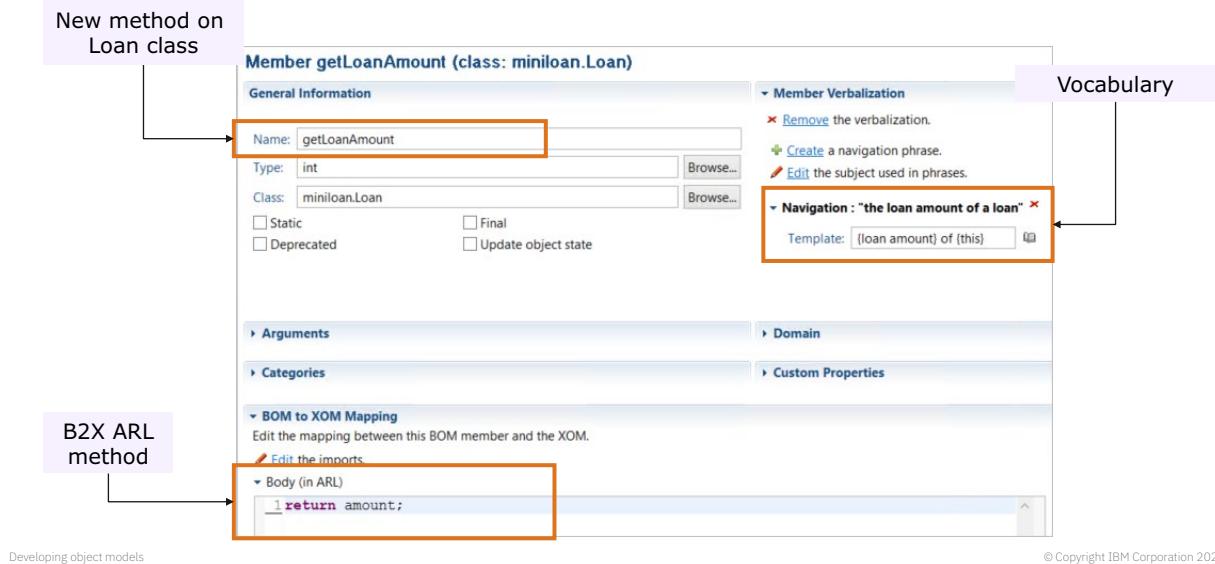


Figure 5-27. Create a method using ARL mapping code

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`

Developing object models

© Copyright IBM Corporation 2020

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

Developing object models

© Copyright IBM Corporation 2020

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

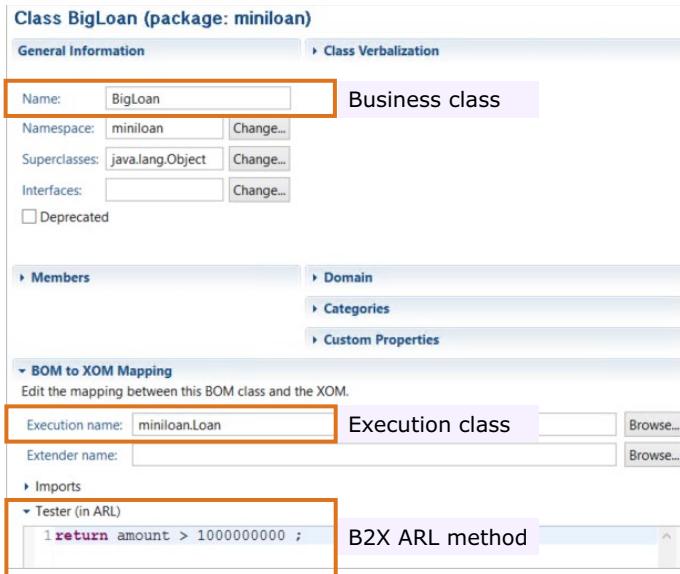
To illustrate how you map a business class to an execution class, let's look at the example here with the Miniloan decision service.

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.

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

Specify the execution class that you are mapping to

Define the B2X method in the **Tester** section



Developing object models

© Copyright IBM Corporation 2020

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

Developing object models

© Copyright IBM Corporation 2020

Figure 5-32. Explicit extender mapping

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.

5.5. Working with the vocabulary

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



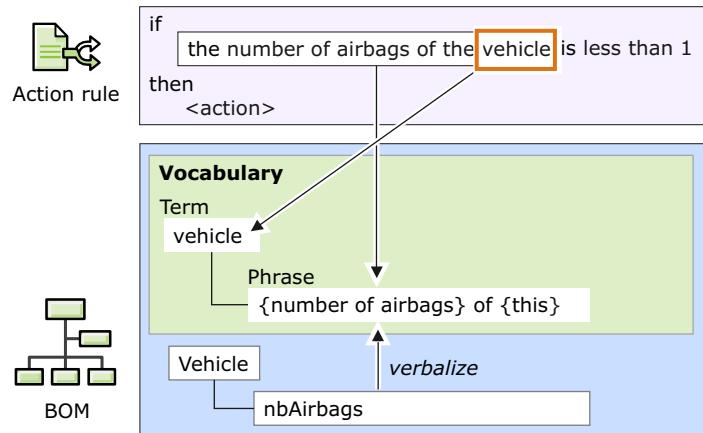
Working with the vocabulary

Figure 5-33. Working with the vocabulary

Rule vocabulary

Vocabulary in the rules comes directly from the BOM

Rule editors “see” set of terms and phrases that are attached to BOM members



Developing object models

© Copyright IBM Corporation 2020

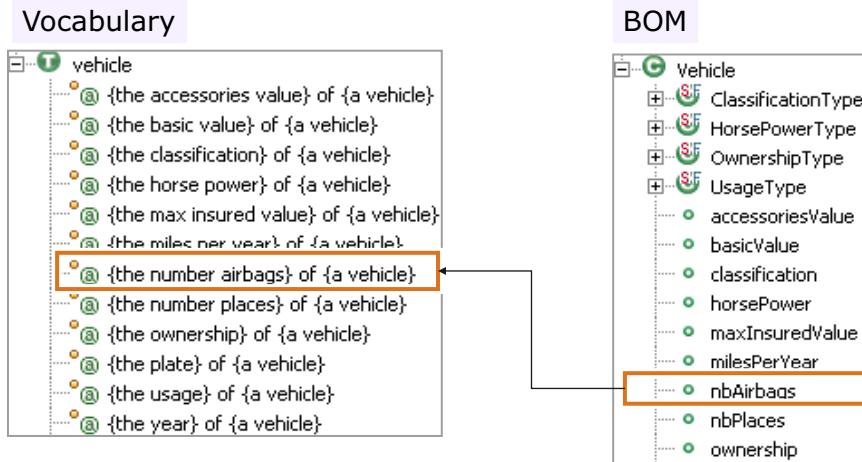
Figure 5-34. Rule vocabulary

The default name of a BOM class member might not be meaningful to business users when they are authoring rules, especially when they are not familiar with the syntax of programming languages.

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 2020

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

Can be automatically generated for selected BOM element

The screenshot shows a software interface for managing BOM elements. On the left, there is a sidebar with a tree view. On the right, a main panel displays the 'Member Verbalization' section for a selected BOM element. The section includes options to 'Remove' or 'Create' verbalizations and to 'Edit' subject phrases. Below this is a 'Navigation' section for the 'amount of a loan' navigation phrase, which includes a template input field containing '{amount} of {this}' and a small icon.

Developing object models

© Copyright IBM Corporation 2020

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 2020

Figure 5-37. Verbalization (1 of 2)



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, branchs`
- 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 2020

Figure 5-38. Verbalization (2 of 2)

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.

Placeholders

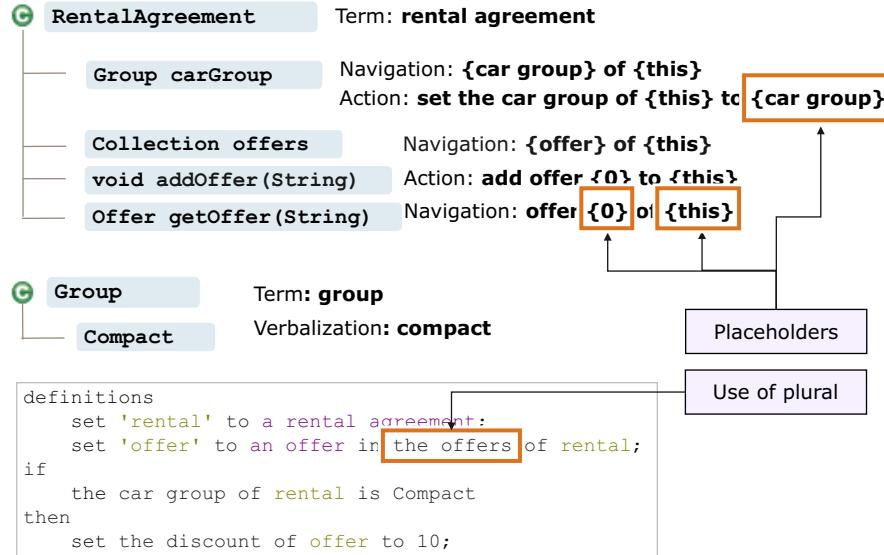
- Vocabulary phrase templates contain placeholders
- Placeholders represent gaps in phrases that can be completed automatically or manually when editing rules
- Braces identify the placeholders: { }

Developing object models

© Copyright IBM Corporation 2020

Figure 5-39. Placeholders

Verbalizing BOM members



Developing object models

© Copyright IBM Corporation 2020

Figure 5-40. Verbalizing BOM members

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} of {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

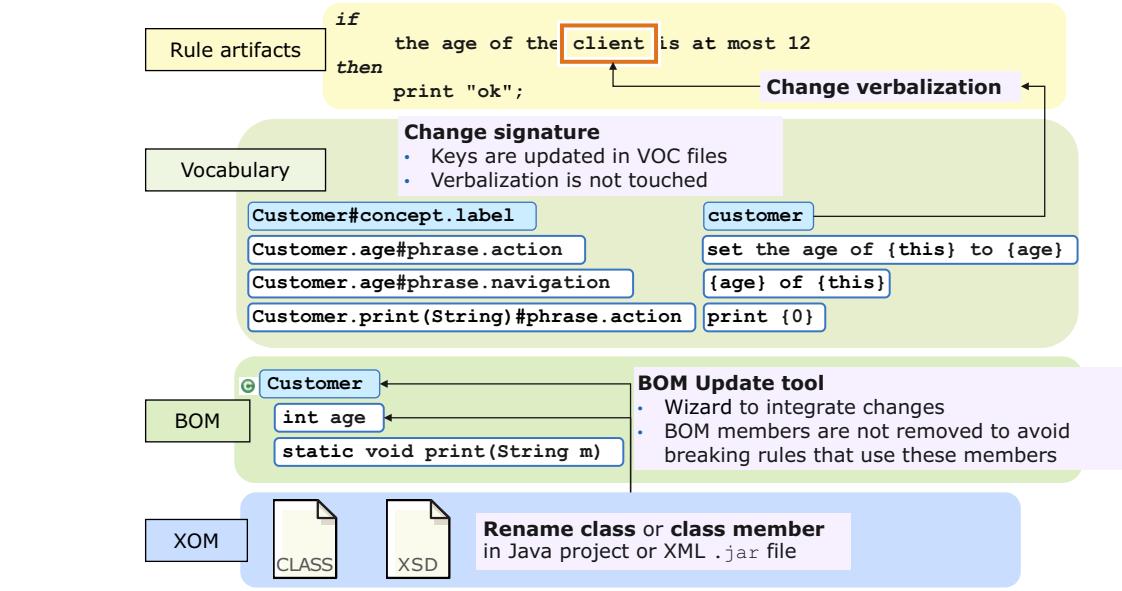
In this topic, you learn why and how you can keep the BOM and the XOM consistent with each other.



Refactoring

Figure 5-41. Refactoring

Refactoring



Developing object models

© Copyright IBM Corporation 2020

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 (*.voc)
 - BOM (*.bom)
 - BOM-to-XOM mapping (*.b2xa)
 - XOM (Java or dynamic)

Rule artifacts

Vocabulary (*.voc)

BOM (*.bom)

BOM-to-XOM mapping (*.b2xa)

XOM (Java or dynamic)

Developing object models

© Copyright IBM Corporation 2020

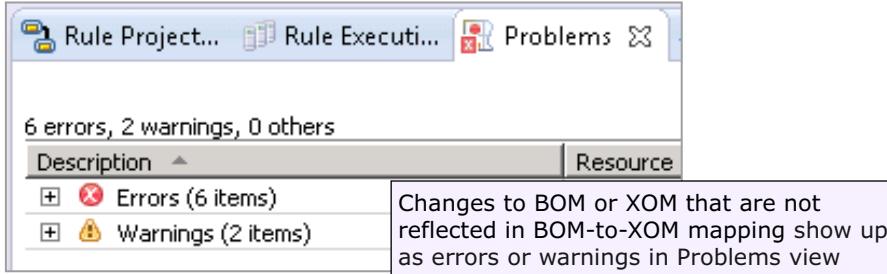
Figure 5-43. From the XOM to the rules

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



Developing object models

© Copyright IBM Corporation 2020

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

Developing object models

© Copyright IBM Corporation 2020

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.

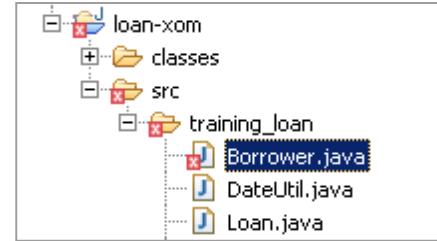
XOM changes (2 of 3)

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



Developing object models

© Copyright IBM Corporation 2020

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 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

Developing object models

© Copyright IBM Corporation 2020

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

Developing object models

© Copyright IBM Corporation 2020

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

Developing object models

© Copyright IBM Corporation 2020

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

© Copyright IBM Corporation 2020

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 2020

Figure 5-51. Review questions

Write your answers here:

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

Review answers



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.

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 5-52. Review answers

Exercise: Working with the BOM



Figure 5-53. Exercise: Working with the BOM

Exercise introduction

- Generate a BOM from an existing XOM
- Verbalize the BOM with natural-language vocabulary

© Copyright IBM Corporation 2020

Figure 5-54. Exercise introduction

Exercise: Refactoring



Figure 5-55. Exercise: Refactoring

Exercise introduction

- Refactor vocabulary changes
- Manage inconsistency issues after updating the XOM and BOM

© Copyright IBM Corporation 2020

Figure 5-56. 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

© Copyright IBM Corporation 2020

Figure 6-1. Unit objectives

Topics

- Controlling rule execution: Overview
- Designing ruleflows
- Controlling rule selection for execution
- Controlling rule order during rule execution

© Copyright IBM Corporation 2020

Figure 6-2. Topics

6.1. Controlling rule execution



Controlling rule execution

© Copyright IBM Corporation 2020

Figure 6-3. Controlling rule execution

What is orchestration?

- A ruleset produces a single business decision
 - Individual rules have no notion of sequence
- Ruleflow defines routing logic
 - Determines appropriate sequence of evaluation from a set of possible paths through the ruleset
 - Enforces the sequence in which rules are selected
 - Controls the data that flows in and out of the rule engine
 - Defines rule engine algorithm to evaluate rules



Orchestrating ruleset execution

© Copyright IBM Corporation 2020

Figure 6-4. What is orchestration?

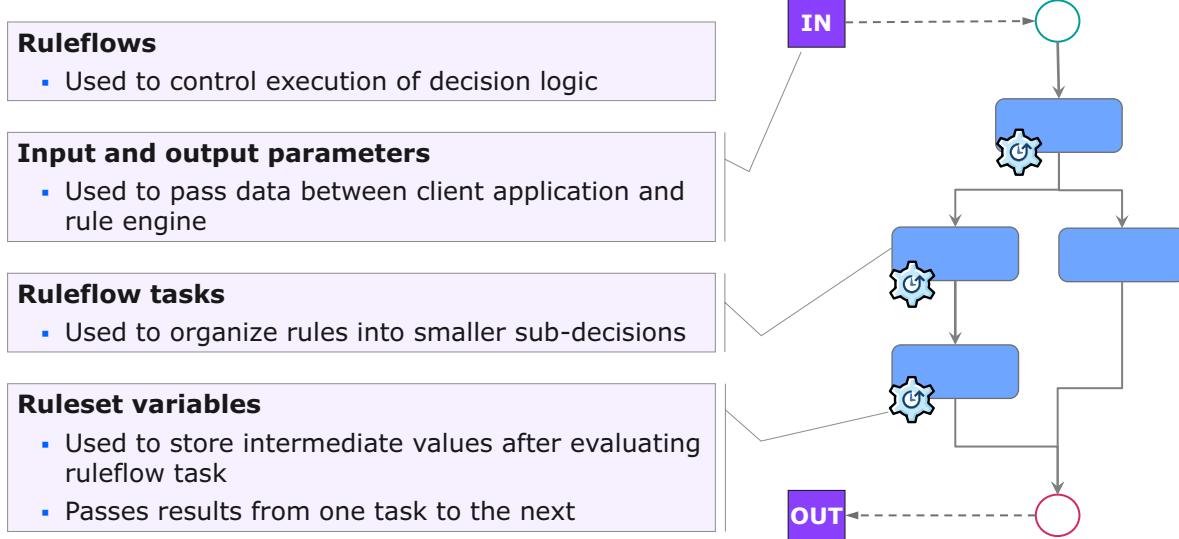
A ruleset is designed to produce a single business decision. But the individual rules in the ruleset have no notion of sequence. How do you ensure that the rule execution produces the correct result?

You use ruleflows to manage and orchestrate the sequence of execution. They provide the means to specify how, when, and under what conditions rules are executed.

Ruleflows control the data that flows in and out of the rule engine, and also defines which method the rule engine should use to evaluate the rules against the data.

The individual rules in the ruleset are evaluated as a series of smaller decisions that lead to the overall business decision. The routing logic determines the appropriate sequence of evaluation from a set of possible paths through the ruleset.

Key elements for orchestration



Orchestrating ruleset execution

© Copyright IBM Corporation 2020

Figure 6-5. Key elements for orchestration

To outline the ruleflow, you need to work with the business analysts and policy managers to ensure the flow of execution produces the correct result. Together you also identify the input and output: what information is required to make the decision, and what type of information should be included as the decision result. This information is passed between the client application and the rule engine as parameters.

When the input parameters are passed to the rule engine, the rule engine evaluates those objects against the rules in each task. After evaluating a task, the results are stored in a ruleset variable. Based on those results, the routing logic determines which task to evaluate next.

After the ruleflow completes, the result is returned to the client application.

6.2. Designing ruleflows



Designing ruleflows

© Copyright IBM Corporation 2020

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.

Ruleflows

- No procedural code required for routing
 - Sequence defined graphically
 - Multiple transitions from a task can use conditions to determine correct path
- A decision service can contain several ruleflows
 - Identify one ruleflow as the **main ruleflow** for the decision operation
 - Additional ruleflows can be used as **subflow tasks**

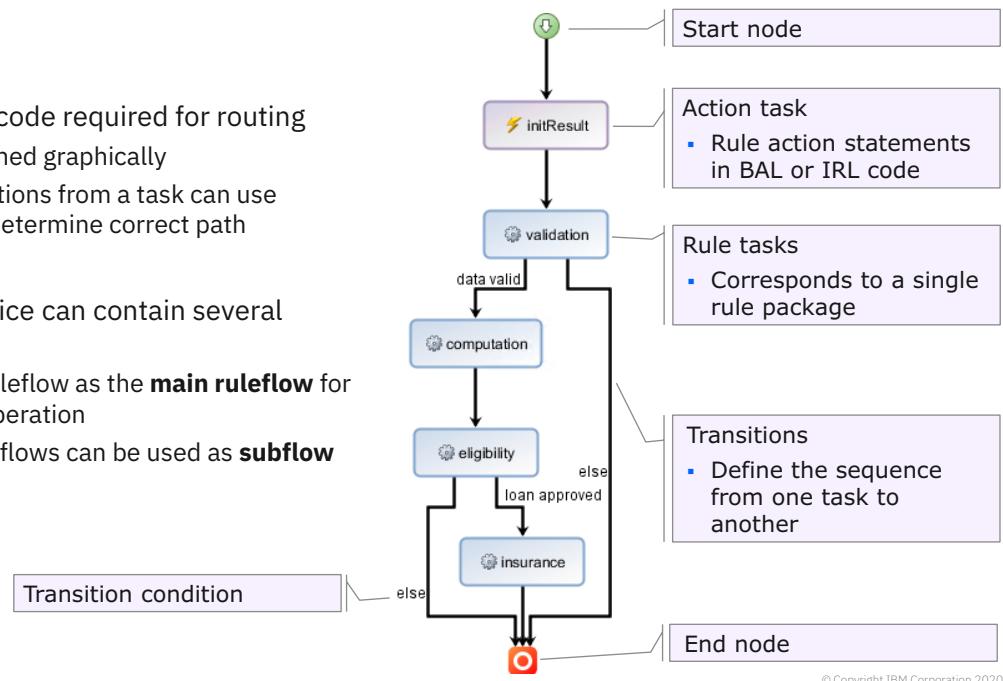


Figure 6-7. Ruleflows

With the ruleflow, you do not have to write procedural code to control the flow of execution of your business rules. You create ruleflows graphically, as a ruleflow diagram in Rule Designer.

A ruleflow has one start node and one or more end nodes. It is composed of tasks and transitions.

Rule tasks generally correspond to a single rule package. You specify the order in which these rule tasks are evaluated by chaining them together with transitions. Transitions determine how, when, and under what conditions to use each rule task. You use *transition conditions* when multiple paths from a task are possible. The conditions determine which transition path to follow.

Within the ruleflow, you can also define properties to control how the rules are selected and evaluated at run time. For each rule task, you can specify rule order, rule selection, and execution modes

A decision service can contain several ruleflows. However, your ruleset must identify one ruleflow as the *main ruleflow* for the project. Additional ruleflows can be accessed as subflow 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 a rule package are included the next time that the ruleset is deployed

Figure 6-8. 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
 - When multiple end nodes are present, 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

Orchestrating ruleset execution

© Copyright IBM Corporation 2020

Figure 6-9. 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 optional and can be used independently of each other.

Transitions

- Transitions
 - Unidirectional arrows that connect tasks and determine the route through the ruleflow
- Transition conditions
 - Boolean condition to determine which transition path to follow
- Can include an “**else**” condition
 - For all cases that do not match other conditions in the transition
 - Maximum one “**else**” per transition

Orchestrating ruleset execution

© Copyright IBM Corporation 2020

Figure 6-10. 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

Orchestrating ruleset execution

© Copyright IBM Corporation 2020

Figure 6-11. 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 6-12. 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`

Orchestrating ruleset execution

© Copyright IBM Corporation 2020

Figure 6-13. 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

© Copyright IBM Corporation 2020

Figure 6-14. Controlling rule selection for execution

Rule selection pipe

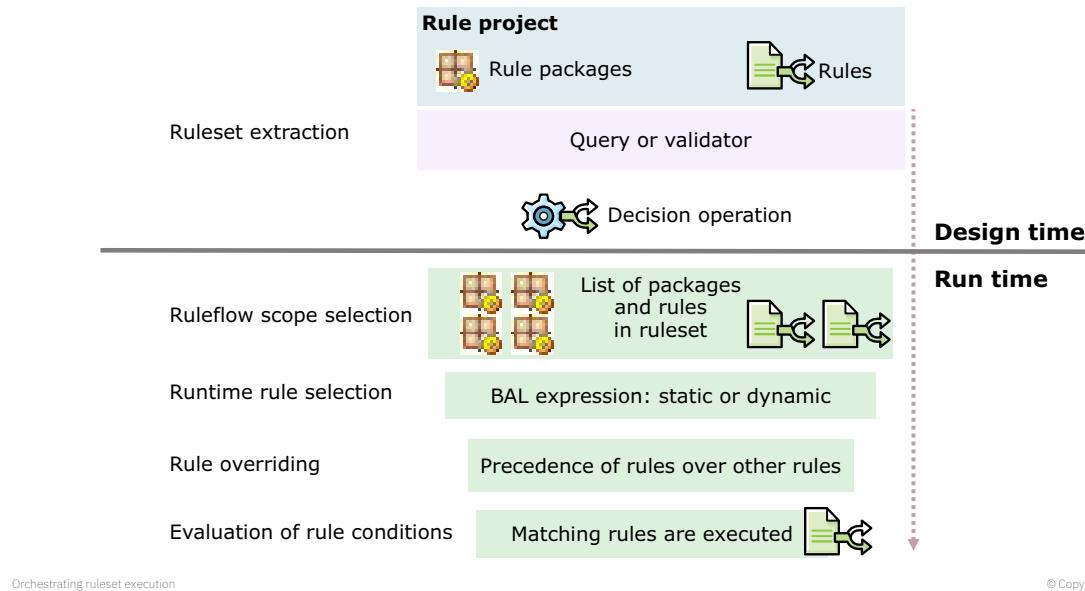


Figure 6-15. Rule selection pipe

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

Orchestrating ruleset execution

© Copyright IBM Corporation 2020

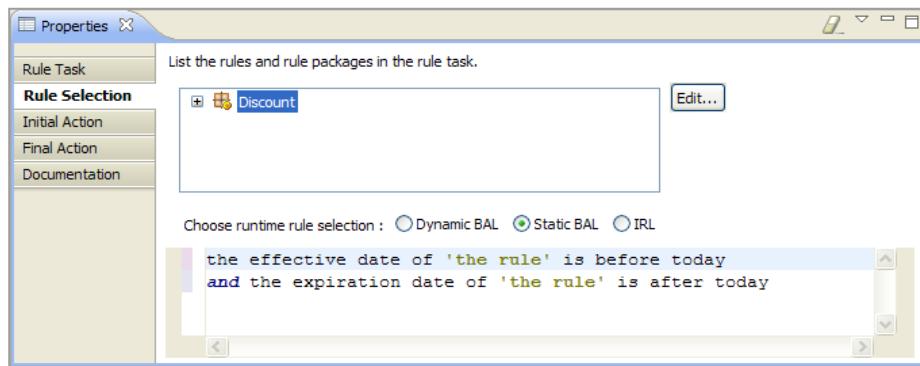
Figure 6-16. Ruleflow scope selection

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 2020

Figure 6-17. 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

Orchestrating ruleset execution

© Copyright IBM Corporation 2020

Figure 6-18. 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

Orchestrating ruleset execution

© Copyright IBM Corporation 2020

Figure 6-19. 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

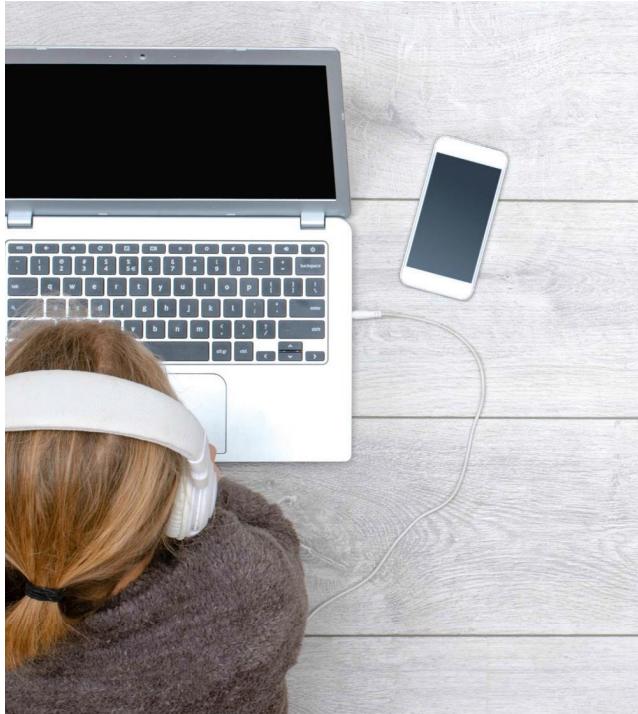
Orchestrating ruleset execution

© Copyright IBM Corporation 2020

Figure 6-20. 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

© Copyright IBM Corporation 2020

Figure 6-21. Controlling rule order during rule execution

Controlling rule order

- When rules are selected to be evaluated, you can control how they are executed by setting these properties:
 - Rule priority
 - Rule task execution order
 - Execution modes

Orchestrating ruleset execution

© Copyright IBM Corporation 2020

Figure 6-22. Controlling rule order

When rules are selected to be evaluated, you can control how they are executed by setting these properties:

- Rule priority
- Rule task execution order
- Execution modes

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

Orchestrating ruleset execution

© Copyright IBM Corporation 2020

Figure 6-23. 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 -10^9 and 10^9 . 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 6-24. Rule task execution order (1 of 2)

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

Orchestrating ruleset execution

© Copyright IBM Corporation 2020

Figure 6-25. 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 6-26. Execution modes

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

© Copyright IBM Corporation 2020

Figure 6-27. 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

Orchestrating ruleset execution

© Copyright IBM Corporation 2020

Figure 6-28. 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. Fastpath[The answer is C.](#)

Figure 6-29. Review answers

Exercise: Working with ruleflows



Figure 6-30. Exercise: Working with ruleflows

Exercise introduction

- Describe the parts of a ruleflow
- Create a ruleflow
- Orchestrate rule selection and execution through the ruleflow

© Copyright IBM Corporation 2020

Figure 6-31. Exercise introduction

Unit 7. Authoring rules

Estimated time

01:30

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

© Copyright IBM Corporation 2020

Figure 7-1. Unit objectives

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

© Copyright IBM Corporation 2020

Figure 7-2. Topics

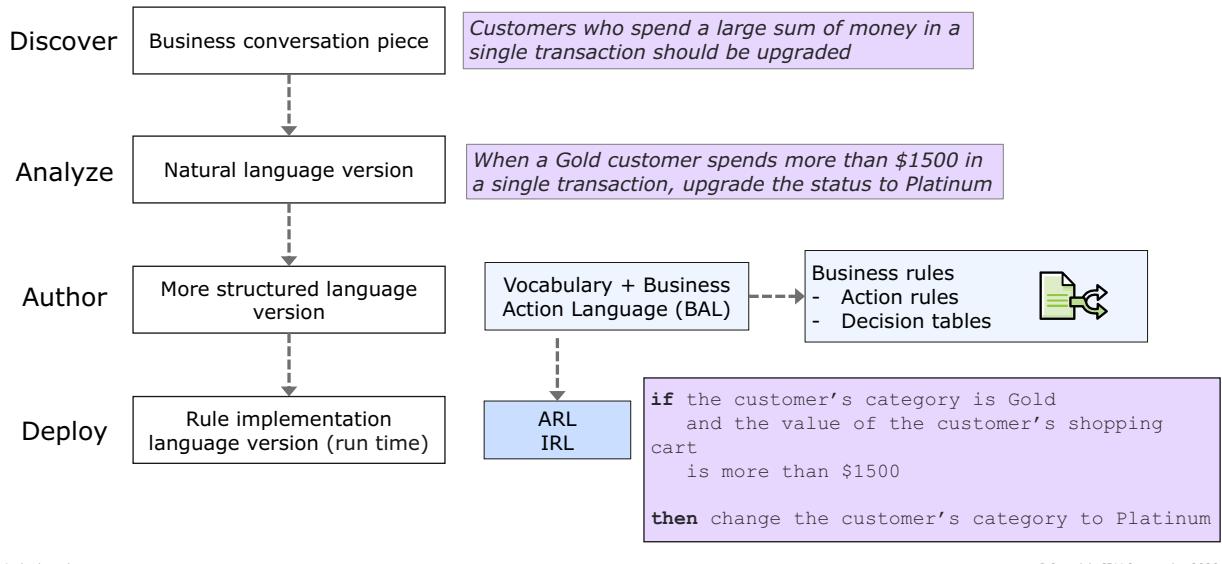
7.1. From business policy to business rules



From business policy to business rules

Figure 7-3. From business policy to business rules

Rule language evolves during a project



Authoring rules

© Copyright IBM Corporation 2020

Figure 7-4. Rule language evolves during a project

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 discussion.
- **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 Advanced Rule Language (ARL) or ARL. Some technical rules are written in ILOG rule language, or IRL.

Operational Decision Manager roles and activities

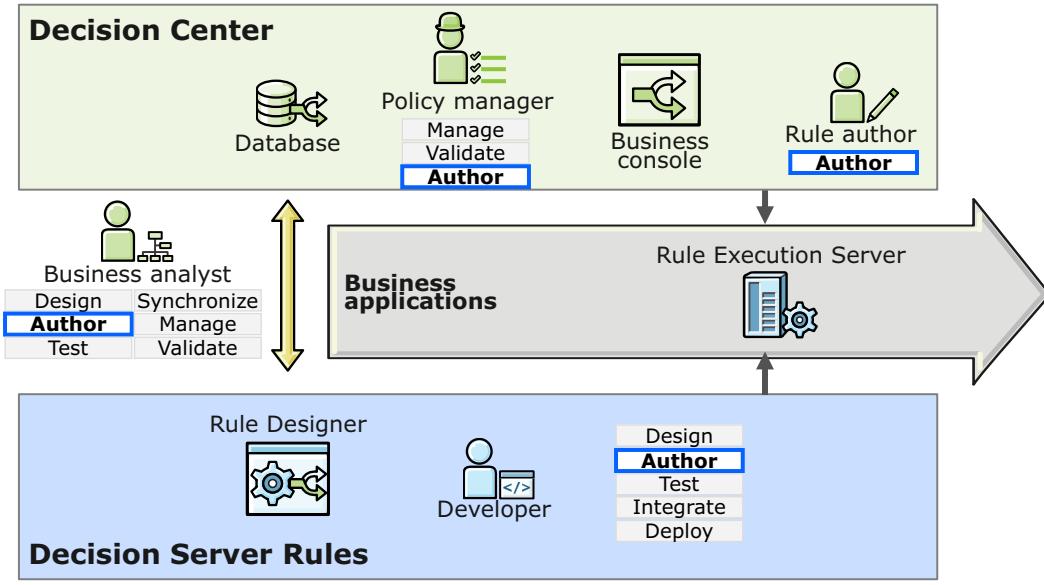


Figure 7-5. Operational Decision Manager roles and activities

During the early stages of a project, a large-scale effort is made to enter the business rules that are identified during discovery and analysis phases. After the business rule application is rolled out to production, rule maintenance becomes a regular task for policy managers and rule authors as business policies shift and more rules are identified. Business console is their main rule authoring and management tool.

While the business team usually does the rule authoring, developers also get involved when more complex or **technical** rules are required. The rule editors in Rule Designer provide support creation of technical rules and functions. You learn about technical rules in this unit.

7.2. Authoring action rules with BAL



Authoring action rules with BAL

Figure 7-6. 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 2020

Figure 7-7. Business Action Language (BAL)

Action rules are written with the Business Action Language (BAL), which provides a simple syntax to express rules.

The BAL provides constructs for defining variables on which your rules work, and for expressing conditions and actions on your business data.

The 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 Business console.

Rule structure

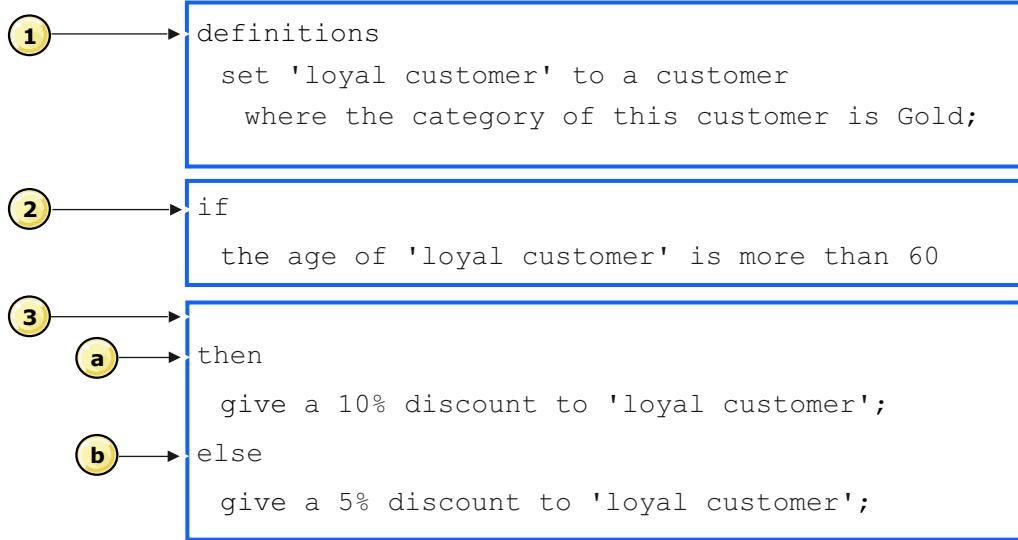
Definitions (set)	<ul style="list-style-type: none">Used to declare rule variables that are used in the rule and tests on these variables, if necessaryDefinitions are optional
Conditions (if)	<ul style="list-style-type: none">Used to define the conditions that determine when a rule is executedRules with no conditions are executed under all circumstancesConditions are optional
Actions (then)	<ul style="list-style-type: none">Used to define the actions that are done when all conditions are metAt least one action is mandatory
Actions (else)	<ul style="list-style-type: none">Statements after the keyword else define the actions when conditions are not metKeyword else is optional, and valid only when the keyword if is present

Authoring rules

© Copyright IBM Corporation 2020

Figure 7-8. Rule structure

Example of action rule



Authoring rules

© Copyright IBM Corporation 2020

Figure 7-9. Example of action rule

In the example on this slide:

1. The part that starts with the keyword `definitions` gives the definitions of the action rule.
 - In the example, the definitions declare a `loyal customer` rule variable that is defined with the pattern “`a customer where the category of this customer is Gold`” to manipulate objects in the working memory.
 - At run time, the rule engine does pattern matching on all the objects in the working memory of the `Customer` class. Each object in the working memory of the `Customer` class that matches the pattern that is indicated in the definitions is a candidate. If the conditions also apply to it, the rule engine creates a rule instance for this object in the working memory to fire the actions between `then` and `else`. Otherwise, because the rule includes actions after `else`, a rule instance is created for this object to fire the actions after `else`.
2. The part that starts with the keyword `if` gives the conditions of the action rule.
3. The part that starts with the keyword `then` defines the actions to take when the conditions are met.
 - 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 7-10. Rule definitions: Overview

In the `definitions` part of the rule, you define rule variables with the keyword `set` and the rule variable name. A rule variable is local to your rule.

You then assign a value to a rule variable by introducing it with the keyword `to`.

You can further add constraints that are introduced with the keywords `in`, `from`, or `where`.

The following slides give more details on these keywords.

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 7-11. Rule definitions: Values

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 7-12. Rule definitions: Constraints

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 7-13. Multiple variables in BAL



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;
```

Authoring rules

© Copyright IBM Corporation 2020

Figure 7-14. Rule conditions: Introduction

The conditions can include one or more logical expressions that produce a Boolean value.

Examples:

- `true`
- `false`
- '`the borrower`' has had a bankruptcy
- '`the report`' is approved

Use the logical conditions to test for:

- A comparison
- An existence
- A count
- A membership evaluation

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 7-15. Rule conditions: Comparison test

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 7-16. Rule conditions: Membership test

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 7-17. Rule conditions: Existence test

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 7-18. Rule conditions: Count test

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

Figure 7-19. Multiple conditions

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)

Authoring rules

© Copyright IBM Corporation 2020

Figure 7-20. Avoiding ambiguity with parentheses (1 of 2)

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 7-21. Avoiding ambiguity with parentheses (2 of 2)

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 7-22. Use of commas (,) in rule conditions

Use commas in the rule conditions to avoid ambiguities in successive conditions. For example, 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 2020

Figure 7-23. Rule actions

Rule actions provide the executable statements of a rule artifact, and are made of one of both of the following blocks:

- 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 actions in the `then` block execute when the definitions, if any, are satisfied and the conditions (`if`) are satisfied.
- The `else` block of action statements is optional and contains all the action statements that are found after the `else` keyword. The actions in the `else` block execute when the conditions are not met.

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`) are not satisfied.

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;
```

Authoring rules

© Copyright IBM Corporation 2020

Figure 7-24. Examples of rule actions

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>] ,]

Authoring rules

© Copyright IBM Corporation 2020

Figure 7-25. BAL number operators

The BAL operators, including and, or, it is not true that, equals, does not equal, is after, and is before, are in fact vocabulary that is associated with a *boot BOM*. The boot BOM is verbalized with the **System** vocabulary, which is visible in the Vocabulary view of your project.

You must work with the boot BOM to change its verbalization. For example, you might need to change the verbalization of the boot BOM 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 Example: set 'full name' to 'first name' + ' ' + 'last name'

Figure 7-26. BAL arithmetic operators

7.3. Authoring decision tables



Authoring decision tables

Figure 7-27. Authoring decision tables

About decision tables

- Convenient way to view and manage large sets of similar business rules
 - Common condition tests and similar actions
- Composed of rows and columns:
 - Columns define the conditions and actions
 - Each row is an if-then statement
- The decision engine considers entire table as a single rule
 - Only one row can execute
- 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
 - Consistency checking features (static rule analysis)

Authoring rules

© Copyright IBM Corporation 2020

Figure 7-28. About decision tables

Decision tables provide a concise way of viewing a set of business rules in the form of a spreadsheet. Use decision tables when your rules share common condition terms and when they have similar actions.

The columns are used to define the conditions and actions. Each row is an if-then statement. The decision engine considers the entire table as one rule.

Some benefits of creating decision tables are that preconditions can be applied to all rules are that you can easily see overlaps and gaps between rules, and decision tables have built-in consistency checking features.

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

Authoring rules

© Copyright IBM Corporation 2020

Figure 7-29. Example: Symmetrical rules

Patterns in rules are a good indicator that a decision table might be the best way to implement those rules.

Each of the rules on this slide can become a row in a decision table.

Notice that 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

Figure 7-30. Example: Symmetrical rules in a decision table

Structure of a decision table

- Columns are conditions or actions
- Each row is an if-then statement
 - When conditions for a row are met, actions in that row are executed

The diagram illustrates the structure of a decision table. It features a header row with three columns: 'Row header' (containing row numbers 1 through 9), 'Condition column' (containing 'Yearly income' with sub-columns 'Min' and 'Max'), and 'Action column' (containing 'Add to corporate score'). A bracket labeled 'Column header' spans the first two columns. Blue arrows point from the labels to their respective parts of the table.

Row header	Condition column		Action column
	Min	Max	
1	< 10,000		21
2	10,000	20,000	50
3	20,000	30,000	80
4	30,000	50,000	120
5	50,000	80,000	150
6	80,000	120,000	200
7	120,000	200,000	250
8	$\geq 200,000$		300
9			

Authoring rules

© Copyright IBM Corporation 2020

Figure 7-31. 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 label column headers with 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.

Notice that the action column has a shaded background, which helps to visually distinguish it from condition columns.

Only one row of a decision table can execute. When condition columns for one of the rows are met, the action columns in that row are executed.

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

Authoring rules

© Copyright IBM Corporation 2020

Figure 7-32. 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 that can be used in all the rules of the decision table.

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

Authoring rules

© Copyright IBM Corporation 2020

Figure 7-33. 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.

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

Authoring rules

© Copyright IBM Corporation 2020

Figure 7-34. Locking facilities

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

Authoring rules

© Copyright IBM Corporation 2020

Figure 7-35. Decision table size

While the decision table editor can support creating large tables, of thousands of rows, such large tables are challenging to maintain. A suggested practice is to keep tables under 1000 rows.

7.4. Advanced Rule Language (ARL)



Advanced Rule Language (ARL)

Figure 7-36. 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:

BAL	<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>
ARL	<pre>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</pre>

Authoring rules

© Copyright IBM Corporation 2020

Figure 7-37. Translation from BAL to ARL

The Advanced Rule Language is designed for the decision engine. This rule language has a similar syntax to Java 7, with some rule-focused additions.

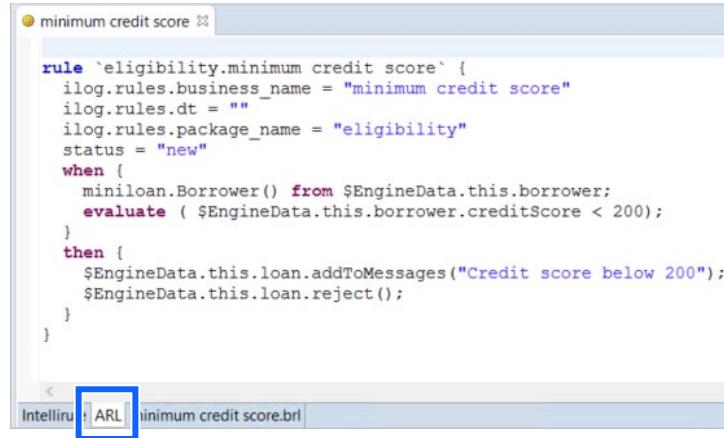
The BAL rule on this slide means that when a customer asks for a loan but the credit score is less than 200, the request is denied. The message 'Credit score below 200' is attached to the loan application.

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**, **rulertask**, 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 window titled "minimum credit score" displays the ARL (Abstract Rule Language) code for a rule named "eligibility.minimum credit score". The code defines a rule with a business name of "minimum credit score", package name "eligibility", and status "new". It has a single condition (when) that checks if a borrower's credit score is less than 200, and if so, adds a message and rejects the loan. The ARL tab is highlighted with a blue border. The status bar at the bottom shows the file path "minimum credit score.brl".

```
minimum credit score

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 2020

Figure 7-38. Viewing ARL

7.5. Technical rules



Technical rules

Figure 7-39. 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 2020

Figure 7-40. 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 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 2020

Figure 7-41. IRL syntax

- 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;
}
```

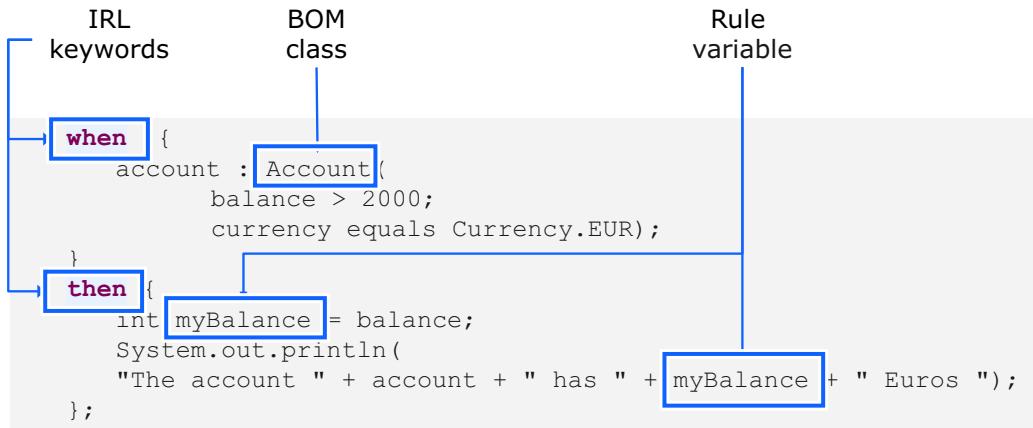
- To import required elements, use the following keywords:
 - **use**: To import ruleset elements (for example, variables)
 - **import**: To import BOM elements



Information

The question mark in variable names such as `?x` is optional.

Technical rule structure



Authoring rules

© Copyright IBM Corporation 2020

Figure 7-42. Technical rule structure

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 {
  ...
}
```

Authoring rules

© Copyright IBM Corporation 2020

Figure 7-43. 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 Technical Rule wizard, notice that the Type is **IRLRule**

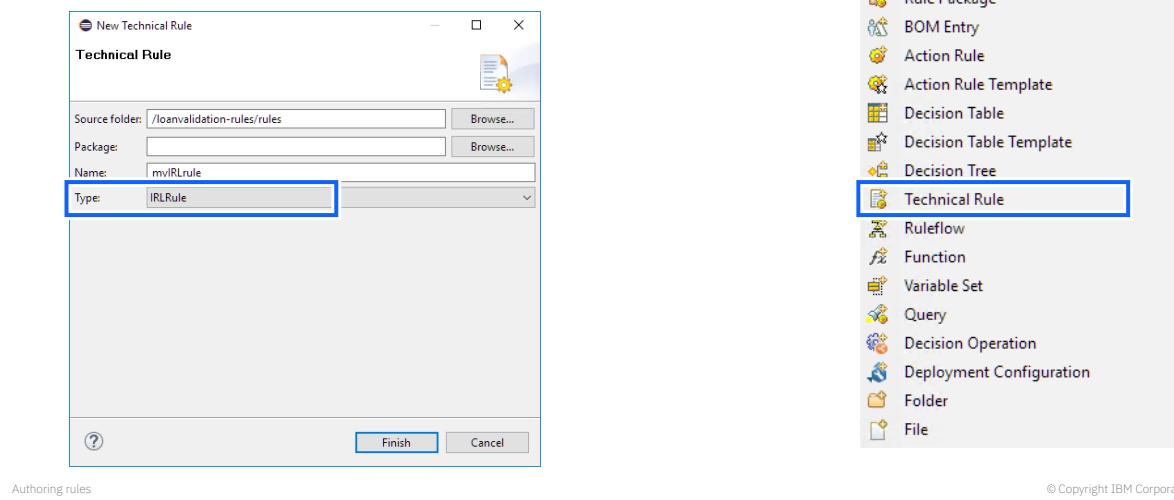


Figure 7-44. Create a technical rule

7.6. Defining objects that are used in rules



Defining objects that are used in rules

Figure 7-45. 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 2020

Figure 7-46. Introduction

To produce decisions, the rule artifacts must manipulate the business objects that your client application provides at run time. You work collaboratively with the business analysts to define these objects, and base their definition on the BOM that models them. Depending on your requirements, you model these objects as *parameters*, as *ruleset variables* in your decision service, or as *objects* in the working memory.

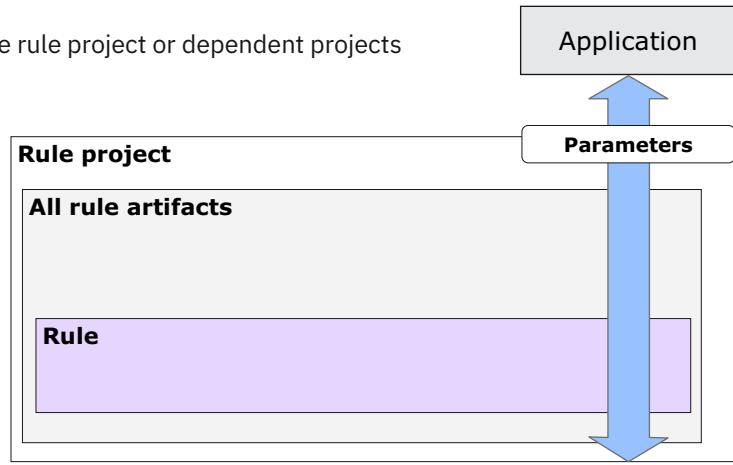


Important

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 2020

Figure 7-47. Parameters

The parameters form the signature of a ruleset and are defined in the decision operation. Parameters are created from the ruleset variables in the rule projects and used to pass data between the ruleset and the client application.

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 2020

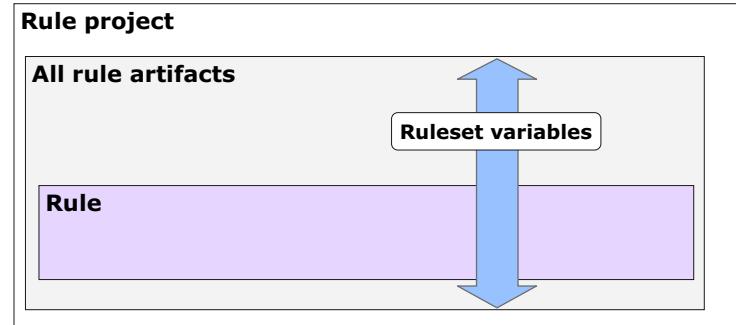


Figure 7-48. Ruleset variables

Ruleset variables define objects internal to the ruleset. Like the parameters, the ruleset variables are accessible to all rule artifacts in your rule project.

You can use ruleset variables in the same manner as internal variables in a regular Java program. You can use ruleset variables to internally manipulate objects that are passed through the parameters from the calling application. By assigning the values from the ruleset parameter to the ruleset variable, you can modify the variables without affecting the original objects. Ruleset variables can be used to exchange information between rules, functions, and ruleflow tasks. For example, they might be used to transfer a value 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 2020

Figure 7-49. Objects in working memory

Your rule artifacts can also work on objects in the working memory. You can also 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 With the verbalization, in BAL With a rule variable, in IRL or BAL
Ruleset variables	With the name, in IRL With the verbalization, in BAL With a rule variable, in IRL or BAL
Objects in the working memory	With a rule variable, in IRL or BAL With an automatic variable, in BAL
Members (methods, attributes) of an object	With the name, in IRL With the verbalization, in BAL With a rule variable, in IRL or BAL

Authoring rules

© Copyright IBM Corporation 2020

Figure 7-50. Working with objects in rule artifacts

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

Authoring rules

© Copyright IBM Corporation 2020

Figure 7-51. Using 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

Authoring rules

© Copyright IBM Corporation 2020

Figure 7-52. 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:

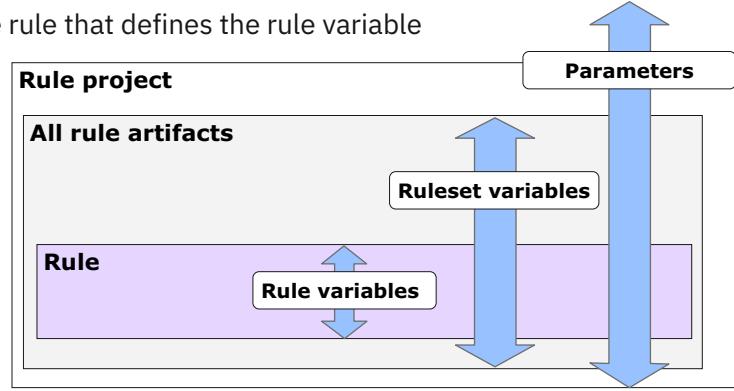
```
if 'the loan report' is approved;
```

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 2020

Figure 7-53. Rule variables (1 of 2)

Rule variables (2 of 2)

- Working memory
 - Objects in the working memory cannot be named or verbalized
 - Bind them to rule variables or automatic variables
- Simplify rule authoring
 - Use rule variables to simplify long vocabulary phrases
 - Improves clarity

Figure 7-54. Rule variables (2 of 2)

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 7-55. Using automatic variables

Unit summary

- Describe rule languages
- Use the various rule editors to author rule artifacts
- Define the objects that rule artifacts manipulate

© Copyright IBM Corporation 2020

Figure 7-56. 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)

Authoring rules

© Copyright IBM Corporation 2020

Figure 7-57. 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)

The answer is C.

Figure 7-58. Review answers

Exercise: Exploring action rules



Figure 7-59. Exercise: Exploring action rules

Exercise introduction

- Identify the parts of an action rule
- Explain the difference between using automatic variables or rule variables

© Copyright IBM Corporation 2020

Figure 7-60. Exercise introduction

Exercise: Authoring action rules



Figure 7-61. 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

© Copyright IBM Corporation 2020

Figure 7-62. Exercise introduction

Exercise: Authoring decision tables



Figure 7-63. Exercise: Authoring decision tables

Exercise introduction

- Use the decision table editor to create a decision table

© Copyright IBM Corporation 2020

Figure 7-64. 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

© Copyright IBM Corporation 2020

Figure 8-1. Unit objectives

Topics

- Simplifying vocabulary with categories
- Defining domains
- Static domains
- Dynamic domains
- Updating dynamic domains in Decision Center

© Copyright IBM Corporation 2020

Figure 8-2. Topics

8.1. Simplifying vocabulary with categories



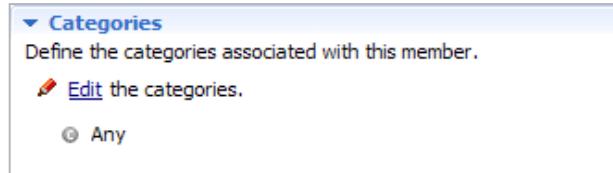
Simplifying vocabulary with categories

© Copyright IBM Corporation 2020

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 2020

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.

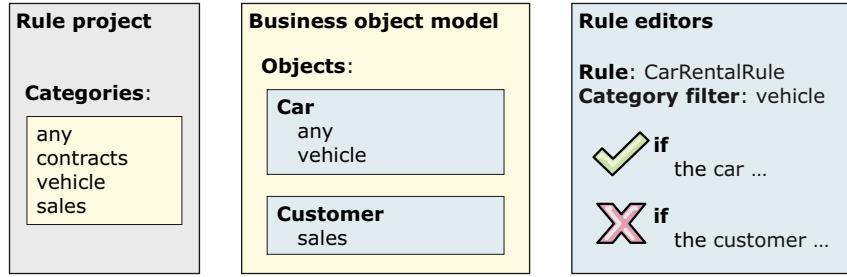
You tag BOM elements with categories to specify whether these BOM elements can be used in specific rules.

- 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.

Use categories to simplify the list of vocabulary available in the rule editors, and ensure 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 2020

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 category2	category1	visible
category1 category2	category3 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 2020

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
- In the BOM element
- In the rule

8.2. Defining domains



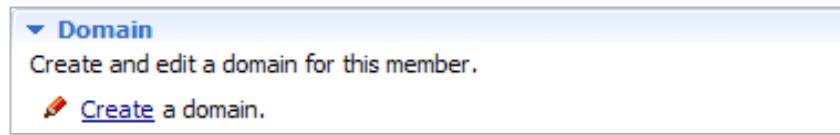
Defining domains

© Copyright IBM Corporation 2020

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 2020

Figure 8-8. Domains section

Domains restrict the values of the BOM elements. Domains can be set on classes, attribute types, method return types, and arguments.

The possible types of domains include:

- **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 you author rule artifacts, code completion in the editors uses domains to propose valid values, and errors and warnings are reported that help validate the values that you specify.
- When you analyze rules, domain types in the BOM are used to check the consistency of action rule semantics.

8.3. Static domains



Static domains

© Copyright IBM Corporation 2020

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

Customizing rule vocabulary with categories and domains

© Copyright IBM Corporation 2020

Figure 8-10. Domains: Literals

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

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 (*)
- 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

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

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"

Customizing rule vocabulary with categories and domains

© Copyright IBM Corporation 2020

Figure 8-14. Domains: Other

As another example of a domain of the Other type, consider the next 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



Dynamic domains

© Copyright IBM Corporation 2020

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 2020

Figure 8-16. Domains: Dynamic (1 of 2)

As their name indicates, domains of type `Dynamic` are managed outside of the business object model. Changes to the domain source file are automatically updated in the BOM.

Similar to other types of domains, you create a domain of type `Dynamic` in the BOM editor. You can manage the domain values in an Excel file.



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

Customizing rule vocabulary with categories and domains

© Copyright IBM Corporation 2020

Figure 8-17. Domains: Dynamic (2 of 2)

Dynamic domains are based on a plug-in, and setting up a domain of type Dynamic can take some time.

However, Operational Decision Manager provides a predefined plug-in and the associated extension points when the source for the values is a Microsoft Excel spreadsheet.

If the source is a Microsoft Excel spreadsheet, then you can do the full creation of the dynamic domain in Rule Designer. Operational Decision Manager predefines all the required elements for you:

- You do not have to create the plug-in that reads from that type of source.
- You do not have to create the associated extension point to Rule Designer and to Decision Center.
- You do not have to deploy the extension points to both environments.

This course includes an exercise on how to create dynamic domains with a Microsoft Excel spreadsheet as the source for the values.

If the source is other than a Microsoft Excel spreadsheet, you must create the plug-in that reads from that source. You create the associated extension point to Rule Designer and to 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

Customizing rule vocabulary with categories and domains

© Copyright IBM Corporation 2020

Figure 8-18. Dynamic domains: Microsoft Excel source (1 of 4)

The **resources** folder is where Operational Decision Manager stores the files that it must share between Rule Designer and Decision Center.

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 that defines the dynamic domain to Decision Center, you also deploy the content of this **resources** folder, that is, the Excel spreadsheet that defines the properties of the domain.

You publish a rule project and its contents from Rule Designer to Decision Center to have it 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

Customizing rule vocabulary with categories and domains

© Copyright IBM Corporation 2020

Figure 8-19. Dynamic domains: Microsoft Excel source (2 of 4)

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
10	Recognition	return "F90";	Recognition
11	Sick	return "G66";	Sick
12	Strike	return "H71";	Strike
13	Vacation	return "J10";	Vacation
14	VolunteerFireAndRescue	return "K29";	Volunteer Fire and Rescue
15	WithoutPay	return "I82";	Without Pay
16			

Customizing rule vocabulary with categories and domains

© Copyright IBM Corporation 2020

Figure 8-20. Dynamic domains: Microsoft Excel source (3 of 4)

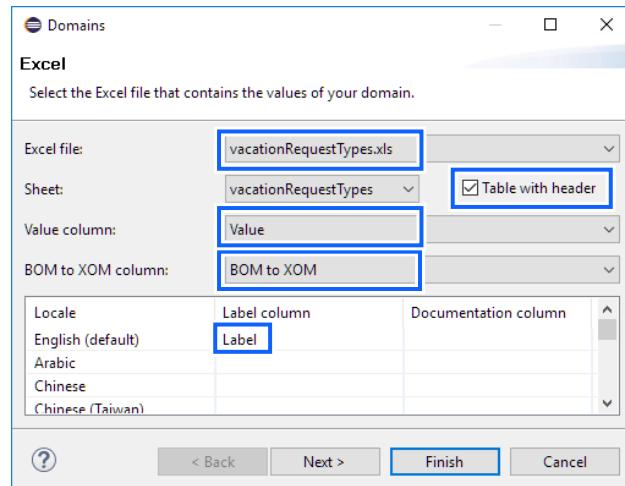
The Documentation column is optional for all locales, including the default one, which is the locale of your Eclipse application.

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 2020

Figure 8-21. Dynamic domains: Microsoft Excel source (4 of 4)

After you create the dynamic domain, the values of the dynamic domain are displayed in the BOM editor. The values are also available in the completion menu of the rule editors. You can now use them when authoring your rules.

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.

You have an exercise in this unit on how to create a dynamic domain that is based on a Microsoft Excel spreadsheet.

This course does not cover how to create the plug-in or deploy an extension point for dynamic domains that are based on a source other than a Microsoft Excel spreadsheet.

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

Customizing rule vocabulary with categories and domains

© Copyright IBM Corporation 2020

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.

The Intellirule editor suggests the values from the enumerated domains when you author action rules based on such domains. An action rule that is based on an enumerated domain (Literals, Static References, and Dynamic) does not compile when it uses a value outside this domain.

Complex domains are not enforced at the action rule level. You might receive warnings that indicate that the action rule is not applicable, but the action rule is not considered to be erroneous.

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 2020

Figure 8-23. Automatic creation of domains

8.5. Updating dynamic domains in Decision Center



Updating dynamic domains in Decision Center

© Copyright IBM Corporation 2020

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 2020

Figure 8-25. Dynamic domains in Decision Center

Business users can work with dynamic domains in Decision Center. If you create a dynamic domain in Rule Designer, the Excel file is stored in the rule project **resources** folder. When you publish the decision service to Decision Center, the dynamic domain Excel file is included.

Accessing the domain file

- 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

The screenshot shows the 'Decision Artifacts' tab selected in the top navigation bar. Below it, the 'Resources' section is displayed. The left sidebar lists projects: 'trucksAndDrivers-tests', 'trucksAndDrivers-bom' (which is expanded to show 'Resources' and 'Rules'), and 'trucksAndDrivers-rules'. The 'Resources' section has a header with 'Resources' and a 'Filter:' input field. It contains two items: 'vacationRequestTypes-1.xls' and 'vacationRequestTypes.xls'. Both items have a small preview icon, the name 'vacationRequestTypes', the author 'rtsAdmin', and the date 'March 3, 2017'.

Customizing rule vocabulary with categories and domains

© Copyright IBM Corporation 2020

Figure 8-26. Accessing the domain file

In Decision Center, business users access the dynamic domain Excel file from the **Decision Artifacts** tab.



Note

All **Types** must be selected on the **Types menu** to view resource artifacts.) The default view in the Business console **Decision Artifacts** tab is set to show only **Rules and Decision Tables**.

The file can be downloaded, a new version can be uploaded, or the file can be deleted.

Updating dynamic domain files

- 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

Customizing rule vocabulary with categories and domains

© Copyright IBM Corporation 2020

Figure 8-27. Updating dynamic domain files

The process of updating a dynamic domain in Decision Center includes the following tasks:

1. Download the dynamic domain Excel file.
2. Modify the file.
3. Upload the modified Excel file.
4. Apply the changes from the file to the BOM.

Viewing domain changes

- In the **Model** tab, you can review the domain and the changes to the domain

Domain	Changes	Provider	Project
domains.VacationRequestType	- Administrative	vacationRequestTypes.xls	trucksAndDrivers-bom
Administrative	Educational		
Educational	FamilyPersonal		
FamilyPersonal	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 2020

Figure 8-28. Viewing domain changes

The **Model** tab lists the dynamic domains that are used in the decision service, and shows domain values and other details, such as 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.

Any changes made to the Excel file are visible in the **Model** tab. On this slide, you see that the **Administrative** vacation request type value was removed, and the **Training** vacation request type was added.

Unit summary

- Simplify rule authoring by using categories
- Define domains

© Copyright IBM Corporation 2020

Figure 8-29. 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.

Customizing rule vocabulary with categories and domains

© Copyright IBM Corporation 2020

Figure 8-30. 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 8-31. Review answers

Exercise: Working with static domains



Figure 8-32. Exercise: Working with static domains

Exercise introduction

- Create various types of static domains
- Use domains in rules

© Copyright IBM Corporation 2020

Figure 8-33. Exercise introduction

Exercise: Working with dynamic domains



Figure 8-34. 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

© Copyright IBM Corporation 2020

Figure 8-35. 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

© Copyright IBM Corporation 2020

Figure 9-1. Unit objectives

Topics

- Searching for rule artifacts
- Querying rules
- Extracting rulesets

© Copyright IBM Corporation 2020

Figure 9-2. Topics

9.1. Searching for rule artifacts



Searching for rule artifacts

© Copyright IBM Corporation 2020

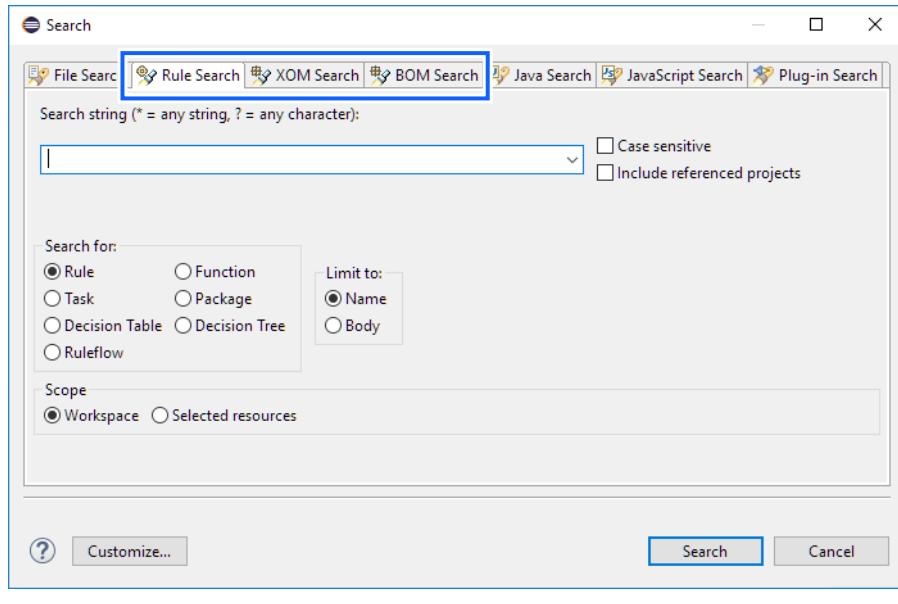
Figure 9-3. Searching for rule artifacts

Search rule artifacts

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 2020

Figure 9-4. Search rule artifacts

With the Rule Designer search features, you can find artifacts of interest.

- You can search packages and rule files, the BOM, and the XOM, by using the Search function, as you learn in the next exercise.
- You can also search for dependencies between rules; that is, search for rules that the execution of other rules might affect, or for rules that affect the execution of other rules.

Search for rule dependencies

Search for rules that the execution of other rules affects

Search for rules that affect the execution of other rules

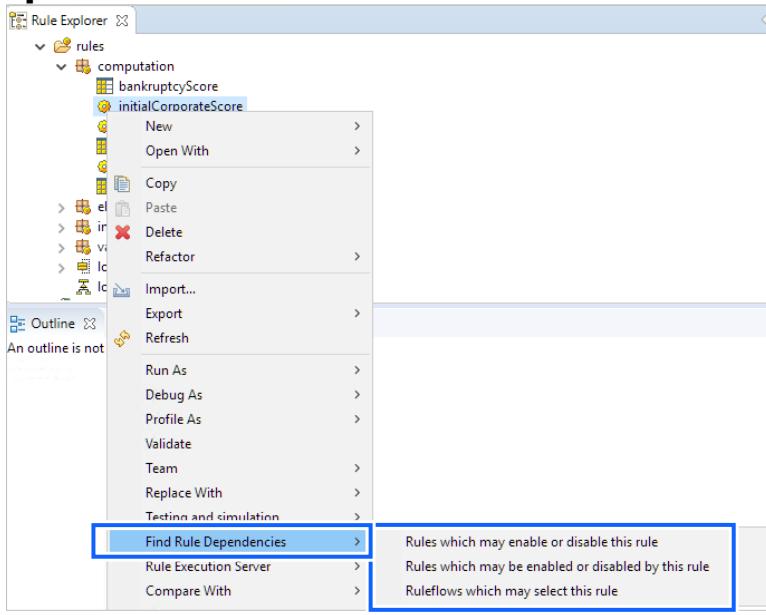


Figure 9-5. Search for rule dependencies

To search for rule dependencies, right-click the rule of interest, click **Find Rule Dependencies**, and select what you want to search for.

- If you want to find the rules that might enable or disable the applicability of the rule of interest, select **Rules which may enable or disable this rule**.
- If the rule of interest might enable or disable the applicability of other rules, and you want to find these other rules, select **Rules which may be enabled or disabled by this rule**.
- Select **Ruleflows that may select this rule** to find the ruleflows that might select the rule of interest.

When you do such a search, you run a predefined **query** on the behavior of the rules.

9.2. Querying rules



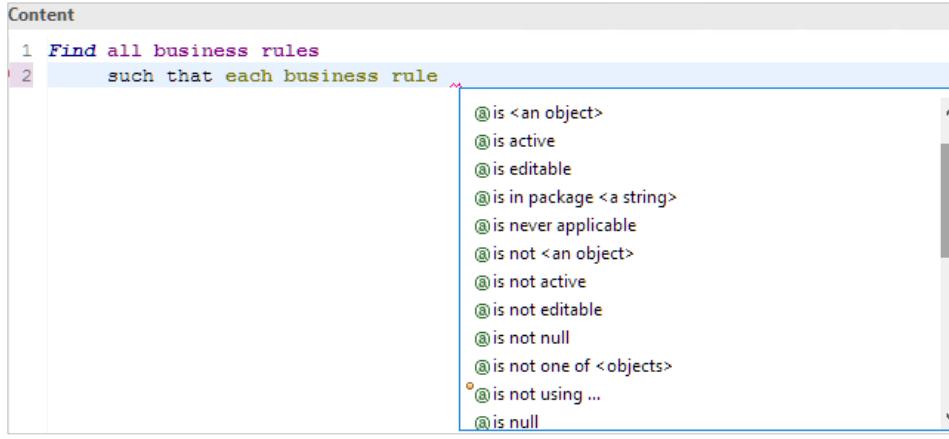
Querying rules

© Copyright IBM Corporation 2020

Figure 9-6. Querying rules

Queries

- In addition to basic searches based on static content or predefined queries, you can define queries for custom searches
- You can create queries in Rule Designer and in Decision Center Business console



Working with queries

© Copyright IBM Corporation 2020

Figure 9-7. 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 Business console.

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

Figure 9-8. When queries are useful (1 of 2)

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)

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

Working with queries

© Copyright IBM Corporation 2020

Figure 9-10. Business Query Language (BQL)

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 2020

Figure 9-11. Query conditions

When you write a query condition, you start by selecting a project element type. You then refine the search by filtering on its properties, its definition, or its behavior.

By default, queries are run on action rules.

You can also search on the following elements:

- Specific types of business rules: Action rules, decision tables, decision trees
- Rule packages, technical rules, ruleflows, templates, functions, variables, or variable sets

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 2020

Figure 9-12. Example of rule query conditions

After you select the element that you want to query, you add a filter by using the **such that** statement with the appropriate conditions, as shown in these 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

Working with queries

© Copyright IBM Corporation 2020

Figure 9-13. Filter on properties

You can query rules by using filters on any rule property, including user-defined properties, classes, data members, and methods that are referenced in rules, or the full text of the rule (by using a text string).

1+1=2 Example

Find all business rules such that the status of each business rule is new

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

Working with queries

© Copyright IBM Corporation 2020

Figure 9-14. Filter on definitions

You can query rules by using filters on project element definitions to find rules that read or modify the value of a member, or call a method.

You set such filters by using the query predicates of the following (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

You can also use negations of these predicates.

1+1=2 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]

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

Working with queries

© Copyright IBM Corporation 2020

Figure 9-15. Filter on behavior

You can run semantic queries that 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**
 - 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
- By default, actions are not run when you run the query
 - You must select **Run query actions** after you run the query

Working with queries

© Copyright IBM Corporation 2020

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.

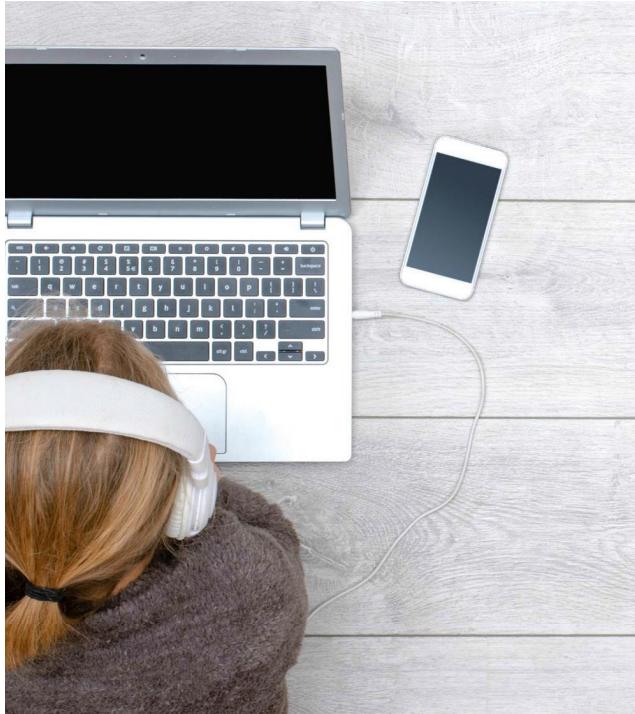
To run the actions on the result of the query, select the **Run query actions** check box when you run the query.

1+1=2 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

9.3. Extracting rulesets



Extracting rulesets

© Copyright IBM Corporation 2020

Figure 9-17. Extracting rulesets

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.

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 9-18. Queries and ruleset extractor

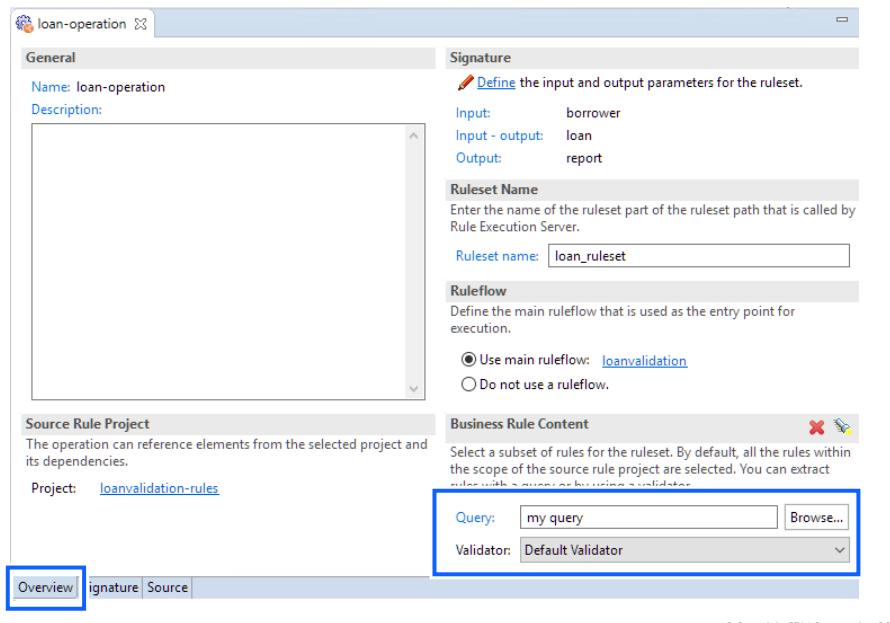
Queries can also be used to create a ruleset extractor. To create a ruleset extractor, create a query and then create the ruleset extractor from this query. You apply the ruleset extractor to create your ruleset archive so that the extracted ruleset contains only a restricted number of rule artifacts.

Ruleset archive

To pass a subset of rules from a decision operation

- Use a query to extract the rules
- Write a validator class to select the required rules

Specify the extractor in the decision operation



Working with queries

© Copyright IBM Corporation 2020

Figure 9-19. Ruleset archive

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.

When you define a decision operation, the default behavior is to include all the rules in all the referenced rule projects. The extractor limits the rules that are included in the decision operation.

For example, you can write a query that finds all business rules that have the status that is defined as follows:

Find all business rules such that the status of each business rule is defined.

Then, you use this query to define a ruleset extractor. When you extract the ruleset from the rule project, only the rules that have the status that is defined are included in the ruleset.

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

© Copyright IBM Corporation 2020

Figure 9-20. Unit summary

Review questions



1. Queries are written in which language?
 - a. Business Query Language (BQL)
 - b. Rule Query Language (RQL)
 - c. Intellirule Query Language (IQL)
2. True or False: A query can be used to extract a subset of rules for a decision operation.

Figure 9-21. Review questions

Write your answer here:

- 1.
- 2.

Review answers



1. Queries are written in which language?

- a. Business Query Language (BQL)
- b. Rule Query Language (RQL)
- c. Intellirule Query Language (IQL)

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.

Figure 9-22. Review answers

Exercise: Working with searches and queries



Figure 9-23. Exercise: Working with searches and 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

© Copyright IBM Corporation 2020

Figure 9-24. Exercise introduction

Unit 10. Debugging rulesets

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 run and debug rulesets
- Work with automatic exception handling
- Work with Rule Designer debugging tools

© Copyright IBM Corporation 2020

Figure 10-1. Unit objectives

Topics

- Working with launch configurations
- Automatic exception handling
- Using Rule Designer debugging tools

© Copyright IBM Corporation 2020

Figure 10-2. Topics

10.1. Working with launch configurations



Working with launch configurations

© Copyright IBM Corporation 2020

Figure 10-3. Working with launch configurations

Running and debugging decision services

- Before deploying rules, 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 Run mode or in Debug mode
 - **Run:** 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

Debugging rulesets

© Copyright IBM Corporation 2020

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.

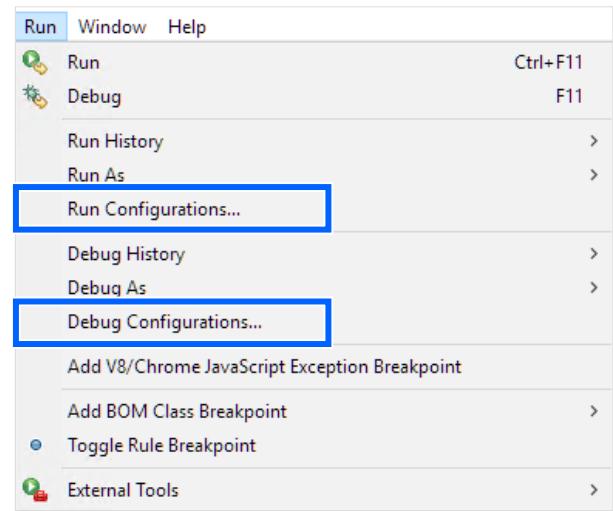
Depending on whether you run the ruleset with or without the debugger, the following terms apply:

- *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



Debugging rulesets

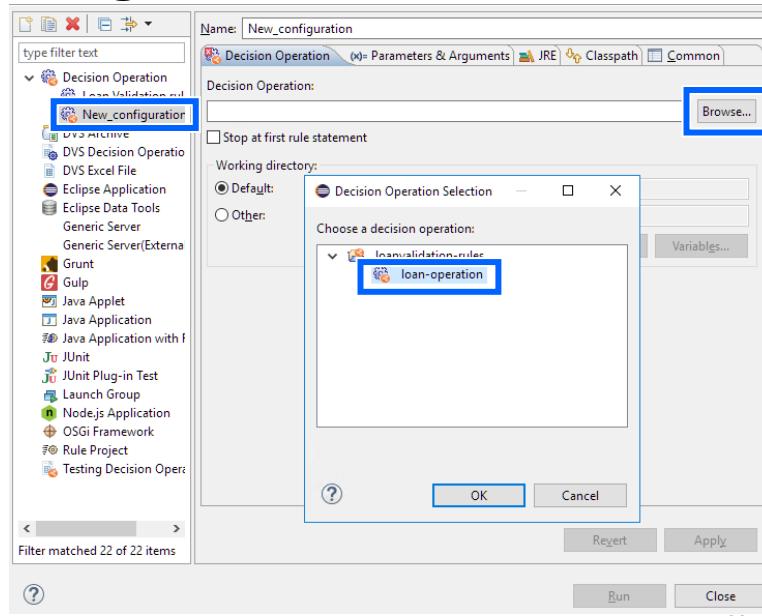
© Copyright IBM Corporation 2020

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)

To create the launch configuration, select **Decision Operation** in the Run Configuration wizard



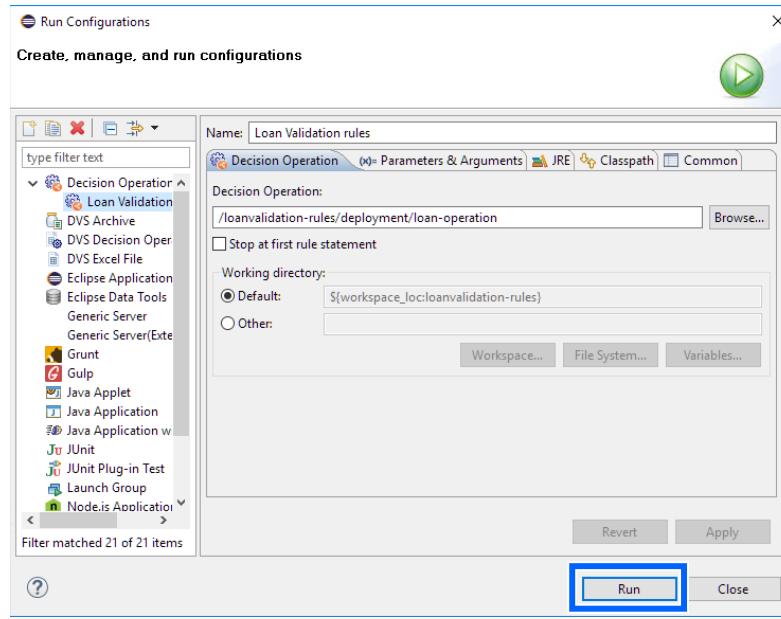
© Copyright IBM Corporation 2020

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.

Run configuration

With a run configuration, the ruleset executes normally



© Copyright IBM Corporation 2020

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.

Debug configuration

With a debug configuration, ruleset execution switches to debug mode

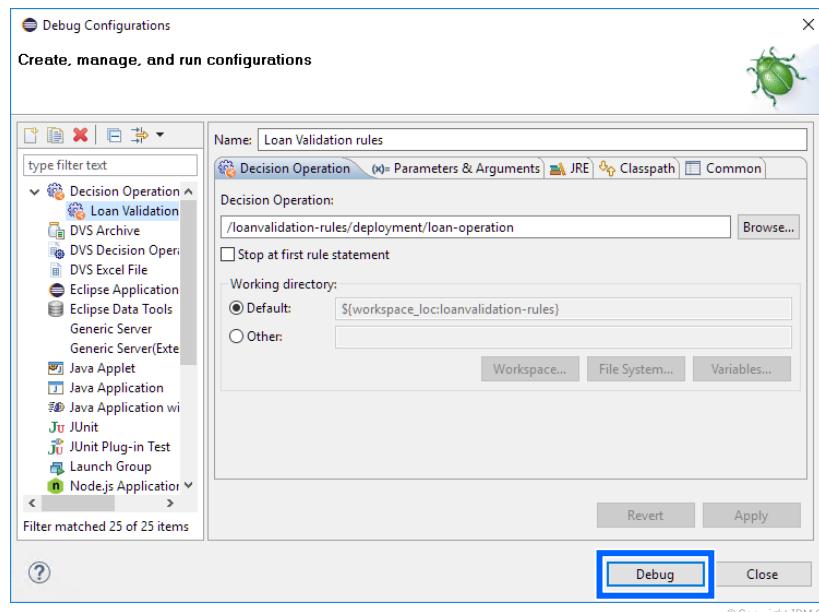
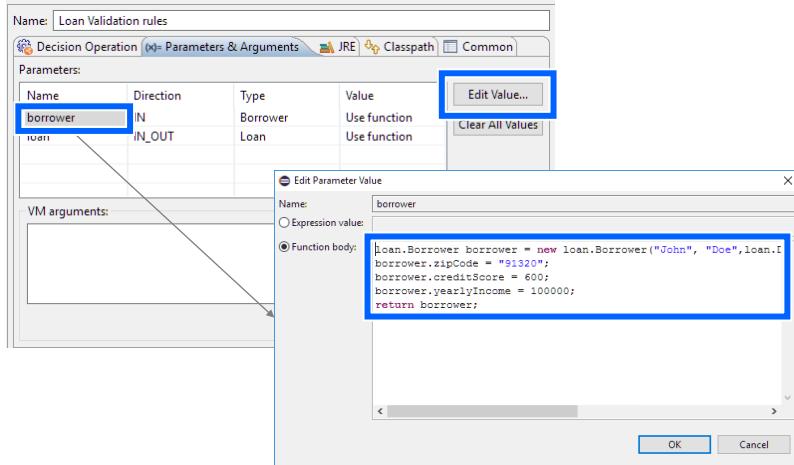


Figure 10-8. Debug configuration

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



Debugging rulesets

© Copyright IBM Corporation 2020

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



Automatic exception handling

© Copyright IBM Corporation 2020

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

Debugging rulesets

© Copyright IBM Corporation 2020

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.

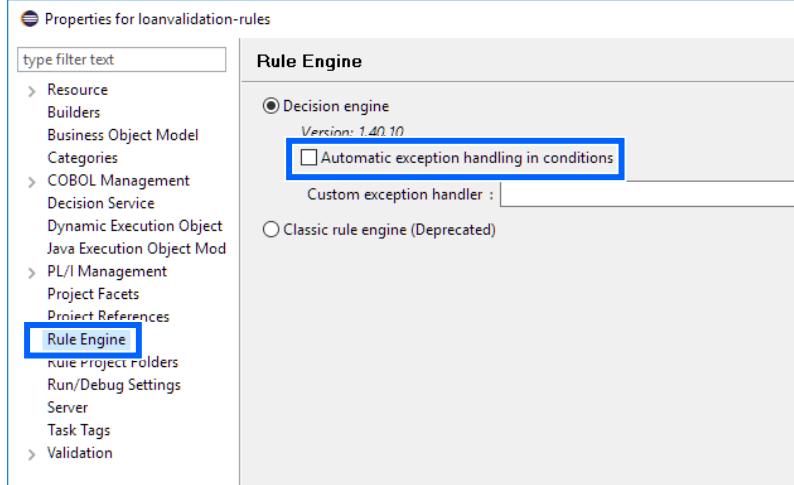
To handle these types of exceptions, rule authors needed to add checks in their conditions to test that an object is not null before testing other conditions on this object. However, you can enable automatic exception handling so that the engine sees the missing data as an “unknown value” instead of 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.

Enabling automatic exception handling

Enable automatic exception handling at the decision service project level

In the Properties dialog box, select **Rule Engine > Automatic exception handling**



Debugging rulesets

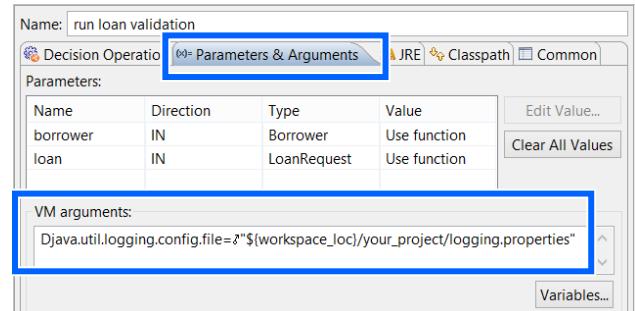
© Copyright IBM Corporation 2020

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



Debugging rulesets

© Copyright IBM Corporation 2020

Figure 10-13. Capturing trace logs in Rule Designer

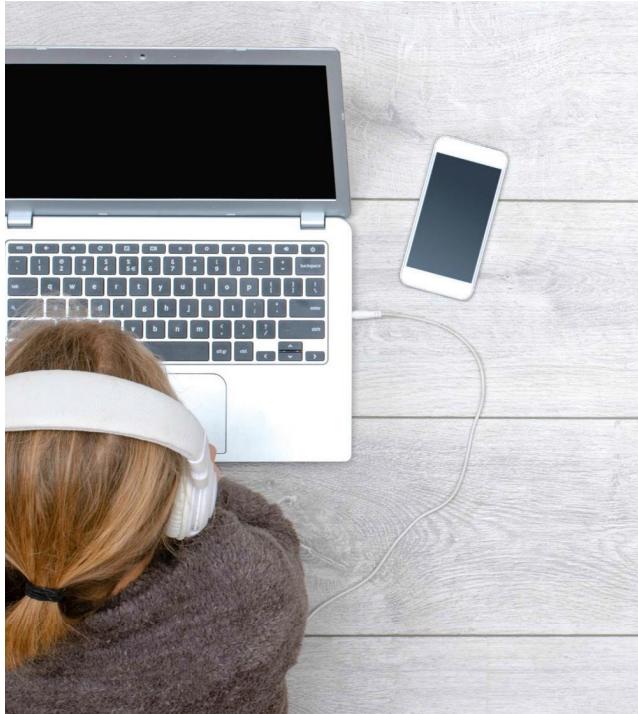
To gather more information about the exceptions, you can define a logging configuration file, and make that file available to your JVM or to the debug configuration.

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

© Copyright IBM Corporation 2020

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

Debugging rulesets

© Copyright IBM Corporation 2020

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:

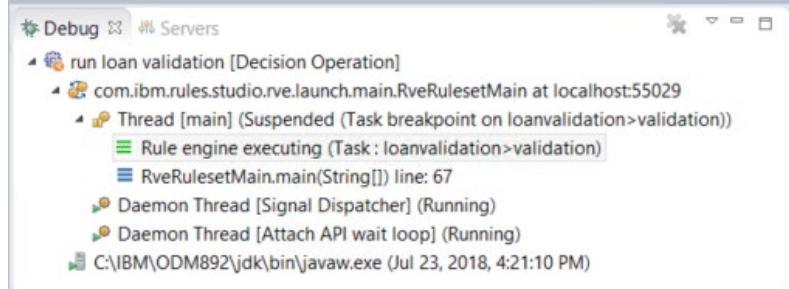
- Inspect rule execution
- Inspect the state of the engine
- Set breakpoints on classes, objects, rules, decision tables, and ruleflows

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



Debugging rulesets

© Copyright IBM Corporation 2020

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)

Debugging rulesets

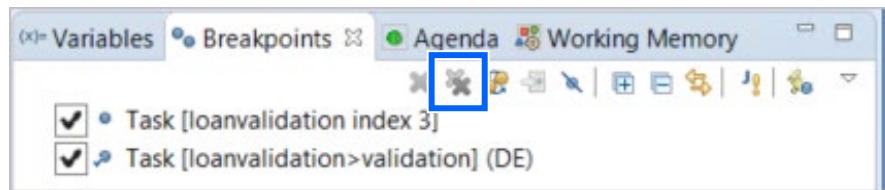
© Copyright IBM Corporation 2020

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



Debugging rulesets

© Copyright IBM Corporation 2020

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

Debugging rulesets

© Copyright IBM Corporation 2020

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 run and debug rulesets
- Work with automatic exception handling
- Work with Rule Designer debugging tools

© Copyright IBM Corporation 2020

Figure 10-20. Unit summary

Review questions



1. Which tool do you use to run a rule project in Rule Designer?
 - a. Decision Warehouse
 - b. Launch configuration
 - c. BOM-to-XOM mapping
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.

Debugging rulesets

© Copyright IBM Corporation 2020

Figure 10-21. Review questions

Write your answers here:

- 1.
- 2.
- 3.

Review answers (1 of 2)



1. Which tool do you use to run a rule project in Rule Designer?
 - a. Decision Warehouse
 - b. Launch configuration
 - c. BOM-to-XOM mapping

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. You can debug iteratively while developing rule artifacts.

Figure 10-22. Review answers (1 of 2)

Exercise: Debugging a ruleset

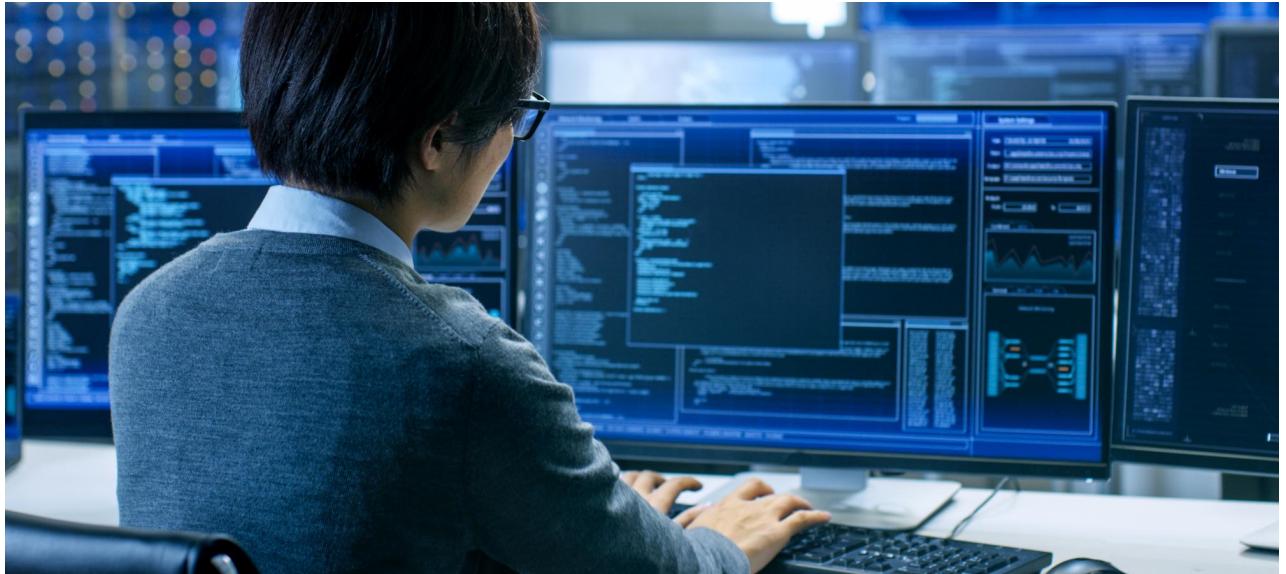


Figure 10-23. Exercise: Debugging a ruleset

Exercise introduction

- Use automatic exception handling
- Set breakpoints in rules, decision tables, and ruleflows
- Run a debugging session

© Copyright IBM Corporation 2020

Figure 10-24. Exercise introduction

Unit 11. Enabling tests and simulations

Estimated time

01:00

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

© Copyright IBM Corporation 2020

Figure 11-1. Unit objectives

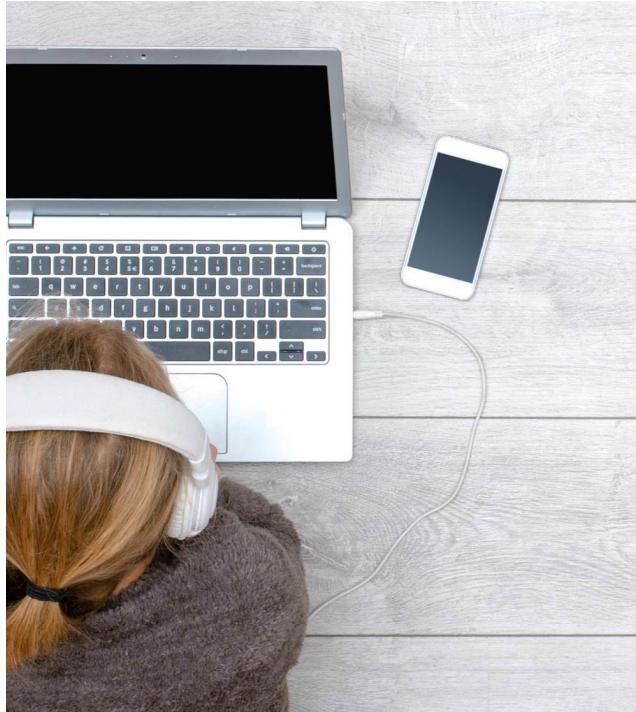
Topics

- Overview of testing and simulation
- Setting up testing and simulation
- Working with scenarios

© Copyright IBM Corporation 2020

Figure 11-2. Topics

11.1. Overview of testing and simulation



Overview of testing and simulation

© Copyright IBM Corporation 2020

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)

Enabling tests and simulations

© Copyright IBM Corporation 2020

Figure 11-4. What is testing and simulation?

You 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 2020

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
11	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 2020

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

Enabling tests and simulations

© Copyright IBM Corporation 2020

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

Enabling tests and simulations

© Copyright IBM Corporation 2020

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

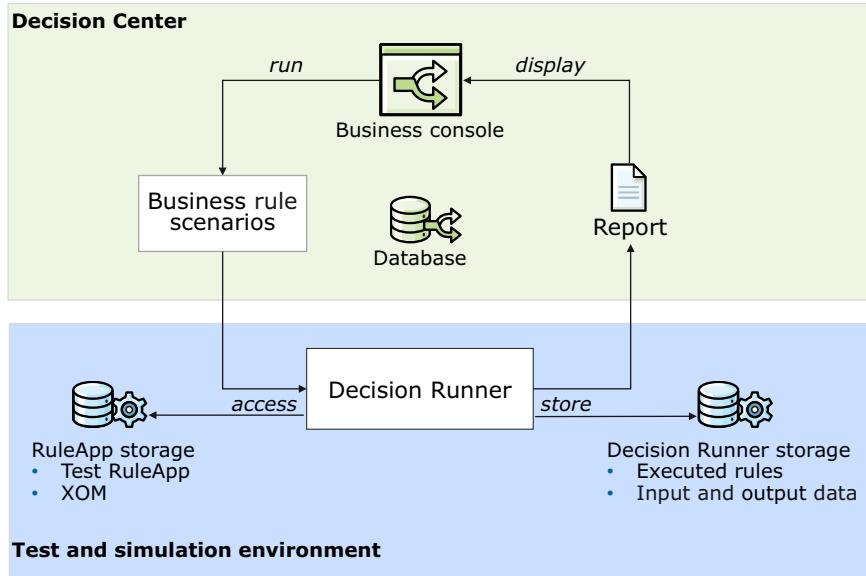


Setting up testing and simulation

© Copyright IBM Corporation 2020

Figure 11-10. Setting up testing and simulation

Enabling remote testing in Business console



Enabling tests and simulations

© Copyright IBM Corporation 2020

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

Enabling tests and simulations

© Copyright IBM Corporation 2020

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

© Copyright IBM Corporation 2020

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

Enabling tests and simulations

© Copyright IBM Corporation 2020

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.

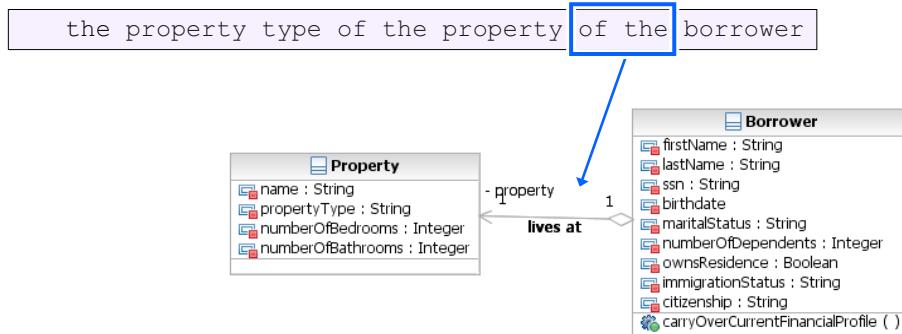
As you see during the exercises, when you create the scenario file template, you select which BOM members to include. The template is generated with the verbalized names of the BOM members. The structure of the scenario file reflects which objects you send as input.

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



Enabling tests and simulations

© Copyright IBM Corporation 2020

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)

The screenshot shows a Microsoft Excel spreadsheet titled "testsuite.xls". The "Scenarios" tab is active, displaying a table with the following columns:

Scenario ID	description	first name	last name	birth date	SSN	credit score	property info	yearly income	zip code	start date	number of monthly payments	amortization period
Scenario 1												

The "PropertyInfo" tab is highlighted in the ribbon. The top sheet contains some descriptive text and links.

Enabling tests and simulations

© Copyright IBM Corporation 2020

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	the borrower	last name	birth date	SSN	credit score	property info	yearly income	zip code	the loan	start date	number of monthly payments	amortization
Scenario 1													

Enabling tests and simulations

© Copyright IBM Corporation 2020

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 screenshot shows a Microsoft Excel spreadsheet titled "testsuite_2.xls [Compatibility Mode] - Microsoft Excel". The spreadsheet contains a single sheet named "Scenarios". The table structure is as follows:

Scenario ID	description	the borrower			the loan					
		first name	last name	birth date	SSN	credit score	property info	early income	zip code	start date
Scenario 1	Joe	Smith	4/12/1983	123456789	60	PropertyInfo1	52,500	12345	4/1/2010	
10										
11										
12										
13										
14										

© Copyright IBM Corporation 2020

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**

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 2020

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

Enabling tests and simulations

© Copyright IBM Corporation 2020

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

Enabling tests and simulations

© Copyright IBM Corporation 2020

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

Enabling tests and simulations

© Copyright IBM Corporation 2020

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 2020

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

Enabling tests and simulations

© Copyright IBM Corporation 2020

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

© Copyright IBM Corporation 2020

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 actual results.
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 2020

Figure 11-26. Review questions

Write your answers here:

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

Review answers



1. True or False: Testing allows business users to validate the behavior of a ruleset by comparing the expected results with actual results.
[The answer is True.](#)
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.](#)
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 11-27. Review answers

Exercise: Enabling rule validation

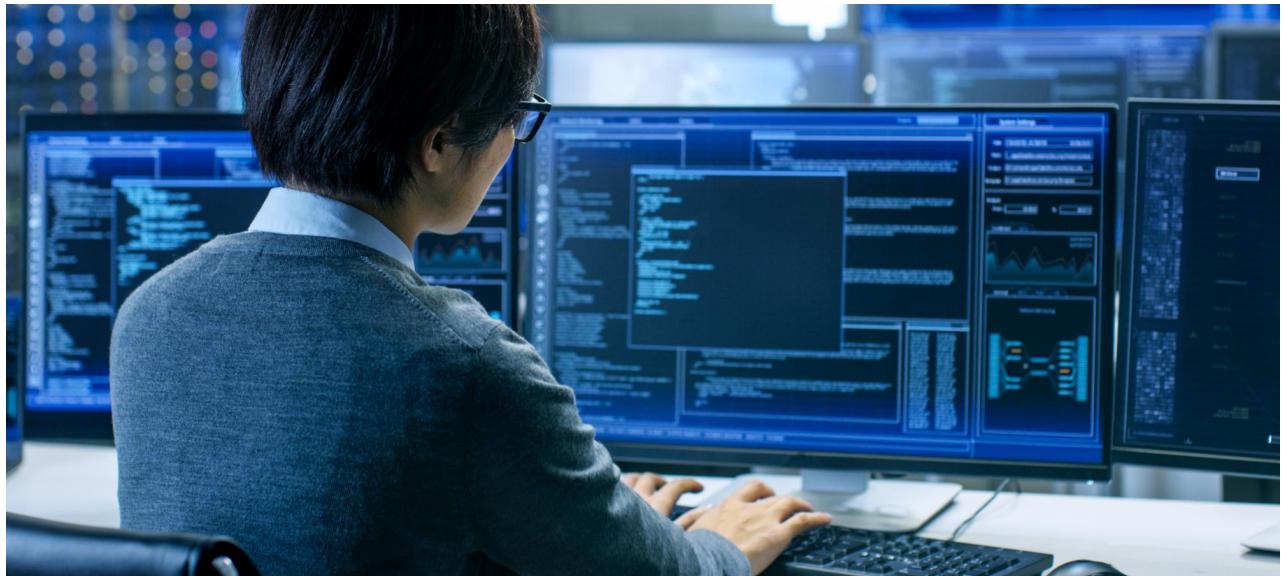


Figure 11-28. Exercise: Enabling rule validation

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

© Copyright IBM Corporation 2020

Figure 11-29. 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

© Copyright IBM Corporation 2020

Figure 12-1. Unit objectives

Topics

- Deploying rules
- Managing XOMs
- Building and deploying with Ant tasks
- Building and deploying with the REST API
- Building projects with the Build Command

© Copyright IBM Corporation 2020

Figure 12-2. Topics

12.1. Deploying rules



Deploying rules

Figure 12-3. Deploying rules

Preparing for deployment

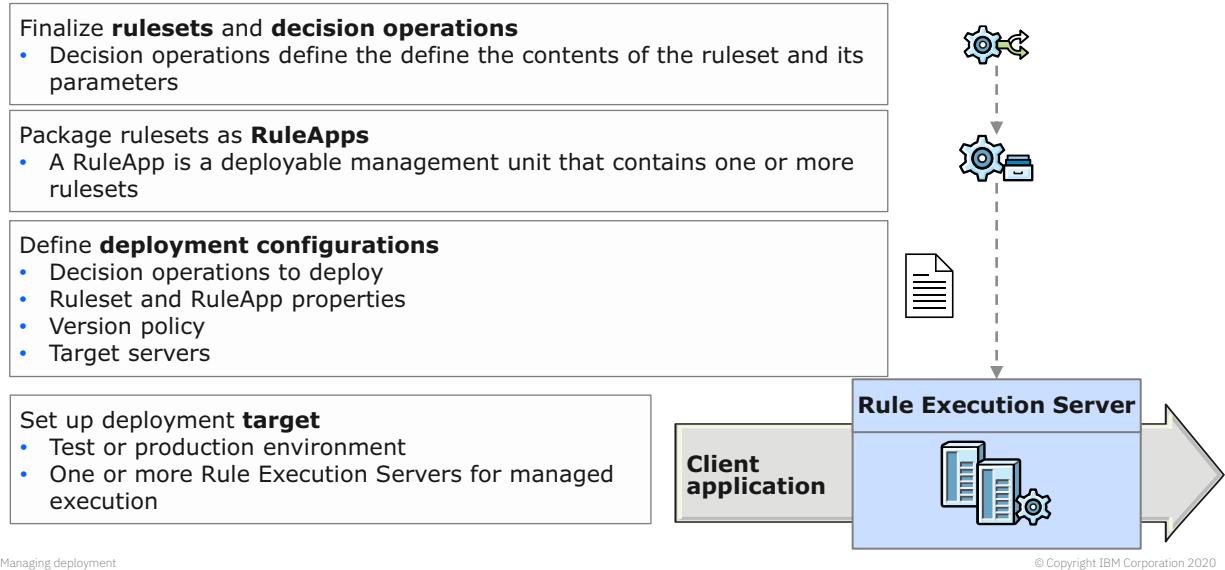


Figure 12-4. Preparing for deployment

When the rules are ready to be used by the client application, you need to package and deploy them to your execution environment.

For managed execution with Rule Execution Server, rulesets are packaged in *RuleApps*. When you deploy a decision service with a deployment configuration, Rule Designer generates a RuleApp archive that contains the rulesets. You can also package and deploy rulesets in other way, for example, by using Ant tasks or REST API.

RuleApp and RuleApp archive

- A RuleApp keeps a record of the following details:
 - RuleApp name
 - RuleApp version
 - Number of rulesets that the RuleApp manages
 - RuleApp creation date
- You archive a RuleApp to a JAR file
 - Example: *RuleApp_name.jar*
- Create a RuleApp archive from a RuleApp
 - Deploy the 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

Managing deployment

© Copyright IBM Corporation 2020

Figure 12-5. RuleApp and 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 and ruleset properties

- Several RuleApp properties and ruleset properties are predefined to handle configuration of the rule engine behavior at run time
- Examples:
 - `ruleset.sequential.trace.enabled`
 - `ruleapp.interceptor.classname`
 - `ruleset.decisionEngine.maxRunningTime`
- Set RuleApp properties to apply to all rulesets in this RuleApp
- Set ruleset properties to apply to a specific ruleset only

Managing deployment

© Copyright IBM Corporation 2020

Figure 12-6. RuleApp 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:

- `ruleset.sequential.trace.enabled` 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.
- `ruleapp.interceptor.classname` 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.

- `ruleset.decisionEngine.maxRunningTime` is a predefined ruleset property that limits how long the ruleset can run.

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.

For more information about RuleApp and ruleset properties, see IBM Knowledge Center:

www.ibm.com/support/knowledgecenter/en/SSQP76_8.10.x/com.ibm.odm.dserver.rules.res.console/topics/con_rescons_rs_prop.html

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

Managing deployment

© Copyright IBM Corporation 2020

Figure 12-7. 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>`

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
- **Canonical** ruleset path includes
 - Name and version number of the RuleApp
 - Name and version number of the ruleset
- **Short** ruleset path
 - Omit the version fields in the ruleset path to refer to the **latest** deployed ruleset

Managing deployment

© Copyright IBM Corporation 2020

Figure 12-8. 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.

Version policy (1 of 2)

- Use version options to control how RuleApp and ruleset versions are numbered and replaced during deployment
- RuleApp version numbering depends whether the RuleApp exists in the server memory:
 - If the RuleApp does not exist, it is deployed to Rule Execution Server with the version number 1.0
 - If the RuleApp exists, it is deployed according to the version policy

Figure 12-9. Version policy (1 of 2)

Version policy (2 of 2)

Policy	Deployed RuleApps	RuleApp to be deployed	Result after deployment
Increment RuleApp major version	/RuleApp/1.0 /ruleset/1.0 /ruleset/2.0	/RuleApp/1.0 /ruleset/1.0	/RuleApp/1.0 /ruleset/1.0 /ruleset/2.0 /RuleApp/2.0 /ruleset/1.0
Increment RuleApp minor version	/RuleApp/1.0 /ruleset/1.0 /ruleset/2.0	/RuleApp/1.0 /ruleset/1.0	/RuleApp/1.0 /ruleset/1.0 /ruleset/2.0 /RuleApp/1.1 /ruleset/1.0
Replace RuleApp version	/RuleApp/1.0 /ruleset/1.0 /ruleset/2.0	/RuleApp/1.0 /ruleset/1.0	/RuleApp/1.0 /ruleset/1.0

Policy	Deployed RuleApps	RuleApp to be deployed	Result after deployment
Increment Ruleset(s) major version	/RuleApp/1.0 /ruleset/1.0 /ruleset/2.0	/RuleApp/1.0 /ruleset/1.0	/RuleApp/1.0 /ruleset/1.0 /ruleset/2.0 /ruleset/3.0
Increment Ruleset(s) minor version	/RuleApp/1.0 /ruleset/1.0 /ruleset/2.0	/RuleApp/1.0 /ruleset/1.0	/RuleApp/1.0 /ruleset/1.0 /ruleset/2.0 /ruleset/2.1
Replace Ruleset(s) versions	/RuleApp/1.0 /ruleset/1.0 /ruleset/2.0	/RuleApp/1.0 /ruleset/1.0	/RuleApp/1.0 /ruleset/1.0 /ruleset/2.0

Managing deployment

© Copyright IBM Corporation 2020

Figure 12-10. Version policy (2 of 2)

Deployment configurations (1 of 3)

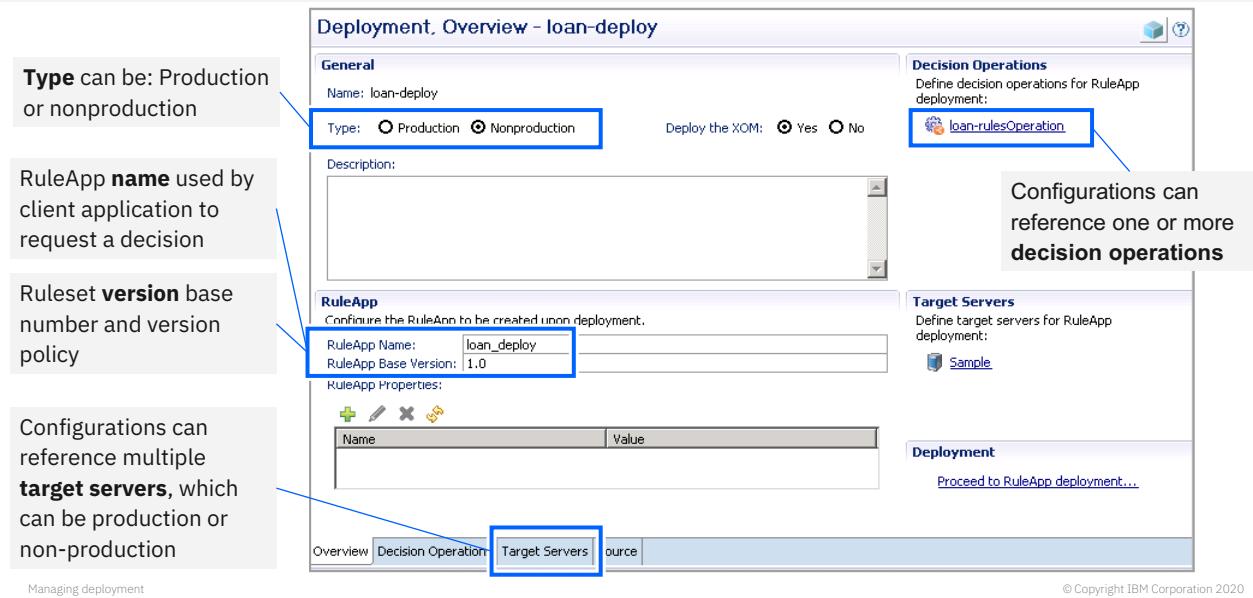


Figure 12-11. Deployment configurations (1 of 3)

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.

A deployment configuration includes the following information:

- A configuration type: production or nonproduction. You can use this option 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 Rule Execution Servers, which can be for nonproduction or production
- The ruleset version base number and version policy

You can create more than one deployment configuration for a decision service. Each deployment configuration can serve a different purpose. For example, one can be used for testing and another for deploying to a production server.

Deployment configurations (2 of 3)

Define the decision operations, ruleset properties, and version policy on the **Decision Operations** tab

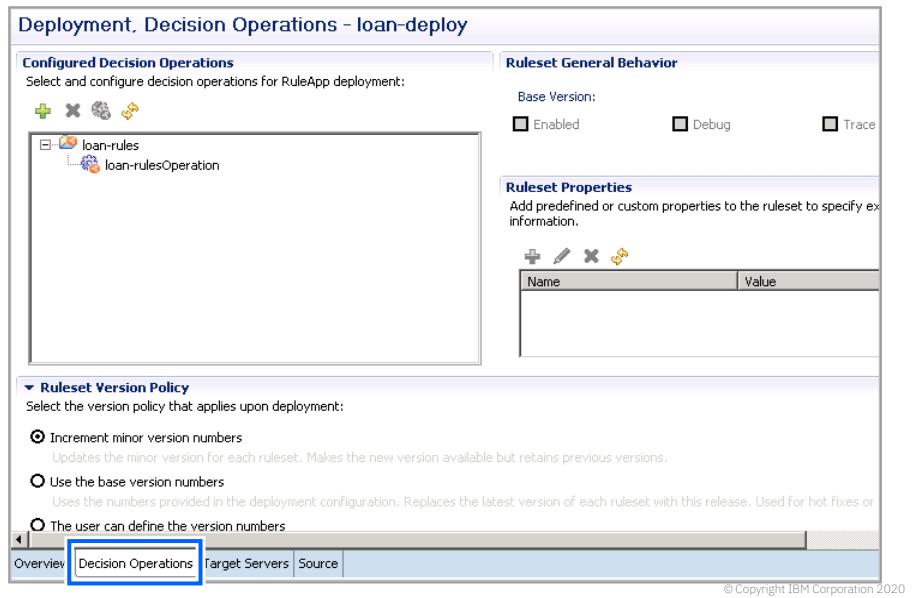
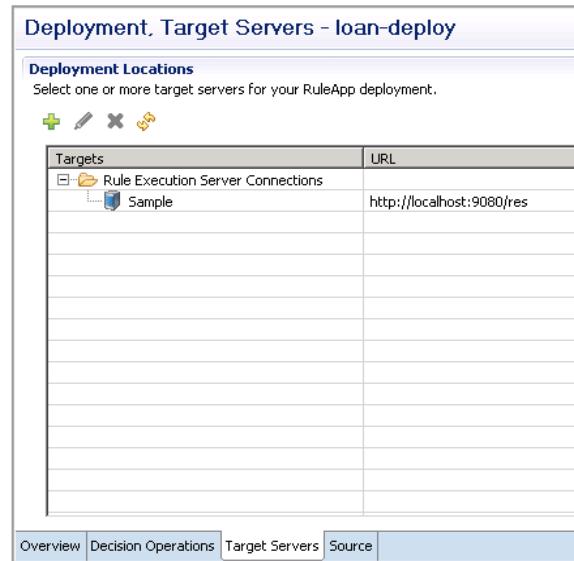


Figure 12-12. Deployment configurations (2 of 3)

On the **Decision Operations** tab, you define the decision operations, ruleset properties, and major/minor version policy.

Deployment configurations (3 of 3)

- 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 2020

Figure 12-13. Deployment configurations (3 of 3)

Configurations can point to multiple target servers, and the servers can be for production or nonproduction.

If the deployment configuration is for a nonproduction, 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. For example, within the governance framework, only decision service releases that are marked complete, can be deployed to a production server.

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 2020

Figure 12-14. Deployment from Decision Center

Business users with administrator rights can deploy 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

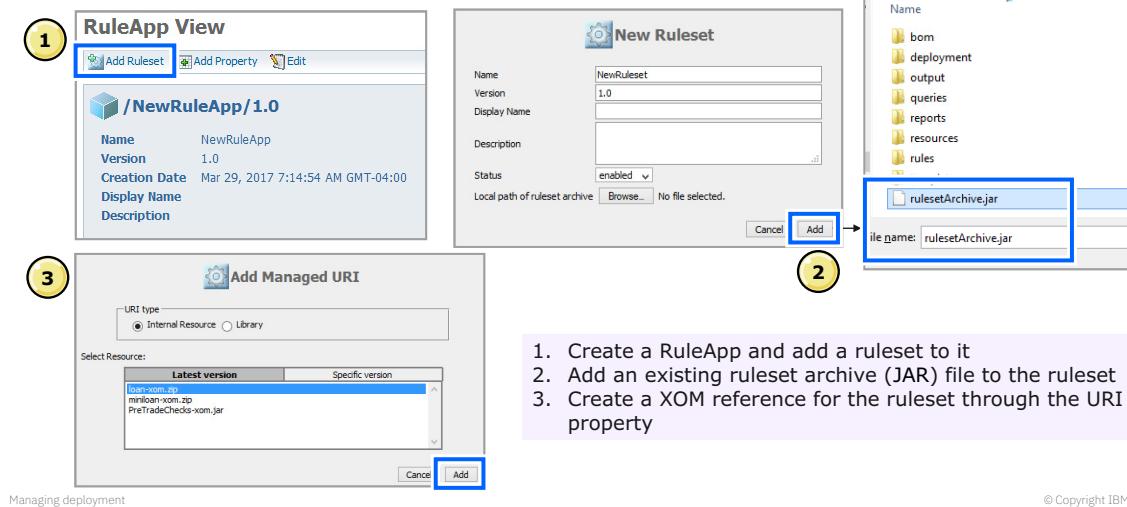


Figure 12-15. Deployment from Rule Execution Server console

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.

12.2. Managing XOMs



Managing XOMs

Figure 12-16. Managing XOMs

XOM deployment with decision services

- XOM management ensures that any RuleApp that is deployed to Rule Execution Server references the correct XOM
- 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

Managing deployment

© Copyright IBM Corporation 2020

Figure 12-17. 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

Managing deployment

© Copyright IBM Corporation 2020

Figure 12-18. Modifying the XOM

Managed Java XOM artifacts

- 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
 - Each library has a unique URI based on the `reslib` protocol
 - Example: `reslib://loan-xom/1.4`

Managing deployment

© Copyright IBM Corporation 2020

Figure 12-19. Managed Java XOM artifacts

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 artifact
- 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

Managing deployment

© Copyright IBM Corporation 2020

Figure 12-20. 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 2020

Figure 12-21. 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

Managing deployment

© Copyright IBM Corporation 2020

Figure 12-22. 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

Managing deployment

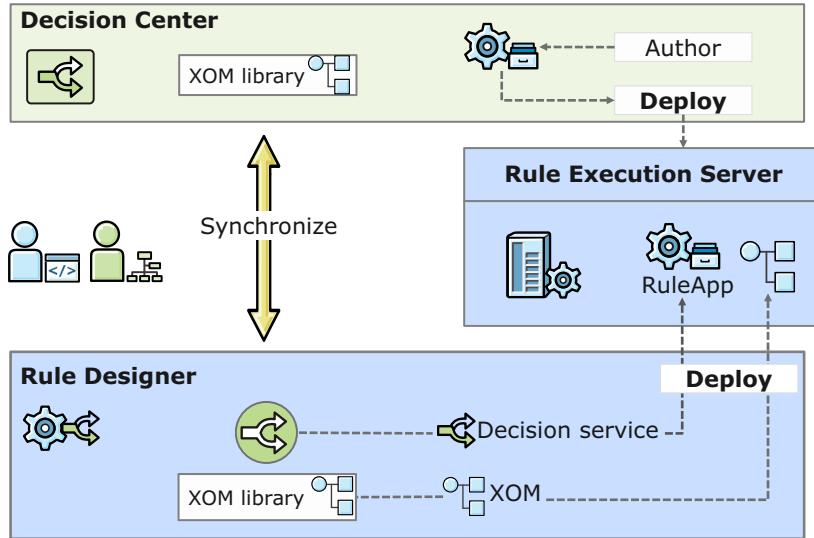
© Copyright IBM Corporation 2020

Figure 12-23. 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 2020

Figure 12-24. 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 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.
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

Managing deployment

© Copyright IBM Corporation 2020

Figure 12-25. 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 (file system or database)
- Managed Java XOMs can be stored 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

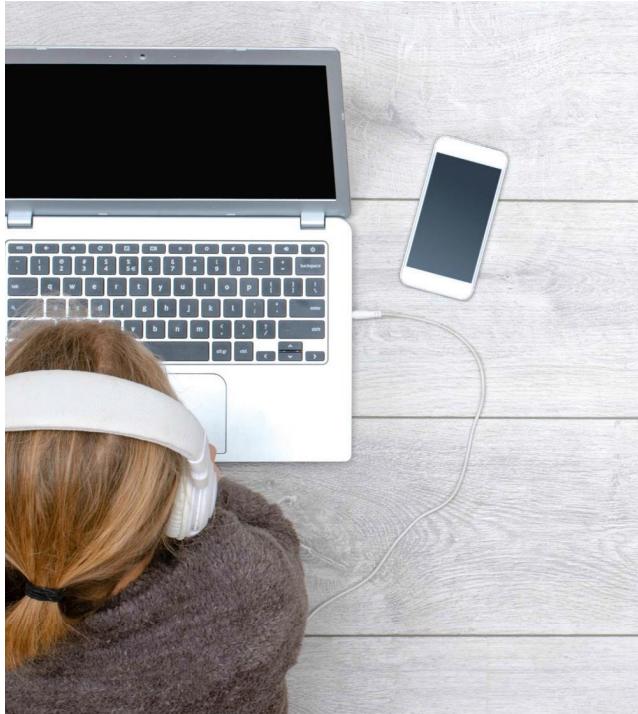
Managing deployment

© Copyright IBM Corporation 2020

Figure 12-26. Managed XOM storage

You can store the managed Java XOM independently of the ruleset, in the persistence layer of Rule Execution Server.

12.3. Building and deploying with Ant tasks



Building and deploying with Ant tasks

Figure 12-27. Building and deploying with Ant tasks

Deployment and management with Ant

- Use Ant scripts to automate RuleApp management
 - Create a `build.xml` file as an Ant script
- Deploy a RuleApp archive by running 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

Managing deployment

© Copyright IBM Corporation 2020

Figure 12-28. 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 12-29. Setting up the Ant environment

12.4. Building and deploying with the REST API



Building and deploying with the REST API

Figure 12-30. Building and deploying with 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

Managing deployment

© Copyright IBM Corporation 2020

Figure 12-31. REST API tool in Rule Execution Server console

Decision Center API console

Use the **Decision Center API console** to manage REST resources

The screenshot shows the IBM Operational Decision Manager Decision Center API console. The main area displays a list of API endpoints under the 'Build' section. The endpoints are:

- POST /v1/decisionservices/{decisionServiceId}/snapshot**: Create a snapshot of a branch in the decision service.
- POST /v1/deployments/{deploymentId}/build**: Build a RuleApp for the deployment configuration.
- POST /v1/deployments/{deploymentId}/deploy**: Deploy a RuleApp to an execution server (Rule Execution Server).
- GET /v1/deployments/{deploymentId}/download**: Download the RuleApp archive for the deployment configuration.
- DELETE /v1/testreports/{testReportId}**: Delete a test report.
- POST /v1/testsuites/{testSuiteId}/run**: Run a test suite.

Managing deployment

© Copyright IBM Corporation 2020

Figure 12-32. Decision Center API console

Decision Center exposes a REST API that you can use to build, test, and deploy decision services. With this REST API, you can easily set up and enforce a continuous deployment process by using the programming language of your choice.

For more information, see IBM Knowledge Center:

https://www.ibm.com/support/knowledgecenter/en/SSQP76_8.10.x/com.ibm.odm.dcenter.ref.dc/topics/con_dc_rest_api_overview.html



Information

The Decision Center API can be used for various administrative tasks, including defining fine-grained permissions. For more information, see the reference for Admin controller permission endpoints in IBM Knowledge Center:

www.ibm.com/support/knowledgecenter/SSQP76_8.10.x/com.ibm.odm.dcenter.ref.dc/topics/dc-swagger.json

12.5. Building projects with the Build Command



Building projects with the Build Command

Figure 12-33. Building projects with the Build Command

Building RuleApps with Build Command

- Use Build Command to build rule and Java projects outside of Eclipse
 - Build Command Line
 - Build Command Maven plugin
- 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`

Managing deployment

© Copyright IBM Corporation 2020

Figure 12-34. 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 Command Line

- Define a configuration file

- Example:

```
project = ../HelloWorld/Hello Main Service/
output = ../HelloWorld/output
dep = simple dep
xom-classpath = XOM jars/hello-xom-1.0.0.jar
```

- Pass the configuration file for the projects to build to the Build Command Line

```
java -jar Build_Command_Line_executable_archive -config configuration_file
```

- *Build_Command_Line_executable_archive*: Path to the executable archive in your installation files
 - *configuration_file*: Path to your configuration file

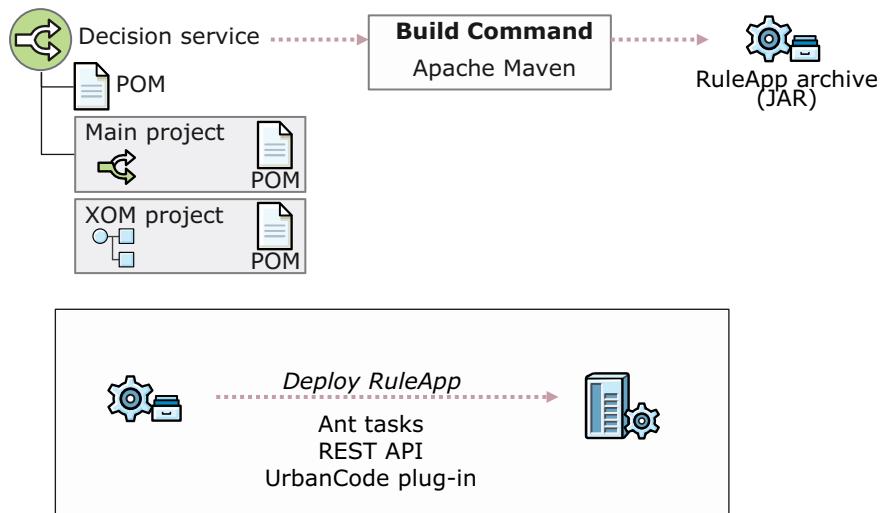
- Builds a RuleApp as a standalone archive in the specified output folder

Figure 12-35. Build Command Line

Build Command Maven plugin

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



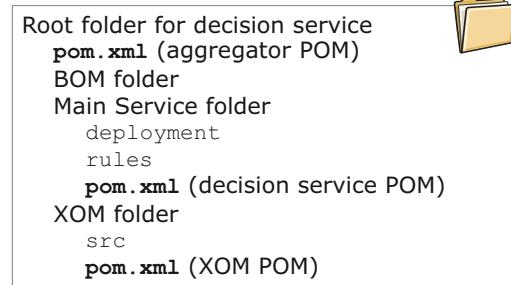
Managing deployment

© Copyright IBM Corporation 2020

Figure 12-36. Build Command Maven plugin

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 2020

Figure 12-37. 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_8.9.0/com.ibm.odm.dserver.rules.designer.rn/build_topics/con_buildcmd_setup_proj.html

Example decision service POM file (1 of 2)

Define a groupID value to identify the set of related decision service projects

Define artifactID to identify the decision service

```

<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 2020

Figure 12-38. 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)

The <resolvers> section identifies the XOM

Identify the XOM project that is referenced by your decision service in the <dependencies> section

```

...
<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>

```

Managing deployment

© Copyright IBM Corporation 2020

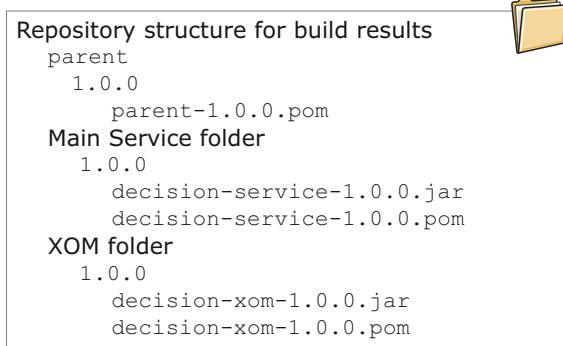
Figure 12-39. Example decision service POM file (2 of 2)

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



Managing deployment

© Copyright IBM Corporation 2020

Figure 12-40. 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

© Copyright IBM Corporation 2020

Figure 12-41. 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. True or False: You can use the Build Command Maven plug-in to package RuleApps for deployment.

Managing deployment

© Copyright IBM Corporation 2020

Figure 12-42. Review questions

Write your answers here:

- 1.
- 2.
- 3.

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. 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 2020

Figure 12-43. Review answers

Exercise: Managing deployment



Figure 12-44. Exercise: Managing deployment

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
- Build and deploy rulesets in the Decision Center API console

© Copyright IBM Corporation 2020

Figure 12-45. Exercise introduction

Unit 13. Executing rules with Rule Execution Server

Estimated time

02:00

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

© Copyright IBM Corporation 2020

Figure 13-1. Unit objectives

Topics

- Introducing managed execution
- Managed execution with Rule Execution Server
- Rule Execution Server modular architecture
- Managed execution in action
- Platforms for Rule Execution Server
- Rule Execution Server API
- Executing rules in Java SE
- Executing rules in Java EE
- Executing rules as transparent decision services

© Copyright IBM Corporation 2020

Figure 13-2. Topics

13.1. Introducing managed execution



Introducing managed execution

Figure 13-3. 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

Executing rules with Rule Execution Server

© Copyright IBM Corporation 2020

Figure 13-4. 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

Executing rules with Rule Execution Server

© Copyright IBM Corporation 2020

Figure 13-5. Key functions of Rule Execution Server

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.

Setting up a Rule Execution Server

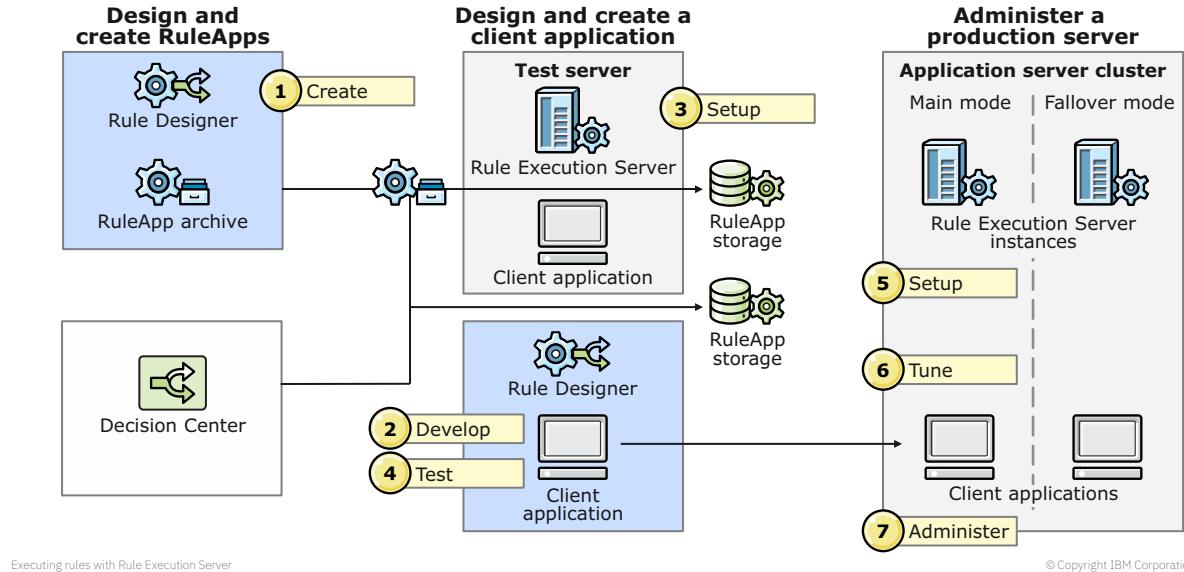


Figure 13-6. Setting up a Rule Execution Server

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 decision service by packaging the ruleset as a RuleApps. 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.

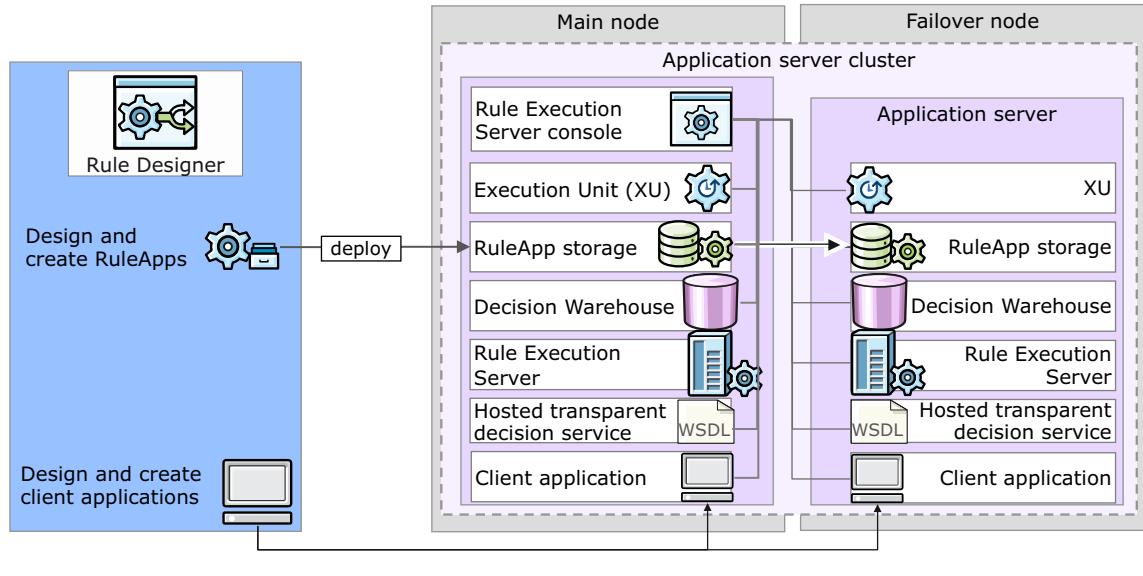
13.2. Managed execution with Rule Execution Server



Managed execution with Rule Execution Server

Figure 13-7. Managed execution with Rule Execution Server

Rule Execution Server in a production enterprise application



Executing rules with Rule Execution Server

© Copyright IBM Corporation 2020

Figure 13-8. 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 13-9. 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

Executing rules with Rule Execution Server

© Copyright IBM Corporation 2020

Figure 13-10. 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.

13.3. Rule Execution Server modular architecture

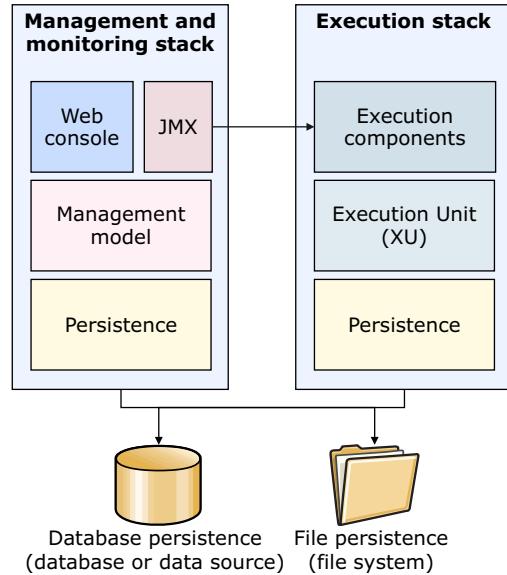


Rule Execution Server modular architecture

Figure 13-11. 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 2020

Figure 13-12. 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:

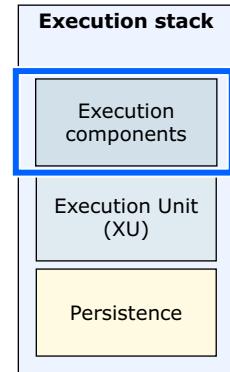
- Package and deploy Rule Execution Server and user components on the application server.
Although how to do this deployment is not covered in this course, you can find some information about this topic later in this unit.
- Manage enterprise applications and control access to:
 - Enterprise data stores
 - 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)



Executing rules with Rule Execution Server

© Copyright IBM Corporation 2020

Figure 13-13. 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)

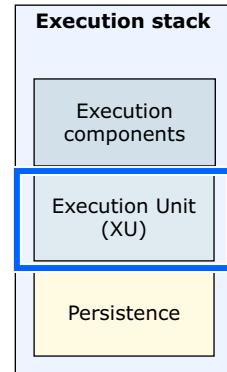


Information

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 (XU)

- 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 2020

Figure 13-14. Execution Unit (XU)

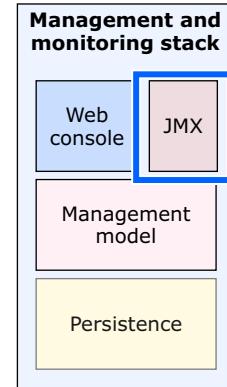
In Rule Execution Server, an Execution Unit (XU) manages rule engines. The XU is a resource adapter for Java EE Connector Architecture (JCA), which handles the low-level details of ruleset execution and provides access to manage its resources. You can also access configuration and runtime data either 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



Executing rules with Rule Execution Server

© Copyright IBM Corporation 2020

Figure 13-15. JMX management and execution model

The management and monitoring stack includes:

- 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:

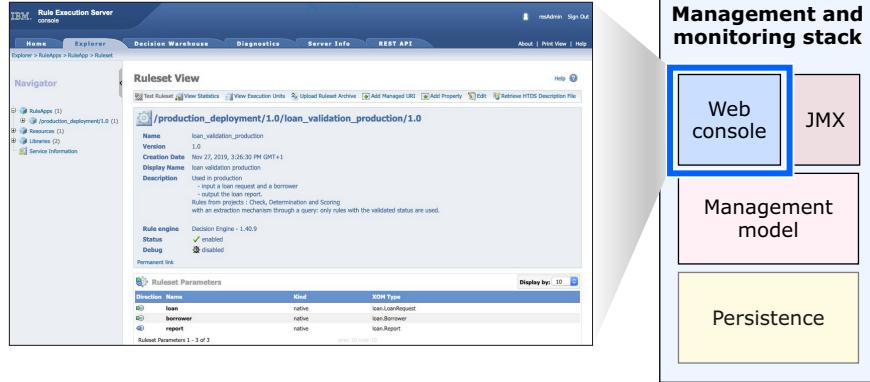
- Hot update and deployment. When a new version of a ruleset is deployed, the JMX layer informs all XUs in the cluster that a new version of the ruleset is available in the database.
- Displaying execution statistics as MBeans Execution statistics so they can then be monitored by using JMX tools like 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 decision service artifacts



Executing rules with Rule Execution Server

© Copyright IBM Corporation 2020

Figure 13-16. Management and monitoring stack: Console

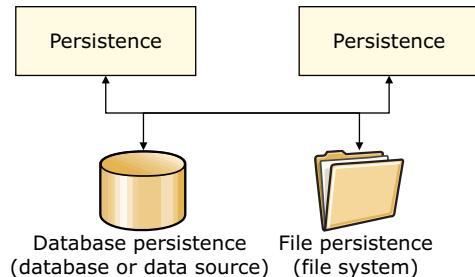
The Rule Execution Server console is a graphical user interface that you use to access and manage Rule Execution Server artifacts.

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 2020

Figure 13-17. 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)

Executing rules with Rule Execution Server

© Copyright IBM Corporation 2020

Figure 13-18. 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.

13.4. Managed execution in action



Managed execution in action

Figure 13-19. Managed execution in action

Elements of a rule-based solution

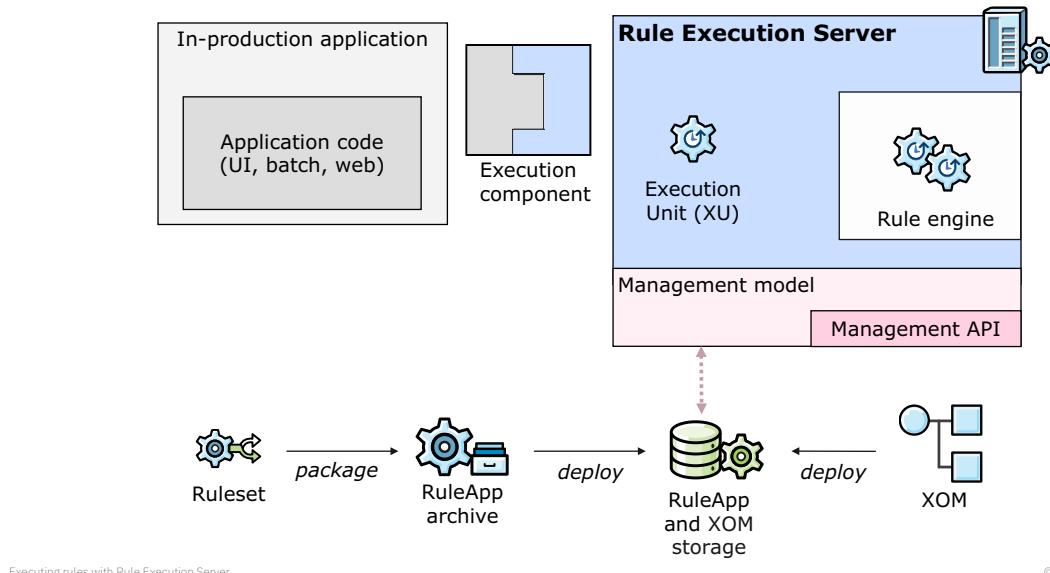


Figure 13-20. Elements of a rule-based solution

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 2020

Figure 13-21. 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

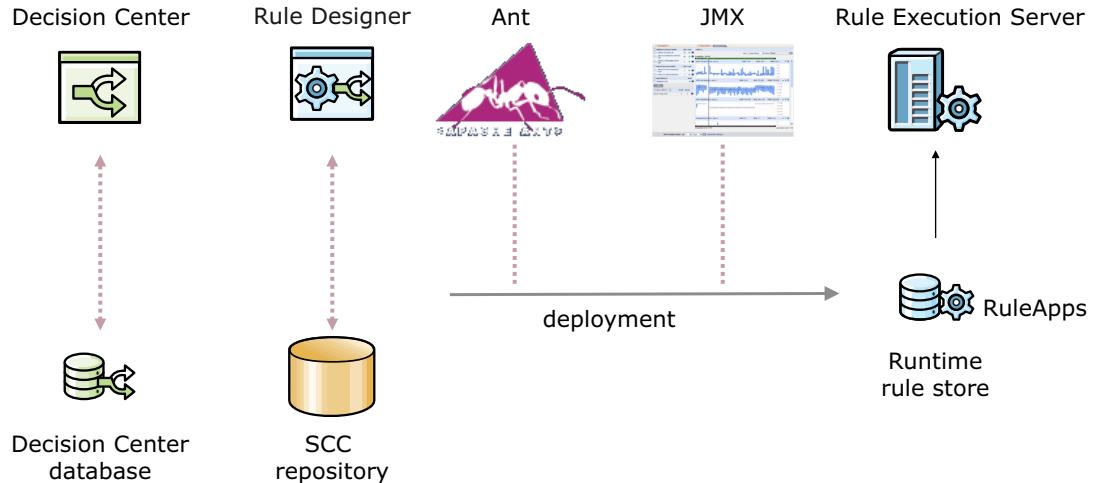
- The XU retrieves a rule engine from the pool of available rule engines, and attaches it to the rule session
- The XU feeds the rule engine with the loaded rulesets, ruleset parameters, and objects in working memory
- The XU then requests the ruleset execution by the rule engine
- After ruleset execution, 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

Executing rules with Rule Execution Server

© Copyright IBM Corporation 2020

Figure 13-22. Ruleset execution

Ruleset update at run time



Executing rules with Rule Execution Server

© Copyright IBM Corporation 2020

Figure 13-23. 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 13-24. 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.

13.5. Platforms for Rule Execution Server



Platforms for Rule Execution Server

Figure 13-25. 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

Executing rules with Rule Execution Server

© Copyright IBM Corporation 2020

Figure 13-26. 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"> • Base your application on Java • Use Rule Execution Server Java SE rule session API
Deployed in a web server or an application server	SOA: Rules are seen as services	(any)	<ul style="list-style-type: none"> • Request execution of business rules as a service • Use hosted transparent decision service or monitored transparent decision service
	Not SOA: Rules are not seen as services	Supported application servers	<ul style="list-style-type: none"> • Use Rule Execution Server Java EE API • Can be synchronous or not, local or remote, with or without EJB
		Non-supported application servers, or web servers	<ul style="list-style-type: none"> • Use Rule Execution Server Java SE API

Executing rules with Rule Execution Server

© Copyright IBM Corporation 2020

Figure 13-27. 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
 - 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

Executing rules with Rule Execution Server

© Copyright IBM Corporation 2020

Figure 13-28. 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)

Executing rules with Rule Execution Server

© Copyright IBM Corporation 2020

Figure 13-29. Java EE

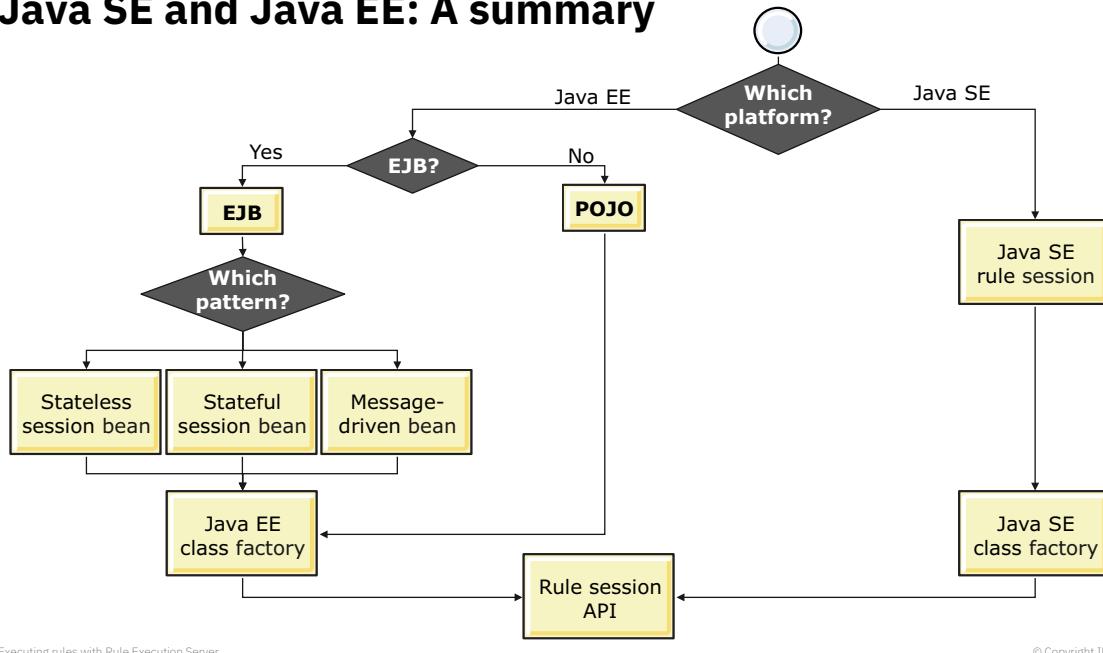
The Java EE platform is based on the Java SE specification. Java EE simplifies enterprise applications by defining and specifying a complete set of common standard services, such as naming, transaction management, concurrency, security, and database access. Java EE also defines a container model, which 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 2020

Figure 13-30. 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

- Ruleset exposed as a web service
- 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)
 - Web service with minimal configuration

Executing rules with Rule Execution Server

© Copyright IBM Corporation 2020

Figure 13-31. *Transparent decision services*

You can deploy rulesets as a web service that is called a hosted transparent decision service. The hosted transparent decision service is installed on the same application server as Rule Execution Server, then integrated with Rule Execution Server. 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.

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 HDTs EAR file.

If you use 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.

13.6. Rule Execution Server API



Rule Execution Server API

Figure 13-32. 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

Executing rules with Rule Execution Server

© Copyright IBM Corporation 2020

Figure 13-33. 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 the *execution patterns*, and 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

Executing rules with Rule Execution Server

© Copyright IBM Corporation 2020

Figure 13-34. 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)

Executing rules with Rule Execution Server

© Copyright IBM Corporation 2020

Figure 13-35. 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

- When using rule sessions to request ruleset execution within the managed environment of Rule Execution Server, your client application completes these steps:
 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
- 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

Executing rules with Rule Execution Server

© Copyright IBM Corporation 2020

Figure 13-36. Rule session API overview

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

Executing rules with Rule Execution Server

© Copyright IBM Corporation 2020

Figure 13-37. 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

Executing rules with Rule Execution Server

© Copyright IBM Corporation 2020

Figure 13-38. 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

Executing rules with Rule Execution Server

© Copyright IBM Corporation 2020

Figure 13-39. 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

Executing rules with Rule Execution Server

© Copyright IBM Corporation 2020

Figure 13-40. 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

Executing rules with Rule Execution Server

© Copyright IBM Corporation 2020

Figure 13-41. 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 13-42. 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

Executing rules with Rule Execution Server

© Copyright IBM Corporation 2020

Figure 13-43. 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

Executing rules with Rule Execution Server

© Copyright IBM Corporation 2020

Figure 13-44. 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

Executing rules with Rule Execution Server

© Copyright IBM Corporation 2020

Figure 13-45. 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

Executing rules with Rule Execution Server

© Copyright IBM Corporation 2020

Figure 13-46. 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

Executing rules with Rule Execution Server

© Copyright IBM Corporation 2020

Figure 13-47. 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 `IlrExecutionTrace` 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`

Executing rules with Rule Execution Server

© Copyright IBM Corporation 2020

Figure 13-48. 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.

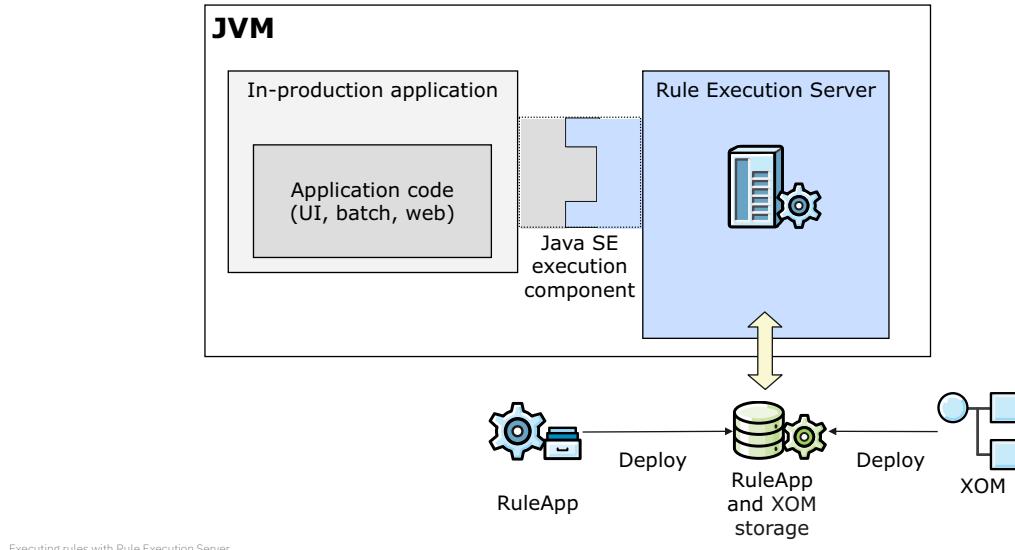
13.7. Executing rules in Java SE



Executing rules in Java SE

Figure 13-49. Executing rules in Java SE

Rule execution in Java SE



Executing rules with Rule Execution Server

© Copyright IBM Corporation 2020

Figure 13-50. 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

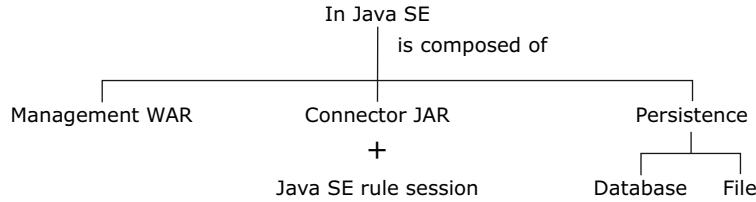


Figure 13-51. Deployed Rule Execution Server artifacts in Java SE

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

Executing rules with Rule Execution Server

© Copyright IBM Corporation 2020

Figure 13-52. 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");
```

Executing rules with Rule Execution Server

© Copyright IBM Corporation 2020

Figure 13-53. 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();
```

- 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);
```

- 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");
```

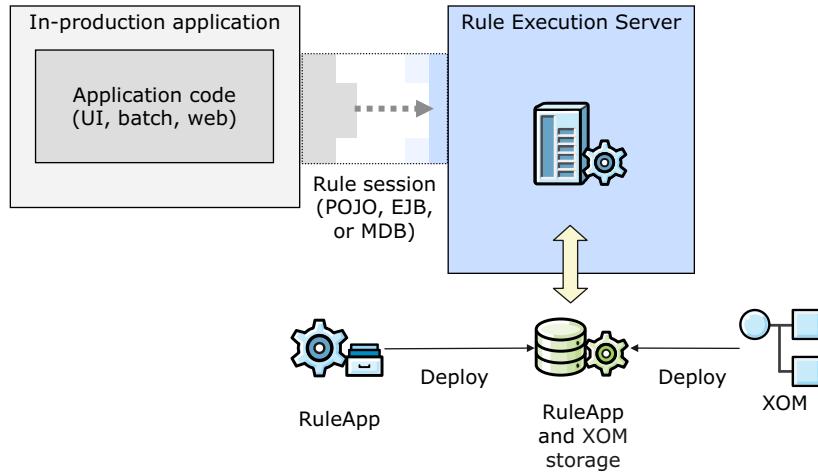
13.8. Executing rules in Java EE



Executing rules in Java EE

Figure 13-54. Executing rules in Java EE

Rule execution in Java EE



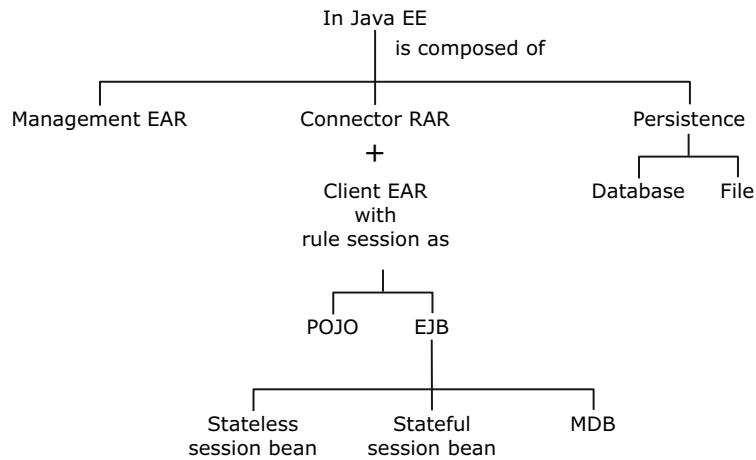
Executing rules with Rule Execution Server

© Copyright IBM Corporation 2020

Figure 13-55. 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 2020

Figure 13-56. 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

Executing rules with Rule Execution Server

© Copyright IBM Corporation 2020

Figure 13-57. 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`)

Executing rules with Rule Execution Server

© Copyright IBM Corporation 2020

Figure 13-58. 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`)

Executing rules with Rule Execution Server

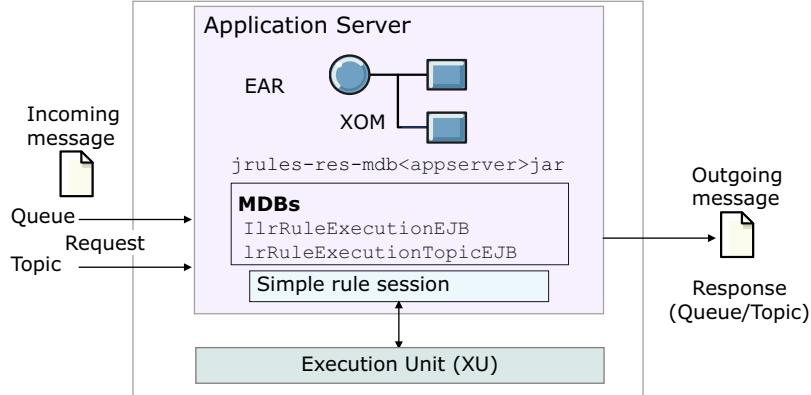
© Copyright IBM Corporation 2020

Figure 13-59. 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 2020

Figure 13-60. 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.

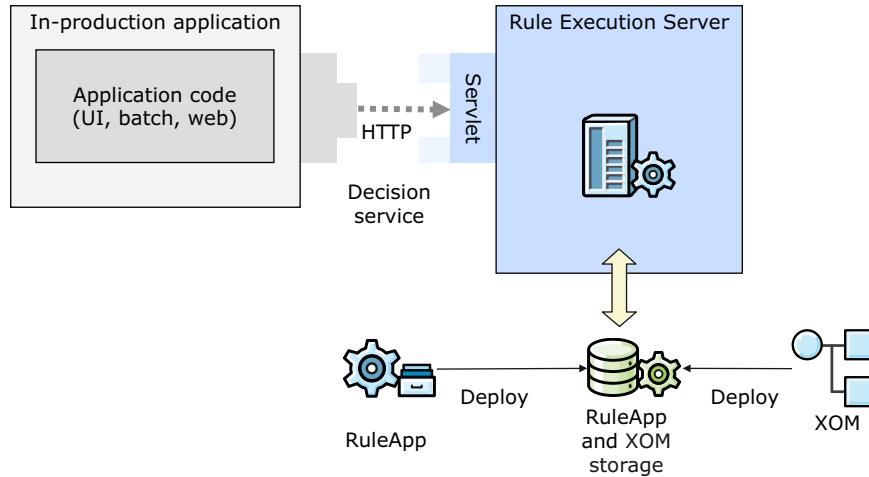
13.9. Executing rules as transparent decision services



Executing rules as transparent decision services

Figure 13-61. Executing rules as transparent decision services

Rules as transparent decision services



Executing rules with Rule Execution Server

© Copyright IBM Corporation 2020

Figure 13-62. Rules as transparent decision services

A hosted transparent decision service is essentially a ruleset that is deployed as a web service. It 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 exposes any deployed rulesets as a web service regardless of the underlying type of XOM (Java or dynamic). These rulesets can be exposed as a web service without any code deployment.

To execute rules as decision services, the client application sends execution requests to Rule Execution Server through HTTP.

You learn more about transparent decision services in the next unit.

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

© Copyright IBM Corporation 2020

Figure 13-63. 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

Executing rules with Rule Execution Server

© Copyright IBM Corporation 2020

Figure 13-64. 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 instances
[The answer is A, B, C, and D.](#)

Executing rules with Rule Execution Server

© Copyright IBM Corporation 2020

Figure 13-65. Review answers

Exercise: Exploring the Rule Execution Server console

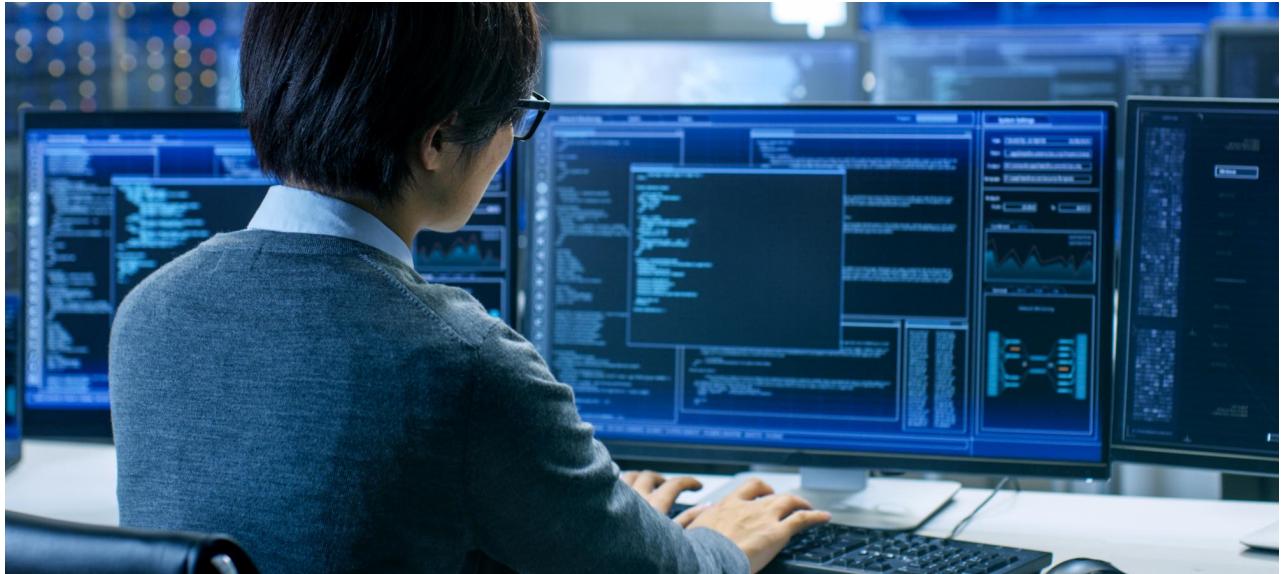


Figure 13-66. Exercise: Exploring the Rule Execution Server console

Exercise introduction

- Work with Rule Execution Server console tools
- Manage RuleApps and rulesets through the Rule Execution Server console

© Copyright IBM Corporation 2020

Figure 13-67. Exercise introduction

Unit 14. Working with transparent decision services

Estimated time

00:45

Overview

This unit teaches you how to work with transparent decision services and use the REST service for ruleset execution.

How you will check your progress

- Review
- Exercise

Unit objectives

- Describe the options for using transparent decision services
- Describe the REST service for ruleset execution
- Expose a decision service as an API

© Copyright IBM Corporation 2021

Figure 14-1. Unit objectives

Topics

- Calling decision services
- Using the REST service for ruleset execution
- Testing ruleset execution with the REST service

© Copyright IBM Corporation 2021

Figure 14-2. Topics

14.1. Calling decision services

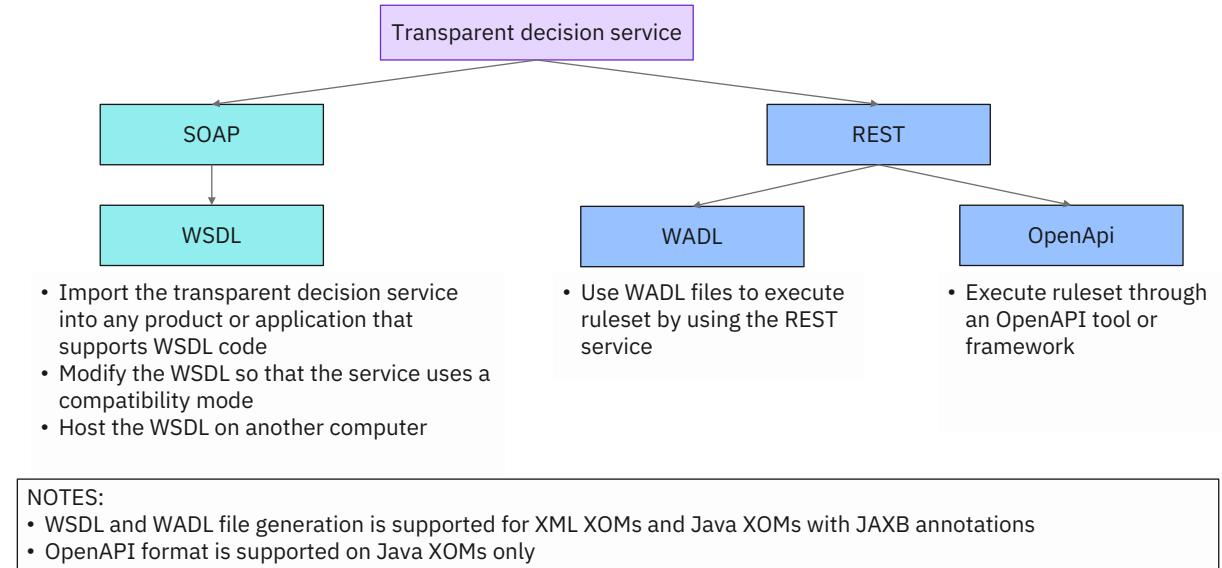


Calling decision services

© Copyright IBM Corporation 2021

Figure 14-3. Calling decision services

Working with HTDS description files



Working with transparent decision services

© Copyright IBM Corporation 2021

Figure 14-4. Working with HTDS description files

You can call decision services as a SOAP web service or by using the REST API.

When you present a ruleset as a SOAP hosted transparent decision service, the description file is generated in WSDL. When you create a transparent decision service to execute a ruleset through the REST API, the description file is generated in WADL or OpenAPI.

You can view or download the description file of a hosted transparent decision service to Web Service Description Language (WSDL), Web Application Description Language (WADL), or OpenAPI format. You can also generate files for WADL and OpenAPI to test ruleset execution in Rule Execution Server console.

Calling a decision service as a SOAP web service

- To invoke a deployed decision service as a SOAP web service
 1. Retrieve the WSDL file for the decision service
 2. Generate the proxy classes from the downloaded WSDL file
 3. Use the proxy classes to invoke the decision service in your client application

Working with transparent decision services

© Copyright IBM Corporation 2021

Figure 14-5. Calling a decision service as a SOAP web service

To call a decision service as a SOAP web service, you need to generate proxy classes from the WDSL file. You then use the proxy classes in your client application to invoke the decision service.

Calling a decision service by using REST API

- Clients use standard HTTP GET, POST, PUT, and DELETE commands to communicate with remote server
- After creating a secure connection with the server, use REST API to call the decision service

Working with transparent decision services

© Copyright IBM Corporation 2021

Figure 14-6. Calling a decision service by using REST API

To call a decision service as a SOAP web service, you

The client calls the decision service by using the standard HTTP GET, POST, PUT, and DELETE commands that web browsers use when they interact with remote servers

14.2. Using the REST service for ruleset execution



Using the REST service for ruleset execution

© Copyright IBM Corporation 2021

Figure 14-7. Using the REST service for ruleset execution

REST service for ruleset execution

- Decision Server provides a Representational State Transfer (REST) service for ruleset execution
 - Provides XML and JSON generation, XSD validation, and execution services
 - Execute rulesets with XML or JSON format through the HTTP protocol
- 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

Working with transparent decision services

© Copyright IBM Corporation 2021

Figure 14-8. REST service for ruleset execution

You can use the REST API to execute rulesets through the HTTP protocol by using the XML format.

The REST service includes:

- XML and JSON payload sample generation
- 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. You can work across environments or from other client applications, such as JavaScript clients. And you can easily switch from local execution to a remote Rule Execution Server instance.

Endpoint URIs

- Primary REST resource is the ruleset
 - Each resource is represented as a URI
 - Each resource has an HTTP method to request execution and receive a response

- URI format:

`http://{host}:{port}/DecisionService/rest/v1/{rulesetpath}/{filetype}?{options}`

URI item	Description
host:port/DecisionService	Host address and port for the HTDS application
/rest/v1	Context root and version of the REST service
rulesetpath	Short ruleset paths with one or no version numbers Canonical ruleset paths
filetype	wadl or openapi
options	Options for generating the file

Working with transparent decision services

© Copyright IBM Corporation 2021

Figure 14-9. Endpoint URIs

The primary REST resource is the ruleset, which are represented as endpoint URIs.

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 rulesetpath, 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.

The filetype can be WADL or OpenAPI.

You can also include various options, depending on the filetype.

HTTP methods and content types

- GET method
 - Generate a sample XML or JSON payload
 - Retrieve the WADL or OpenAPI for a specified ruleset
- POST method
 - Create an execution request
 - Validate an XML payload
- HTTP status codes
 - Returned to notify the calling client of the failure or success of your requests for WSDL, WADL, or OpenAPI format
- For more information about WADL representations, see

Working with transparent decision services

© Copyright IBM Corporation 2021

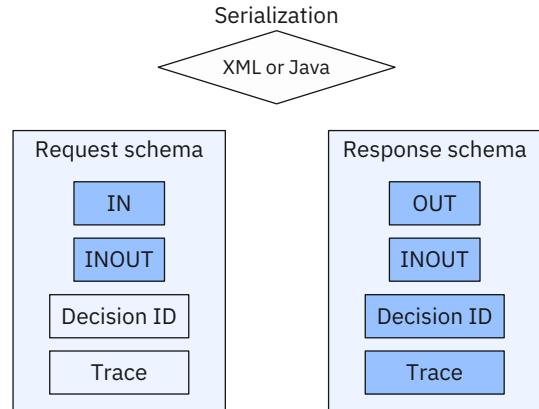
Figure 14-10. HTTP methods and content types

The REST API supports HTTP methods. You use the GET method to generate the sample payload or retrieve the WADL representation for the ruleset.

You use the POST method to execute the ruleset. If you use XML, you can also validate that the XML is well-formatted by posting a “validate” request.

Request and response schema

- Request contains:
 - The ruleset IN and INOUT parameters in alphabetical order
 - A decision ID (optional)
 - A trace filter (optional)
- Execution responses are returned in the same format as request (XML or JSON)
 - The ruleset INOUT and OUT parameters, in alphabetical order
 - The decision ID
 - The trace, if specified in the request



Working with transparent decision services

© Copyright IBM Corporation 2021

Figure 14-11. Request and response schema

Execution 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. In the REST service for ruleset execution, ruleset parameters are handled differently depending on whether they are expressed as XML elements or as Java types. The request and response schema results from the signature of the target ruleset.

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.

For more information about XML serialization of ruleset XOMs, see the IBM Knowledge Center:
https://www.ibm.com/support/knowledgecenter/SSQP76_8.10.x/com.ibm.odm.dserver.rules.ref.res/topics/con_resref_xml_serialization.html

WADL representation

- You generate WADL representation to write the XML or JSON 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

Working with transparent decision services

© Copyright IBM Corporation 2021

Figure 14-12. 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.

Exposing decision services as APIs

- OpenAPI is a standardized version of the Swagger specification
 - Language-independent way to present REST APIs
 - Allows API consumers to easily understand and use APIs
- OpenAPI based decisions
 - Can be invoked directly by applications
 - Can be published to an IBM API Connect catalog for managed APIs
- Use OpenAPI definitions to prototype and invoke ODM decisions
 - Generate the OpenAPI representation directly from Rule Execution Server console
 - Format of the OpenAPI definition file is either YAML or JSON
- Decision services that are exposed as web services must be transformed from web service interface to Swagger API

Working with transparent decision services

© Copyright IBM Corporation 2021

Figure 14-13. Exposing decision services as APIs

Decisions can be exposed as OpenAPI based public APIs to be invoked directly by applications. The OpenAPI based decisions can also be published and executed through an OpenAPI tool or framework, such as IBM API Connect.

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 web service.

Operational Decision Manager supports V3.0.1 of the Swagger specification. For more information about using OpenAPI, see:

www.ibm.com/support/knowledgecenter/en/SSQP76_8.10.x/com.ibm.odm.dserver.rules.res.developing/topics/tsk_res_restapi_rlset_exec_opnapi.html

For more information about API Connect, see: developer.ibm.com/apiconnect

14.3. Testing ruleset execution with the REST API



Testing ruleset execution with the REST API

© Copyright IBM Corporation 2021

Figure 14-14. Testing ruleset execution with the REST API

Steps for using the REST service for ruleset execution

1. Ensure that the ruleset is deployed and associated with a XOM
2. Generate a sample XML or JSON fragment as a starting point to write your request

Example:

```
GET http://host:port/DecisionService/rest/v1/rulesetPath/xml
```

3. For XML requests, validate the XML structure by posting a validate request

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 the same format as the request

Figure 14-15. Steps for using the REST service for ruleset execution

First, you make sure that the ruleset is deployed.

Then, you use the REST service to generate a sample XML or JSON payload. You use the sample fragment as a starting point to write the request.

For XML requests, you can post a “validate” request to ensure that it is well-formatted. You can also verify that XML or JSON requests are formatted correctly by testing in Rule Execution Server console, which you do during the exercise.

After validating the request, you can then send the request as the payload of an HTTP call by using POST method. If the request is not valid, error messages are returned in JSON format.

Retrieving the HTDS description (1 of 2)

- In the Ruleset View page for a deployed ruleset, retrieve the HTDS description file

The screenshot shows the 'Rule Execution Server console' interface. The top navigation bar includes links for Home, Explorer, Decision Warehouse, Diagnostics, Server Info, REST API, About, and Print View. The current page is 'Ruleset View'. On the left, a 'Navigator' pane shows a tree structure with 'RuleApps (2)', including '/loan_deploy/1.0 (2)' which further contains '/loan_ruleset/1.0' and '/loan_ruleset/1.1'. The main content area is titled 'Ruleset View' and displays a single entry: '/loan_deploy/1.0/loan_ruleset/1.1'. Below this entry, there is a table with two rows: 'Name' (loan_ruleset) and 'Version' (1.1). At the bottom of the content area, there is a row of buttons: Test Ruleset, View Statistics, View Execution Units, Upload Ruleset Archive, Add Managed URI, Add Property, and Edit. A red box highlights the 'Edit' button, which has a sub-option labeled 'Retrieve HTDS Description File'.

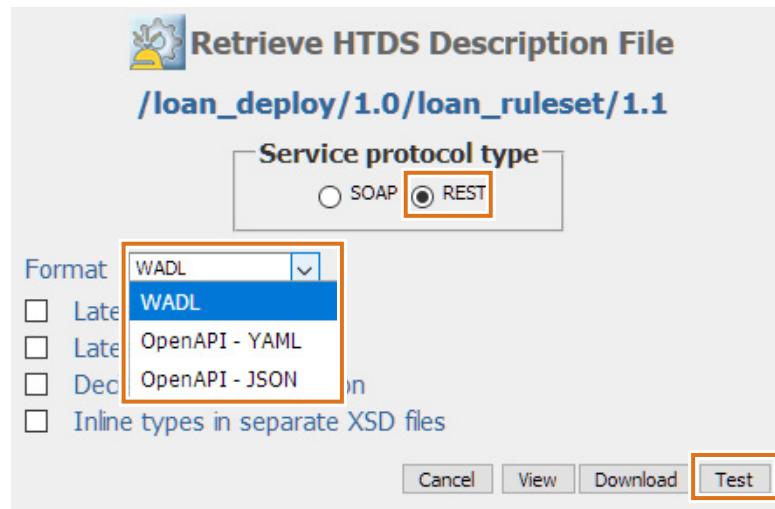
Working with transparent decision services

© Copyright IBM Corporation 2021

Figure 14-16. Retrieving the HTDS description (1 of 2)

Retrieving the HTDS description (2 of 2)

When you retrieve the HTDS description file for your ruleset, you select the REST option for testing



Working with transparent decision services

© Copyright IBM Corporation 2021

Figure 14-17. Retrieving the HTDS description (2 of 2)

Generating WADL execution request

WADL requests can be formatted in XML or JSON

Validate the XML before sending the request

The screenshot shows the IBM Rule Execution Server interface. At the top, it says "IBM. Rule Execution Server Hosted Transparent Decision Service". Below that, it says "Decision Service : /loan_deploy/1.0/loan_ruleset/1.1 REST Service". Underneath, there's a section titled "Execution Request" with three buttons: "XML" (highlighted with a yellow box), "JSON", and "HTML". Below the buttons is some XML code:

```

1 <par:Request xmlns:par="http://www.ibm.com/rules/decisionservice/Loan_deploy/Loan_ruleset/param">
2   <!--Optional:-->
3   <par:DecisionID>string</par:DecisionID>
4   <!--Optional:-->
5   <par:borrower>
6     <!--Optional:-->
7     <firstName>string</firstName>
8     <!--Optional:-->
9     <lastName>string</lastName>

```

Working with transparent decision services

© Copyright IBM Corporation 2021

Figure 14-18. Generating WADL execution request

You can choose to use XML or JSON to can create the execution request. For XML requests, you can validate the XML structure before you send the request.

Example: OpenAPI JSON request

Decision Service : /loan_deploy/1.0/loan_ruleset/1.1 REST Service

Execution Request JSON

```
1 {
2   "loan": {
3     "numberOfMonthlyPayments": 70,
4     "startDate": "2021-01-19",
5     "amount": 100000,
6     "loanToValue": 0.7
7   },
8   "borrower": {
9     "firstName": "Joe",
10    "lastName": "Doe",
11    "birth": "1987-09-28",
12    "SSN": {
13      "areaNumber": "424",
14      "groupCode": "56",
15      "serialNumber": "7942"
16    },
17    "yearlyIncome": 105000,
18    "zipCode": "95372",
19    "creditScore": 600,
20    "latestBankruptcy": {
21      "date": "2020-09-18",
22      "chapter": 3,
23      "reason": "Pandemic"
24    }
25 }
```

Cancel Execute Request

Working with transparent decision services

© Copyright IBM Corporation 2021

Figure 14-19. Example: OpenAPI JSON request

Here is an example of an OpenAPI JSON request.

Execution: OpenAPI JSON response

The execution response is returned in the same format as the request

The Server Response confirms that the request was well-formatted and can be used by your client application



```

        "validData": true,
        "corporateScore": 6140,
        "grade": "A",
        "insuranceRequired": true,
        "insuranceRate": 0.02
      },
      "approved": true,
      "messages": [
        "Very low risk loan",
        "Congratulations! Your loan has been approved"
      ],
      "insurance": "2%",
      "message": "Very low risk loan\nCongratulations! Your loan has been approved\n"
    },
    "DecisionID": "test",
    "loan": {
      "numberOfMonthlyPayments": 180,
      "startDate": "2021-01-19T00:00:00.000+0000",
      "amount": 100000,
      "loanToValue": 0.9,
      "yearlyInterestRate": 0.067,
      "monthlyRepayment": 882.1396772187496
    }
  }
}

```

Working with transparent decision services

© Copyright IBM Corporation 2021

Figure 14-20. Execution: OpenAPI JSON response

Monitoring transparent decision services

To monitor statistics on transparent decision services in Rule Execution Server console, you click **Service Information** in the Navigator pane of the Explore tab

The screenshot shows the IBM Rule Execution Server console interface. The top navigation bar includes Home, Explorer (selected), Decision Warehouse, Diagnostics, Server Info, and REST API. Below the navigation is a breadcrumb trail: Explorer > Transparent Decision Service Information > Transparent Decision Service > Statistics. On the left, a Navigator pane lists RuleApps (2), Resources (3), and Libraries (1). A section titled 'Service Information (2)' is expanded, showing two entries: HTDS/loan_deploy/1.0/loan_ruleset/1.1 and HTDS/mydeployment/1.0/Miniloan_ServiceRuleset/1.0. The 'HTDS/loan_deploy/1.0/loan_ruleset/1.1' entry is highlighted with an orange box. To the right, the 'Transparent Decision Service Statistics View' is displayed. It shows the executed canonical ruleset path as /loan_deploy/1.0/loan_ruleset/1.1. Below this, a table provides detailed statistics for the localhost server:

Server	Statistics
Metric	Value
Errors Count	0
Error Last Date	Not Available
Count	31
Total Time (ms)	2047
Average Time (ms)	66.032
Min. Time (ms)	15
Max. Time (ms)	438
Last Execution Time (ms)	31
First Execution Date	Nov 29, 2020, 8:36:53 PM GMT-5
Last Execution Date	Dec 8, 2020, 7:09:54 AM GMT-5

Working with transparent decision services

© Copyright IBM Corporation 2021

Figure 14-21. Monitoring transparent decision services

You can monitor transparent decision services in Rule Execution Server console. Transparent decision service statistics are not saved when the service is unregistered. If the path of the executed ruleset changes during the life of the MBean, the statistics are automatically erased.

Unit summary

- Describe the options for using transparent decision services
- Describe the REST service for ruleset execution
- Expose a decision service as an API

© Copyright IBM Corporation 2021

Figure 14-22. Unit summary

Review questions



1. True or False: To execute ruleset through an OpenAPI tool or framework, you generate a WSDL file.
2. True or False: You cannot use REST service for ruleset execution on a remote Rule Execution Server.
3. True or False: To verify that your execution requests are well-formatted, you can test execution through the Rule Execution Server console.

Working with transparent decision services

© Copyright IBM Corporation 2021

Figure 14-23. Review questions

Write your answers here:

- 1.
- 2.
- 3.

Review answers



1. True or False: To execute ruleset through an OpenAPI tool or framework, you generate a WSDL file.
[The answer is False. You generate an OpenAPI file.](#)
2. True or False: You cannot use REST service for ruleset execution on a remote Rule Execution Server.
[The answer is False. You can use REST service for local or remote Rule Execution Server execution.](#)
3. True or False: To verify that your execution requests are well-formatted, you can test execution through the Rule Execution Server console.
[The answer is True.](#)

Figure 14-24. Review answers

Exercise: Executing rules as a hosted transparent decision service (HTDS)



Figure 14-25. Exercise: Executing rules as a hosted transparent decision service (HTDS)

Exercise introduction

- Retrieve HTDS description files
- Test ruleset execution by using the REST service

© Copyright IBM Corporation 2021

Figure 14-26. Exercise introduction

Unit 15. 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

© Copyright IBM Corporation 2020

Figure 15-1. Unit objectives

Topics

- Auditing ruleset execution
- Monitoring ruleset execution

© Copyright IBM Corporation 2020

Figure 15-2. Topics

15.1. Auditing ruleset execution



Auditing ruleset execution

© Copyright IBM Corporation 2020

Figure 15-3. 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

Auditing and monitoring ruleset execution

© Copyright IBM Corporation 2020

Figure 15-4. 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 2020

Figure 15-5. Decision Warehouse

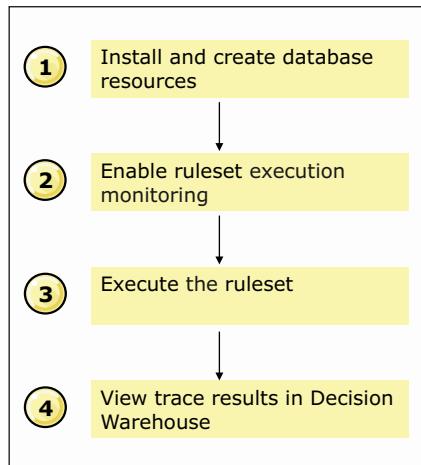
In Decision Warehouse, you can view stored decision traces.

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 2020

Figure 15-6. 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 2020

Figure 15-7. 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)

Auditing and monitoring ruleset execution

© Copyright IBM Corporation 2020

Figure 15-8. 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:
 - A client application
 - A hosted transparent decision service

Auditing and monitoring ruleset execution

© Copyright IBM Corporation 2020

Figure 15-9. 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**

Decision ID	Date	Ruleset Version	Number of rules fired	Decision Trace	Processing Time (ms)
5ab2332c-97ef-4787-b892-58166e39cebc	2009-03-26 18:02:58	/prodra1238086969350_90b385fe_4e4c_4842_b408_4d1e9dcadfd6	1	View Decision details	416
50a3601b-20fa-4f78-abf7-69356d4516c	2009-03-19 15:30:10	/prodra1237473006170_519ba66c_0ad6_4142_a3e9_3d7df42c7945/1.0	1	View Decision details	35
6c4f221-5d0e-486f-89d9-ceab48042d84	2009-03-19 15:30:10	/prodra1237473006170_519ba66c_0ad6_4142_a3e9_3d7df42c7945/1.0	0	View Decision details	37
a18b5329-ce90-4ea5-9a37-fb01e0be3e6	2009-03-19 15:30:08	/prodra1237473006170_519ba66c_0ad6_4142_a3e9_3d7df42c7945/1.0	1	View Decision details	205
385ee32d-6cb8-49b1-a648-d51c0cf07d	2009-03-19 15:27:48	/prodra1237472858960_f57dfa85_481b_4e0e_a50a_9bd94a70e73e/1.0	1	View Decision details	172
f10839340a28	2009-03-19 15:27:48	/prodra1237472858960_f57dfa85_481b_4e0e_a50a_9bd94a70e73e/1.0	0	View Decision details	47
41565ac8-2a56-47c4-b5ce-8d925cb89fa	2009-03-19 15:27:47	/prodra1237472858960_f57dfa85_481b_4e0e_a50a_9bd94a70e73e/1.0	1	View Decision details	505
57705e95-ff38-45b3-a716-e0ba45fb676	2009-03-13 11:57:58	/prodra1236941862005_eee830e0_cefa_4105_ab6e_7851d0bca552	4	View Decision details	1138
64efc105-58cd-4e71-8350-e550b957a610	2009-03-06 19:22:52	/prodra1236363766960_fad3c26c_ab69_46d9_9763_8d4ca0a42fc/1.0	3	View Decision details	18
f91fe910-2585-4d18-987a-dbc9e0b4f77	2009-03-06 19:22:52	/prodra1236363766960_fad3c26c_ab69_46d9_9763_8d4ca0a42fc/1.0	3	View Decision details	13

Auditing and monitoring ruleset execution

© Copyright IBM Corporation 2020

Figure 15-10. 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

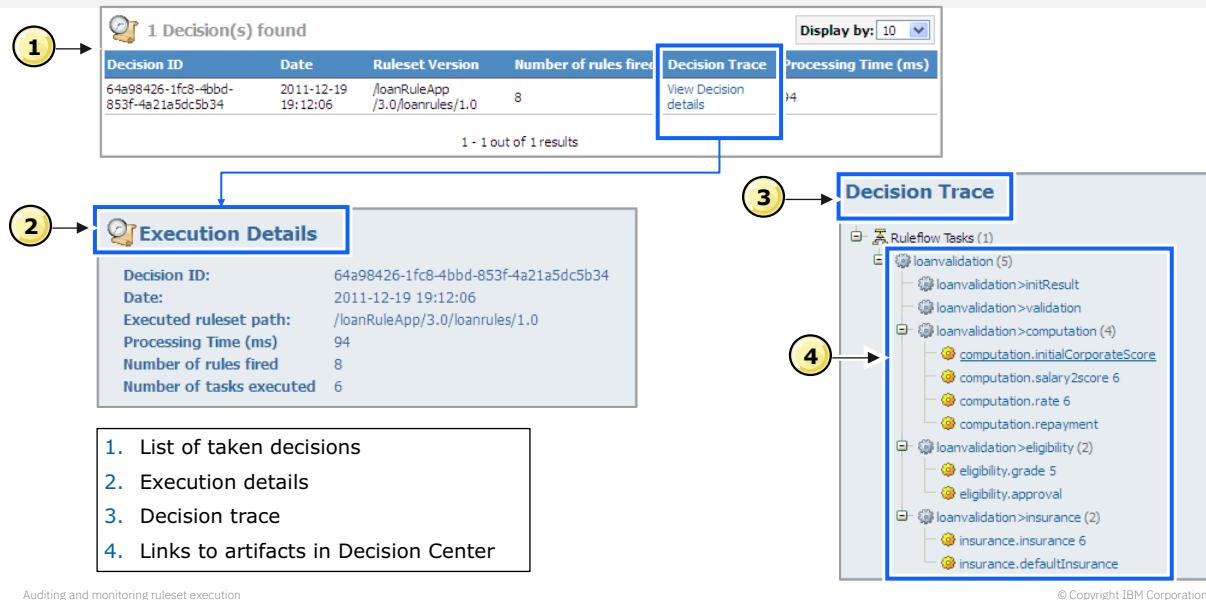
Auditing and monitoring ruleset execution

© Copyright IBM Corporation 2020

Figure 15-11. 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



Auditing and monitoring ruleset execution

© Copyright IBM Corporation 2020

Figure 15-12. 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.

[draft]If you deploy a RuleApp from Decision Center console, the **Decision Trace** section also contains a link to the rule artifacts stored in the Decision Center repository. When you click those links, it opens Business console so you can view the corresponding artifact.

15.2. Monitoring ruleset execution



Monitoring ruleset execution

© Copyright IBM Corporation 2020

Figure 15-13. 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 2020

Figure 15-14. 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 2020

Figure 15-15. 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

Auditing and monitoring ruleset execution

© Copyright IBM Corporation 2020

Figure 15-16. 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 figure consists of two side-by-side screenshots of a software interface for managing ruleset monitoring options. The left screenshot shows a list of '2 properties': 'monitoring.enabled' and 'rulesetSEQUENTIAL.trace.enabled'. Below this is a section for 'Upload properties from file' with a 'Choose file:' input field and a 'Browse...' button. Underneath is a 'Hide Monitoring Options' section with a 'monitoring options' icon and a checked checkbox for 'Enable tracing in the Decision Warehouse'. It also includes a radio button group for storing values: 'BOM format with optional filter:' (selected) and 'Native format'. The right screenshot shows a list of '3 properties': 'monitoring.enabled', 'ruleset.bom.enabled' (highlighted with a blue selection box), and 'rulesetSEQUENTIAL.trace.enabled'. It has the same 'Upload properties from file' and 'Hide Monitoring Options' sections. The 'ruleset.bom.enabled' row is highlighted with a blue selection box.

Auditing and monitoring ruleset execution

© Copyright IBM Corporation 2020

Figure 15-17. 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.

Ruleset monitoring options (3 of 5)

The screenshot shows a web-based configuration interface for monitoring ruleset execution. At the top, there is a table titled 'Hide Properties' with 4 properties listed:

Name	Value
monitoring.enabled	true
monitoring.inout.filters	borrower:some.heavy.property,loan.longfield
ruleset.bom.enabled	true
ruleset.sequential.trace.enabled	true

Below the table, there is a section for 'Upload properties from file' with a 'Choose file:' input field, a 'Browse...' button, and several action buttons: 'Proceed to update', 'Preview update', and 'Override existing'. A blue box highlights the 'monitoring.inout.filters' row.

Underneath this, there is a 'Hide Monitoring Options' section with a 'monitoring options' icon. It contains a checked checkbox for 'Enable tracing in the Decision Warehouse' and a dropdown menu for 'Store the values of the ruleset parameters to:' with two options: 'BOM format with optional filter: borrower:some.heavy.pi' (selected) and 'Native format'. A blue box highlights the 'BOM format with optional filter' option.

Auditing and monitoring ruleset execution

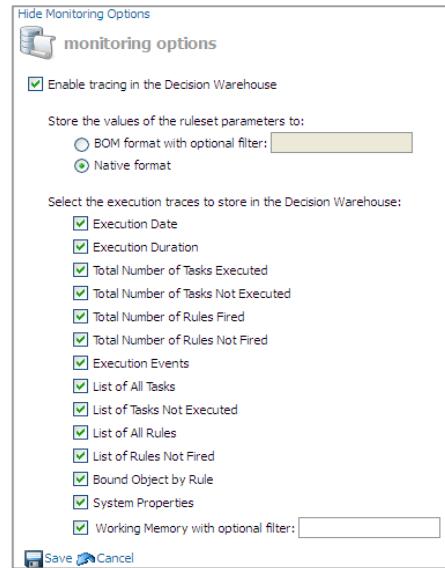
© Copyright IBM Corporation 2020

Figure 15-18. 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)

- 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 2020

Figure 15-19. Ruleset monitoring options (4 of 5)

On this slide, you see a list of some of the possible trace options.

Ruleset monitoring options (5 of 5)

Name	Value
monitoring.enabled	true
monitoring.filters	INFO_EXECUTION_DATE=true,INFO_EXECUTION_DURATION=true
monitoring.inout.filters	borrower.some.heavy.property,loan.longfield
ruleset.bom.enabled	true
rulesetSEQUENTIAL.trace.enabled	true

Upload properties from file
Choose file:

Hide Monitoring Options
monitoring options
✓ Tracing in the Decision Warehouse is currently enabled.
Project parameters will be stored as ROM 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 2020

Figure 15-20. 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 GMT +01:00	Not Available
		Last Execution Date	Dec 21, 2011 4:36:00 PM GMT +01:00	Not Available

Auditing and monitoring ruleset execution

© Copyright IBM Corporation 2020

Figure 15-21. 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

Auditing and monitoring ruleset execution

© Copyright IBM Corporation 2020

Figure 15-22. 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

Auditing and monitoring ruleset execution

© Copyright IBM Corporation 2020

Figure 15-23. 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

Auditing and monitoring ruleset execution

© Copyright IBM Corporation 2020

Figure 15-24. 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

Auditing and monitoring ruleset execution

© Copyright IBM Corporation 2020

Figure 15-25. 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

© Copyright IBM Corporation 2020

Figure 15-26. Unit summary

Review questions



1. True or False: To have decision traces, you must enable ruleset monitoring.
2. True or False: In the Rule Execution Server console, you can test any deployed ruleset.

Auditing and monitoring ruleset execution

© Copyright IBM Corporation 2020

Figure 15-27. Review questions

Review answers



- True or False: To have decision traces, you must enable ruleset monitoring.

The answer is **True**.

- True or False: In the Rule Execution Server console, you can test any deployed ruleset.

The answer is **False**. If a deployed ruleset is not associated with a managed XOM, you can't test it in the console.

Figure 15-28. Review answers

Exercise: Auditing ruleset execution through Decision Warehouse



Figure 15-29. 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

© Copyright IBM Corporation 2020

Figure 15-30. Exercise introduction

Unit 16. Applying decision governance

Estimated time

01:00

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

© Copyright IBM Corporation 2020

Figure 16-1. Unit objectives

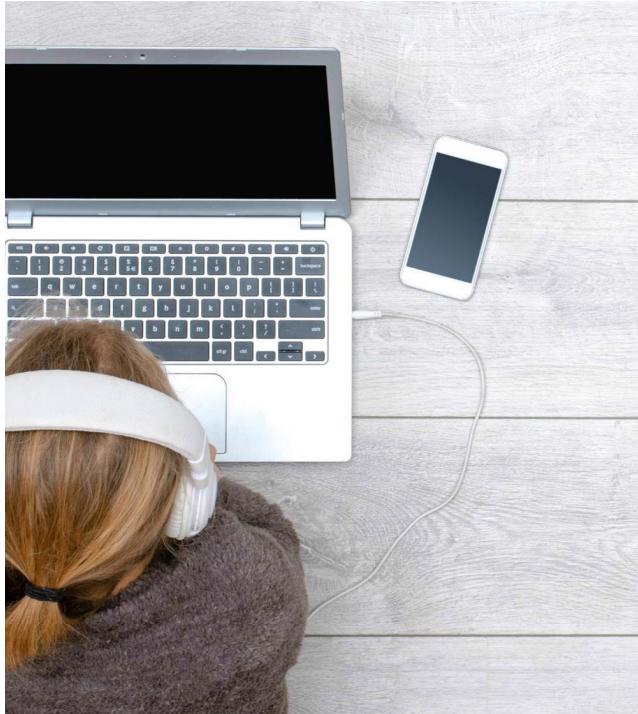
Topics

- What is decision governance?
- Operational Decision Manager support for governance
- Decision governance framework

© Copyright IBM Corporation 2020

Figure 16-2. Topics

16.1. What is decision governance?



What is decision governance?

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

Applying decision governance

© Copyright IBM Corporation 2020

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

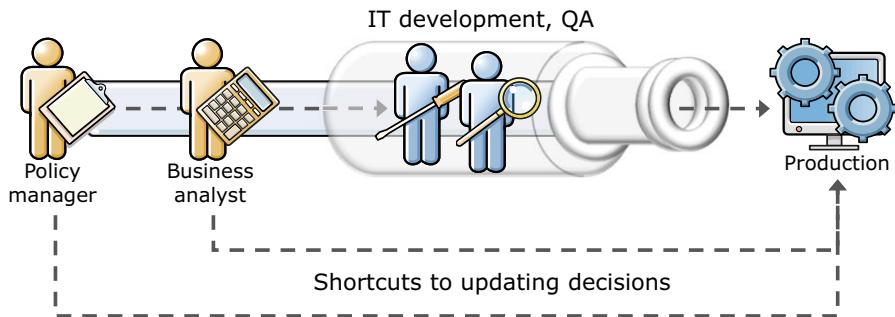
Applying decision governance

© Copyright IBM Corporation 2020

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



Applying decision governance

© Copyright IBM Corporation 2020

Figure 16-6. Traditional approach to maintenance

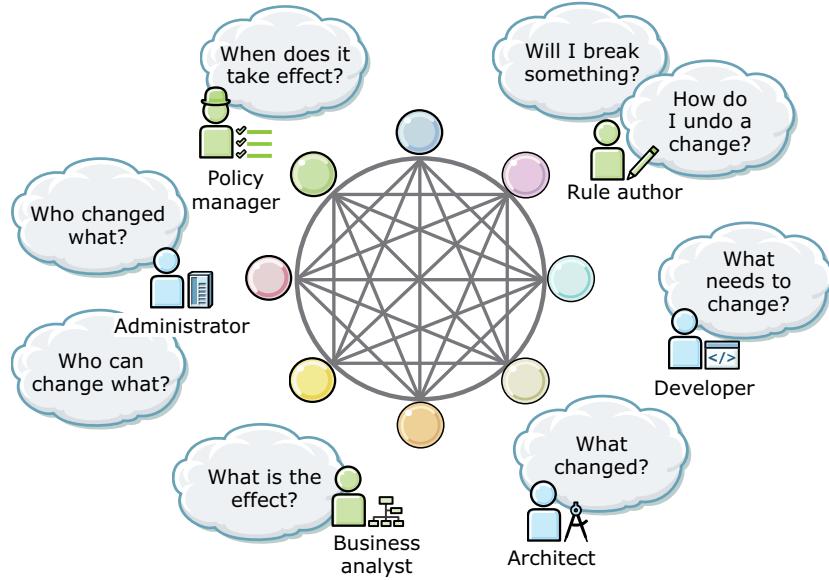
In the traditional approach, the IT development-QA pair can represent a bottleneck. Business policy updates, which are the new requirements that come from the business team, must go through that bottleneck.

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



Applying decision governance

© Copyright IBM Corporation 2020

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.

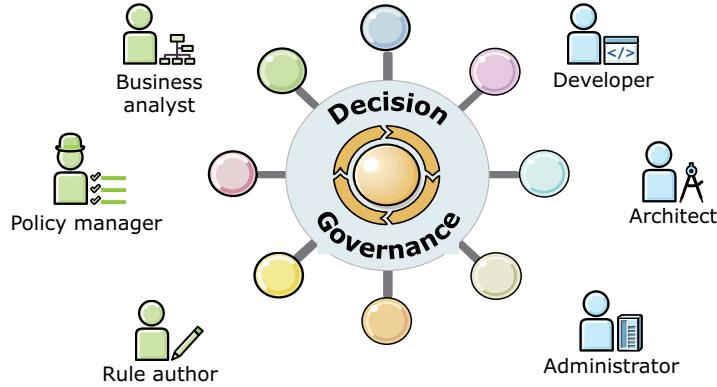
Examples:

- 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



Applying decision governance

© Copyright IBM Corporation 2020

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

Applying decision governance

© Copyright IBM Corporation 2020

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

Applying decision governance

© Copyright IBM Corporation 2020

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.

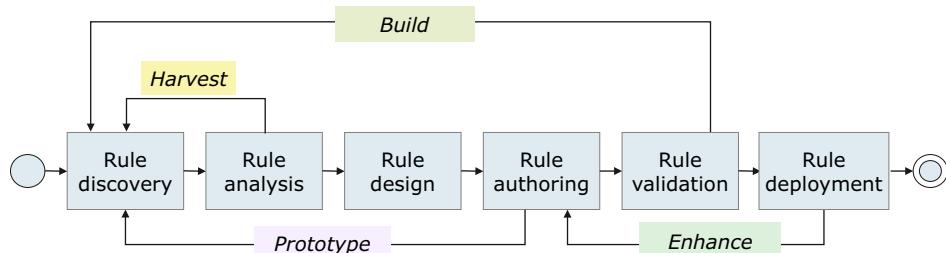
The group is formed from core team members who are involved in initial development of the application, including business analysts and policy managers, who can contribute to the governance objectives. Include members that represent various entities and facilitate communication between these entities 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



Applying decision governance

© Copyright IBM Corporation 2020

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

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



Applying decision governance

© Copyright IBM Corporation 2020

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.

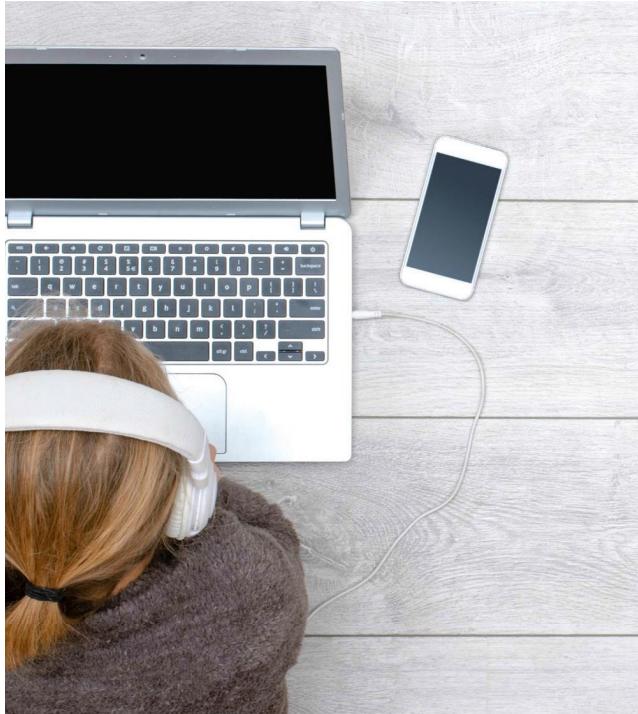
- Define each process with a Business Process Modeling Notation (BPMN) map

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. Decision governance framework



Decision governance framework

Figure 16-13. 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

Applying decision governance

© Copyright IBM Corporation 2020

Figure 16-14. Decision governance framework overview

The decision governance framework provides 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 in Decision Center Business console.

The decision governance framework is based on decision service *releases*, which follow a well-defined change process that is supported by Decision Center. Tasks that business users complete depend on user roles, such as rule author or tester.

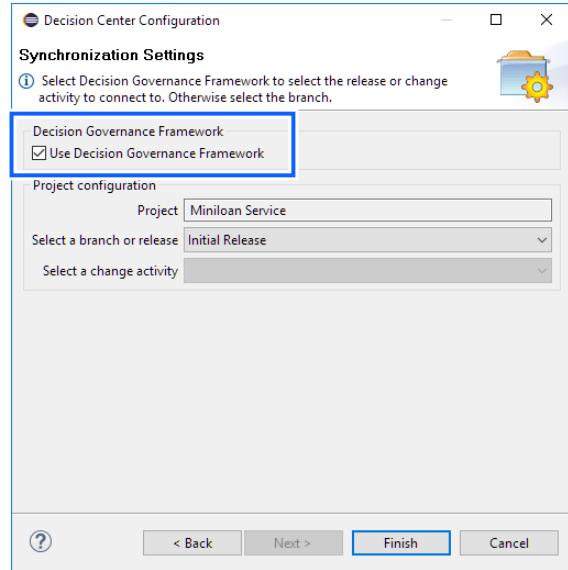
For effective workflow control, you set permissions for each phase to determine who can work with the rule during a particular phase, and who can promote an artifact from one phase to another.

Governance framework includes:

- Decision service releases
- Change activities
- Validation activities
- Deployment of releases

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**



Applying decision governance

© Copyright IBM Corporation 2020

Figure 16-15. 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
-
- WATCH: [Demonstration of using the decision governance framework in Business console](#)

Applying decision governance

© Copyright IBM Corporation 2020

Figure 16-16. Decision Center: Governance in Business console

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

Applying decision governance

© Copyright IBM Corporation 2020

Figure 16-17. 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**: A set of business-driven goals
 - A **period in time**: 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

Applying decision governance

© Copyright IBM Corporation 2020

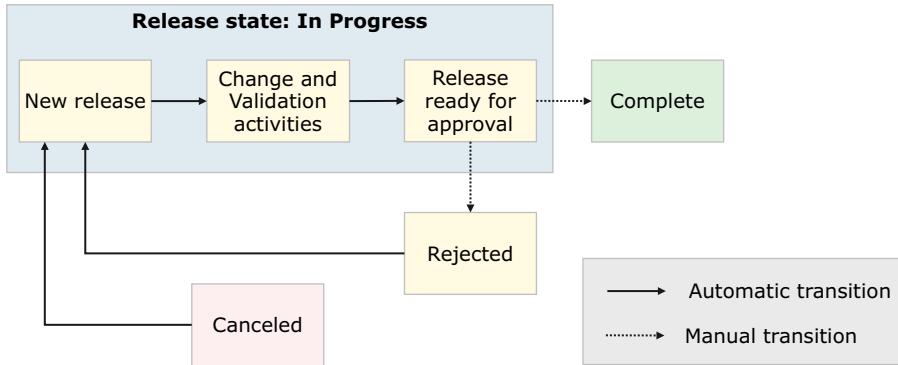
Figure 16-18. Releases

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



Applying decision governance

© Copyright IBM Corporation 2020

Figure 16-19. 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 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

Applying decision governance

© Copyright IBM Corporation 2020

Figure 16-20. 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.

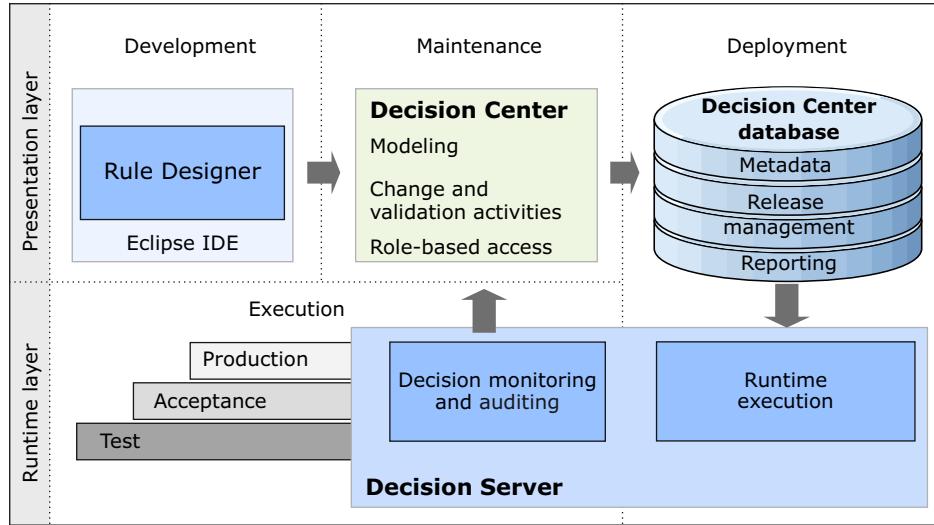
16.3. Operational Decision Manager support for governance



Operational Decision Manager support for governance

Figure 16-21. Operational Decision Manager support for governance

Decision lifecycle in Operational Decision Manager (1 of 2)



Applying decision governance

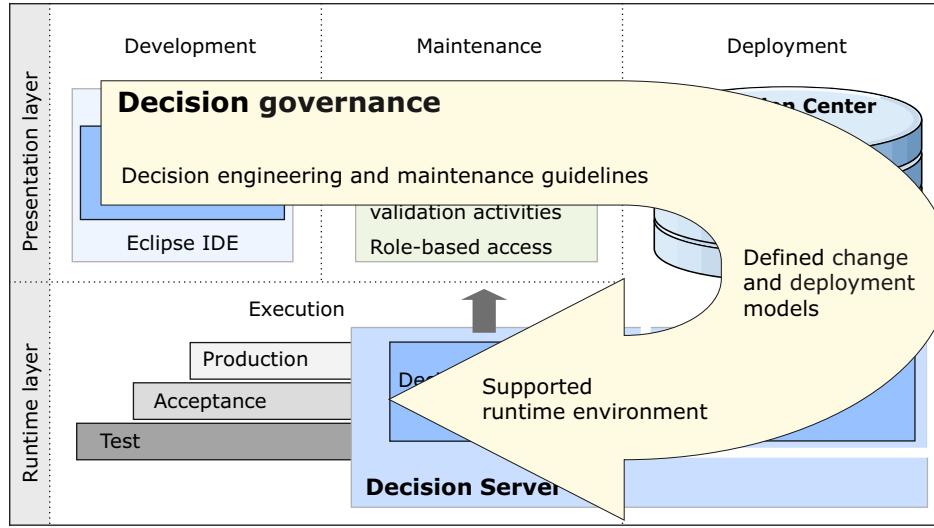
© Copyright IBM Corporation 2020

Figure 16-22. 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.
- In Decision Center, you can 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)



Applying decision governance

© Copyright IBM Corporation 2020

Figure 16-23. 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 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?



Figure 16-24. How can you apply governance?

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?

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

© Copyright IBM Corporation 2020

Figure 16-25. 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: When using the decision governance framework, business users cannot deploy decision services from Business console.

Applying decision governance

© Copyright IBM Corporation 2020

Figure 16-26. 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 16-27. 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: When using 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 16-28. 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
- Identify IBM credentials that are related to this course
- Locate resources for further study and skill development

© Copyright IBM Corporation 2021

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 ruleset execution
- Orchestrate rule execution through ruleflows
- Author rule artifacts to implement business policies

© Copyright IBM Corporation 2021

Figure 17-2. Course objectives

Course objectives

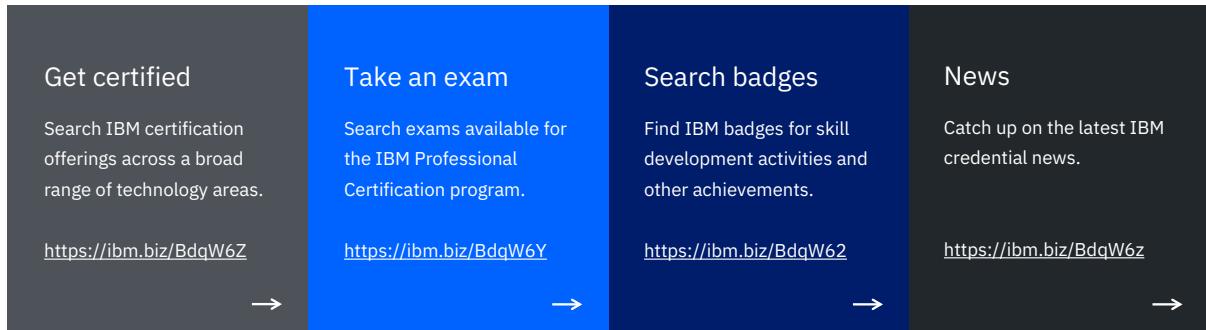
- Debug business rule applications to ensure that the implemented business logic is error-free
- Set up and customize testing and simulation for business users
- Package and deploy decision services to test and production environments
- Integrate decision services for managed execution within an enterprise environment
- Monitor and audit execution of decision services
- Apply governance principles to decision management

© Copyright IBM Corporation 2021

Figure 17-3. Course objectives

IBM credentials: Badges and certifications

- Certify your skills with IBM digital credentials
 - <https://www.ibm.com/training/credentials>



Course summary

© Copyright IBM Corporation 2021

Figure 17-4. IBM credentials: Badges and certifications

Learn more about this product

- **IBM Automation Community**

- Learn about Blockchain, Bluemix Live, BPM, Workflow, Case, Content Management, Decision Management, Robotic Process Automation, Platform and Cloud Pak for Automation
- <https://community.ibm.com/community/user/automation/home>



Course summary

© Copyright IBM Corporation 2021

Figure 17-5. Learn more about this product



Additional resources (1 of 5)

- **IBM Cloud Education course information**
 - View and download course materials and course corrections.
 - <http://ibm.biz/CourseInfo>
- **IBM Developer**
 - IBM's official developer program offers access to software trials and downloads, how-to information, and expert practitioners.
 - <https://developer.ibm.com/>

© Copyright IBM Corporation 2021

Figure 17-6. Additional resources (1 of 5)



Additional resources (2 of 5)

- **IBM Training**

- Search the IBM Training website for courses and education information.
- <https://www.ibm.com/training>

- **Learning Journeys**

- Learning Journeys describe a recommended collection of learning content to acquire skills for a specific technology or role.
- <https://www.ibm.com/training/journeys/#tab-ibm-cloud>

© Copyright IBM Corporation 2021

Figure 17-7. Additional resources (2 of 5)



Additional resources (3 of 5)

- **IBM Redbooks**

- IBM Redbooks are developed and published by the IBM International Technical Support Organization (ITSO). Redbooks typically provide positioning and value guidance, installation and implementation experiences, typical solution scenarios, and step-by-step "how-to" guidelines.

- <http://www.redbooks.ibm.com/>

- **IBM Knowledge Center**

- IBM Knowledge Center is the primary home for IBM product documentation.
- <https://www.ibm.com/support/knowledgecenter>

© Copyright IBM Corporation 2021

Figure 17-8. Additional resources (3 of 5)



Additional resources (4 of 5)

- **IBM Middleware User Community**
 - Learn about API Connect, App Connect, MQ, DataPower, Aspera, Event Streams, and Cloud Pak for Integration
 - <https://community.ibm.com/community/user/middleware/communities/cloud-integration-home>
- **IBM Marketplace**
 - Learn about IBM offerings for Cloud, Cognitive, Data and Analytics, Mobile, Security, IT Infrastructure, and Enterprise and Business Solutions.
 - <https://www.ibm.com/products>

© Copyright IBM Corporation 2021

Figure 17-9. Additional resources (4 of 5)



Additional resources (5 of 5)

- **IBM Training blog, Twitter, and Facebook**
 - Official IBM Training accounts provide information about IBM course offerings, industry information, conference events, and other education-related topics.
 - <https://www.ibm.com/blogs/ibm-training>
 - <https://twitter.com/IBMTTraining>
 - <https://www.facebook.com/ibmtraining>

© Copyright IBM Corporation 2021

Figure 17-10. Additional resources (5 of 5)

Unit summary

- Explain how the course met its learning objectives
- Identify IBM credentials that are related to this course
- Locate resources for further study and skill development

© Copyright IBM Corporation 2021

Figure 17-11. Unit summary

Course completion

You have completed this course:

Developing Rule Solutions in IBM Operational Decision Manager V8.10.5

Do you have any questions?



Course summary

© Copyright IBM Corporation 2021

Figure 17-12. Course completion

Appendix A. List of abbreviations

ABRD	Agile Business Rule Development
API	application programming interface
ATM	automatic teller machine
B2X	BOM to XOM mapping
BA	business analyst
BAL	Business Action Language
BEP	business event processing
BOM	business object model
BPM	business process management
BPMN	Business Process Modeling Notation
BQL	Business Query Language
BRM	business rule management
BRMS	business rule management system
CICS	Customer Information Control System
CPU	central processing unit
CRM	customer relationship management
CVS	Concurrent Versions System
Db	Database
DHCP	Dynamic Host Configuration Protocol
DVS	Decision Validation Services
DW	Decision Warehouse
EAR	enterprise archive
EE	Enterprise Edition (Java EE)
EJB	Enterprise JavaBeans
ERC	edition revision code
ESB	enterprise service bus
GRL	Guided Rule Language
GUI	graphical user interface
HTDS	hosted transparent decision service
HTTP	Hypertext Transfer Protocol

IBM	International Business Machines Corporation
ICP	IBM Cloud Private
IDE	integrated development environment
IP	Internet Protocol
IQL	Intellirule Query Language
IRL	ILOG Rule Language
IT	information technology
JAR	Java archive
Java EE	Java Platform, Enterprise Edition
Java SE	Java Platform, Standard Edition
JAXB	Java Architecture for XML Binding
JCA	Java EE Connector Architecture
JDBC	Java Database Connectivity
JDK	Java Development Kit
JMS	Java Message Service
JMX	Java Management Extension
JNDI	Java Naming and Directory Interface
JRE	Java Runtime Environment
JSON	JavaScript Object Notation
JVM	Java virtual machine
KPI	key performance indicator
LAN	local area network
LOB	line of business
MBean	management bean
MDB	message-driven bean
ODM	Operational Decision Manager
POJO	plain old Java object
POM	Project Object Model
POS	Point of sale
PVU	Processor Value Unit
QA	quality assurance
RAR	resource adapter archive
RES	Rule Execution Server
REST	Representational State Transfer

RFID	radio frequency identification
RMI	Remote Method Invocation
RQL	Rule Query Language
SCA	Service Component Architecture
SCC	source code control
SDK	software development kit
SDO	Service Data Object
SE	Standard Edition (Java SE)
SME	subject matter expert
SOA	service-oriented architecture
SOAP	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.
SPSS	Statistical Product and Service Solutions
SSN	Social Security Number
SSP	Scenario Service Provider
TCP	Transmission Control Protocol
TRL	Technical Rule Language
UI	user interface
UML	Unified Modeling Language
URI	Uniform Resource Identifier
URL	Uniform Resource Locator
VIN	vehicle identification number
WADL	Web Application Description Language
WAR	web archive
WSDL	Web Services Description Language
WSE	Workgroup Server Edition
WTDS	web transparent decision service
XML	Extensible Markup Language
XOM	execution object model
XSD	XML Schema Definition
XU	Execution Unit
YAML	YAML Ain't Markup Language
z/OS	Z Series Operating System

Appendix B. ODM on Cloud

Estimated time

00:00

Overview

This appendix provides supplementary information about ODM 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
- Compatibility ODM on-premises

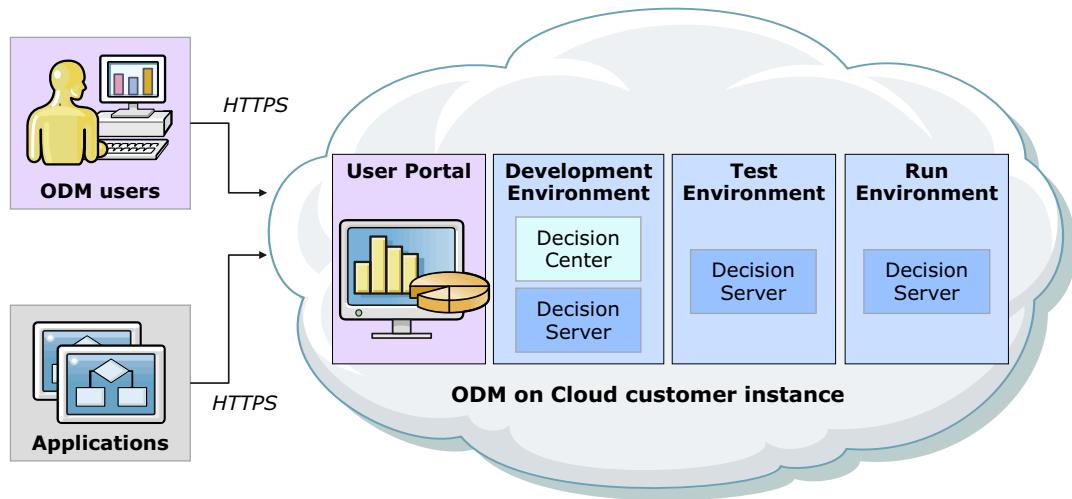
- Free trial available
 - Click **Explore trial options** at this website:
<https://www.ibm.com/products/operational-decision-manager/editions>

ODM on Cloud

© Copyright IBM Corporation 2021

Figure B-1. Introduction to IBM ODM on Cloud

IBM ODM on Cloud: Three runtime environments



ODM on Cloud

© Copyright IBM Corporation 2021

Figure B-2. IBM ODM on Cloud: Three runtime environments

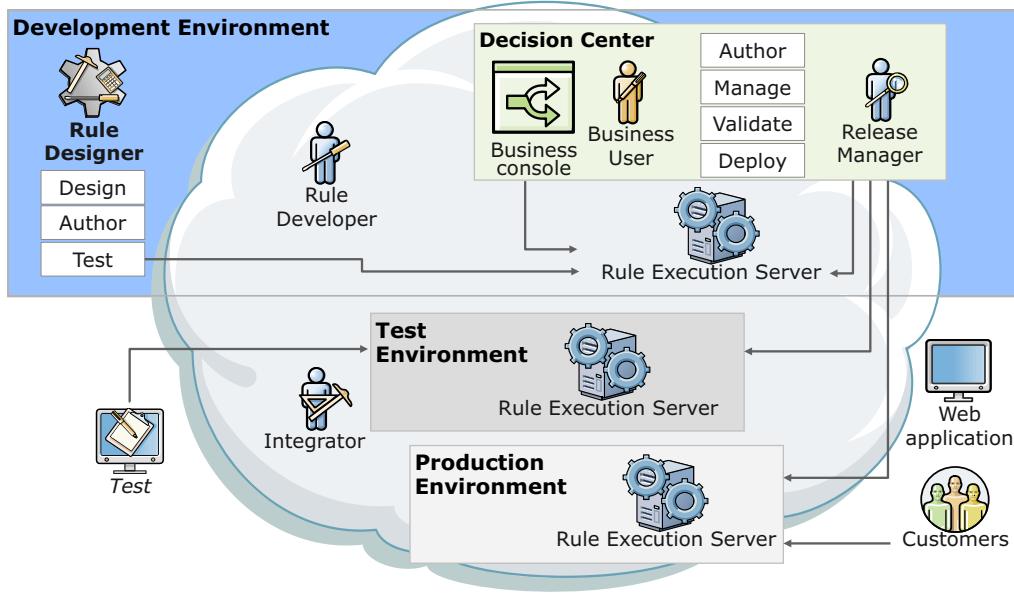
IBM ODM on Cloud provides three runtime environments for decision management:

- Development
- Test
- 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.

Workflow



ODM on Cloud

© Copyright IBM Corporation 2021

Figure B-3. 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.

Activating access and logging in to IBM ODM on Cloud

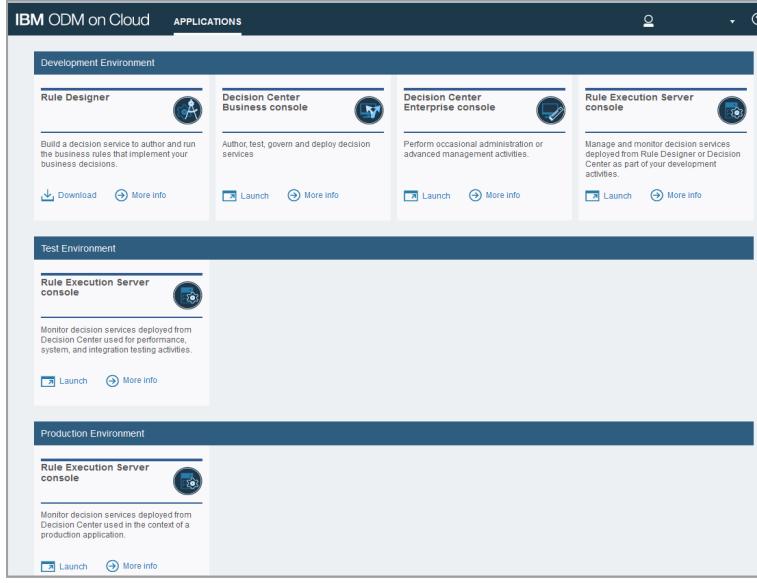
- Welcome email includes the following information:
 - Link to activate ODM on Cloud access
 - Link to ODM on Cloud instance
- Activation link is tied to a specific email
- After activating access, you can log in to your ODM on Cloud instance

ODM on Cloud

© Copyright IBM Corporation 2021

Figure B-4. Activating access and logging in to IBM ODM on Cloud

IBM ODM on Cloud user portal



© Copyright IBM Corporation 2021

Figure B-5. 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`



ODM on Cloud

© Copyright IBM Corporation 2021

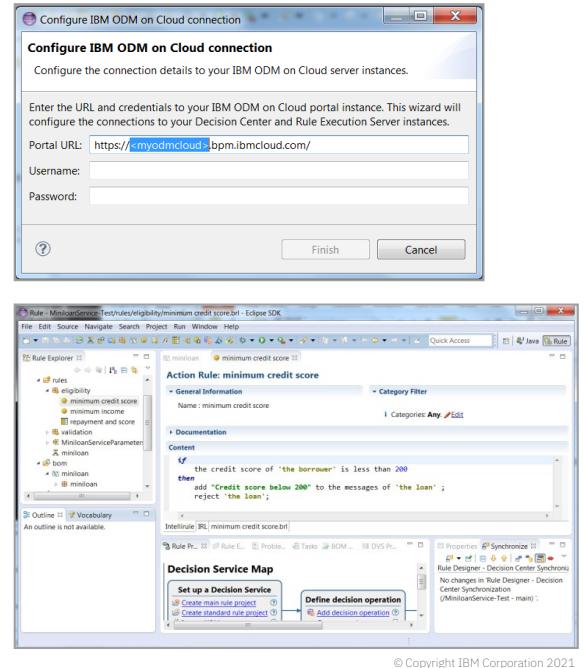
Figure B-6. Using Rule Designer

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

ODM on Cloud

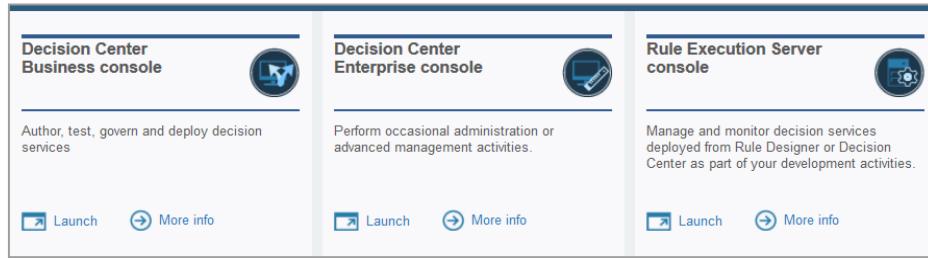
Figure B-7. Using Rule Designer (2 of 2)



© Copyright IBM Corporation 2021

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



ODM on Cloud

© Copyright IBM Corporation 2021

Figure B-8. 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

ODM on Cloud

© Copyright IBM Corporation 2021

Figure B-9. 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
 - Complete product documentation for IBM ODM on Cloud, including a “Getting Started” tutorial
- IBM ODM Support Portal
 - https://www.ibm.com/mysupport/s/topic/0TO50000000IMpWGAW/operational-decision-manager?language=en_US&productId=01t50000004uSSg
 - Support Portal provides tools and resources for help with IBM Operational Decision Manager
 - Open service requests, view fix lists, access community resources, and more

ODM on Cloud

© Copyright IBM Corporation 2021

Figure B-10. 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.



IBM Training



© Copyright International Business Machines Corporation 2021.