

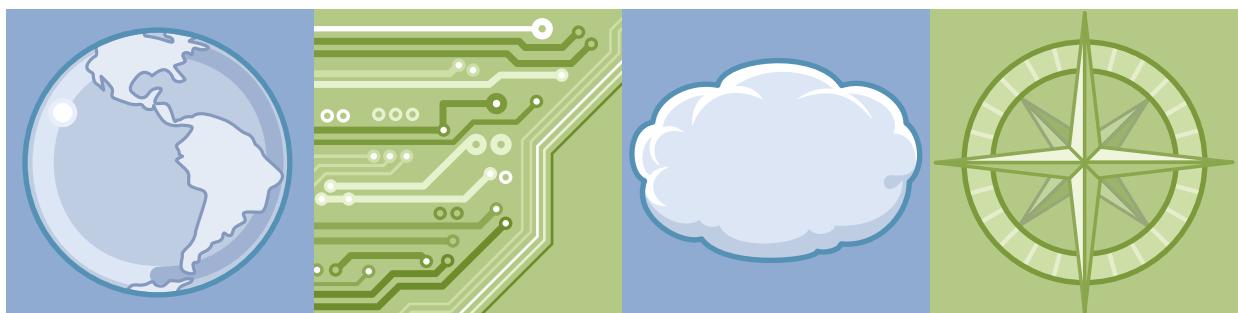


IBM Training

Student Notebook

Developing Rule Solutions in IBM Operational Decision Manager V8.8

Course code WB395 ERC 1.2



IBM Cloud
Middleware

Trademarks

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®	developerWorks®	Express®
ILOG®	Insight™	Insights™
Orchestrate®	PartnerWorld®	Redbooks®
SPSS®	Tivoli®	WebSphere®
z/OS®		

Intel is a trademark or registered trademark 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 and the VMware "boxes" logo and design, Virtual SMP and VMotion are registered trademarks or trademarks (the "Marks") of VMware, Inc. in the United States and/or other jurisdictions.

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

March 2016 edition

The information contained in this document has not been submitted to any formal IBM test and is distributed on an "as is" basis without any warranty either express or implied. The use of this information or the implementation of any of these techniques is a customer responsibility and depends on the customer's ability to evaluate and integrate them into the customer's operational environment. While each item may have been reviewed by IBM for accuracy in a specific situation, there is no guarantee that the same or similar results will result elsewhere. Customers attempting to adapt these techniques to their own environments do so at their own risk.

Contents

Trademarks	xix
Course description	xxiii
Agenda	xxv
Unit 1. Introducing IBM Operational Decision Manager V8.8.....	1-1
Unit objectives	1-2
How to check online for course material updates	1-3
Topics	1-4
1.1. What is a business rule?	1-5
What is a business rule?	1-6
What is a business rule?	1-7
From business policy to business rules	1-8
Discussion	1-9
Discussion: What is a rule?	1-10
1.2. Why the need for decision management?.....	1-11
Why the need for decision management?	1-12
Why is there a need for decision management?	1-13
What is decision management?	1-15
1.3. What is operational decision management?	1-17
What is operational decision management?	1-18
Challenges to decision automation challenges	1-19
Using operational decision management	1-20
What is operational decision management?	1-22
Synchronized business and IT cycles	1-23
1.4. Operational decision management: How events and rules work together	1-25
Operational decision management: How events and rules work together	1-26
Contextual data: Use business rules	1-27
Situational triggers: Use event rules	1-28
Complementary decision strategies	1-29
Combining business rules and events (1 of 3)	1-30
Combining business rules and events (2 of 3)	1-32
Combining business rules and events (3 of 3)	1-33
Runtime and design time activities	1-36
1.5. Introducing IBM Operational Decision Manager	1-37
Introducing IBM Operational Decision Manager	1-38
Operational Decision Manager (1 of 3)	1-39
Operational Decision Manager (2 of 3)	1-41
Operational Decision Manager (3 of 3)	1-43
Decision Center: Authoring and maintaining business rules	1-45
Decision Server: Development and execution	1-46
Decision Server Events	1-47
Decision Server Insights	1-48
Operational Decision Manager components	1-50
Operational Decision Manager editions	1-52
Installable components	1-54
1.6. IBM ODM on Cloud	1-55

IBM ODM on Cloud	1-56
Introducing IBM ODM on Cloud	1-57
IBM ODM on Cloud: Three runtime environments	1-58
IBM ODM on Cloud user portal	1-59
ODM on Cloud: Rule Designer	1-60
ODM on Cloud: Decision Center and Rule Execution Server	1-61
1.7. Integration with IBM product family	1-63
Integration with IBM product family	1-64
Synergies with other IBM products	1-65
1.8. Operational Decision Manager roles	1-67
Operational Decision Manager roles	1-68
Operational Decision Manager user roles	1-69
Interaction between roles	1-70
Business analyst	1-71
Policy manager	1-72
Rule author	1-73
Architect	1-74
Developer	1-75
Administrator	1-76
Discussion	1-77
1.9. Governance and decision management	1-79
Governance and decision management	1-80
Applying governance to decision management	1-81
Governance in Operational Decision Manager	1-83
1.10. Working with Decision Server Rules and Decision Center	1-85
Working with Decision Server Rules and Decision Center	1-86
Operational Decision Manager roles and activities	1-87
Unit summary	1-88
Checkpoint questions	1-89
Checkpoint answers	1-90
Exercise 1	1-91
Exercise objectives	1-92
Exercise workflow	1-93
Unit 2. Operational Decision Manager: Business rules	2-1
Unit objectives	2-2
Topics	2-3
2.1. Using an agile development approach	2-5
Using an agile development approach	2-6
Lifecycle of business decisions	2-7
Agile Business Rule Development methodology	2-8
Agile development approach	2-9
Discovery: Identifying decisions in the process (1 of 2)	2-11
Discovery: Identifying decisions in the process (2 of 2)	2-12
Discovery: Identifying the rules	2-14
Discovery: Identifying vocabulary	2-15
Implementing the model	2-16
Using an agile development approach	2-17
2.2. Designing the business rule application	2-19
Designing the business rule application	2-20
Overview of developer tasks	2-21

Starting point for development: Design	2-23
Rule projects (1 of 2)	2-24
Rule projects (2 of 2)	2-25
Decision service rule projects	2-27
2.3. Setting up decision services	2-29
Setting up decision services	2-30
Decision service map	2-31
Create a decision service rule project in Rule Designer	2-32
Decision service rule project templates	2-33
Design: XOM and BOM	2-34
Design: XOM	2-35
Design: BOM	2-36
Defining the decision operation	2-37
Using the Operation Map	2-38
Designing the signature	2-39
Designing the signature	2-40
2.4. Project properties	2-41
Project properties	2-42
Properties: Ruleset parameters	2-43
Properties: Folders	2-44
Properties: Project references	2-45
Properties: Rule engine type	2-46
2.5. Using modular project organization	2-47
Using modular project organization	2-48
Modular structure (1 of 4)	2-49
Modular structure (2 of 4)	2-50
Modular structure (3 of 4)	2-51
Modular structure (4 of 4)	2-52
2.6. Sharing and synchronizing decision services	2-53
Sharing and synchronizing decision services	2-54
Synchronization	2-55
Synchronization between users	2-56
Synchronization tools	2-57
Synchronization architecture	2-58
Connection entries	2-59
Choosing how to synchronize	2-60
Synchronization commands	2-61
Choosing the master source	2-62
Unit summary	2-63
Checkpoint questions	2-64
Checkpoint answers	2-65
Exercise 2	2-66
Exercise objectives	2-67
 Unit 3. Programming with business rules	 3-1
Unit objectives	3-2
Topics	3-3
3.1. Introducing ruleset integration and execution	3-5
Introducing ruleset integration and execution	3-6
Integrating Operational Decision Manager	3-7
Integration steps	3-8

Execution environments	3-9
3.2. What is a rule engine?	3-11
What is a rule engine?	3-12
Rule engine	3-13
Rule execution by the rule engine	3-15
Execution modes	3-16
Rule engine types	3-17
3.3. Understanding rule execution modes	3-19
Understanding rule execution modes	3-20
Execution modes: RetePlus (1 of 3)	3-21
Execution modes: RetePlus (2 of 3)	3-23
Execution modes: RetePlus (3 of 3)	3-24
Execution modes: Sequential (1 of 2)	3-25
Execution modes: Sequential (2 of 2)	3-26
Execution modes: Fastpath (1 of 2)	3-27
Execution modes: Fastpath (2 of 2)	3-28
RetePlus: Rule order and selection	3-29
Refraction (1 of 2)	3-30
Refraction (2 of 2)	3-31
Priority	3-32
Recency (1 of 2)	3-33
Recency (2 of 2)	3-34
3.4. RetePlus example: Trucks and drivers	3-35
RetePlus example: Trucks and drivers	3-36
RetePlus in action	3-37
Input objects in working memory	3-38
Evaluating rules with objects	3-39
Rule instances in the agenda for execution (1 of 2)	3-40
Rule instances in the agenda for execution (2 of 2)	3-41
ChangeTire executes	3-42
Side effects: Updated objects create matches	3-43
AssignDriver executes	3-44
Side effects: Updated objects no longer match	3-45
AssignDriver executes again	3-46
Side effects of AssignDriver	3-47
Rule execution completes when agenda is empty	3-48
Unit summary	3-49
Checkpoint questions	3-50
Checkpoint answers	3-51
Unit 4. Developing object models	4-1
Unit objectives	4-2
Topics	4-3
4.1. Designing the models	4-5
Designing the models	4-6
Introduction to the models	4-7
BOM and XOM at run time	4-8
Designing the BOM and the XOM	4-9
Example: Car insurance class diagram	4-10
Example: Implementation of insurance model	4-11
Example: Implementation of insurance model	4-12

4.2. Defining business and execution object models	4-13
Defining business and execution object models	4-14
Business object model (BOM)	4-15
BOM entries	4-16
BOM path	4-17
Execution object model (XOM)	4-18
Types of XOM	4-19
Rule project and XOM	4-20
4.3. Editing the business object model	4-21
Editing the business object model	4-22
Introduction	4-23
BOM editor (1 of 2)	4-24
BOM editor (2 of 2)	4-25
General information for methods	4-26
General information for attributes	4-27
4.4. Mapping the BOM to the XOM	4-29
Mapping the BOM to the XOM	4-30
Introduction	4-31
BOM to XOM Mapping section in the BOM editor	4-33
Types of BOM-to-XOM mapping	4-34
Use of explicit mappings	4-36
Explicit IRL mapping	4-37
Explicit extender mapping	4-38
4.5. Working with the vocabulary	4-39
Working with the vocabulary	4-40
Rule vocabulary	4-41
Vocabulary and verbalization	4-42
Verbalization in the BOM editor	4-43
Verbalization (1 of 2)	4-44
Verbalization (2 of 2)	4-45
Placeholders	4-46
Verbalizing BOM members	4-47
4.6. Refactoring	4-49
Refactoring	4-50
Refactoring	4-51
From the XOM to the rules	4-52
BOM and XOM evolution	4-53
XOM changes (1 of 3)	4-54
XOM changes (2 of 3)	4-55
XOM changes (3 of 3)	4-56
BOM changes	4-57
Vocabulary changes	4-58
Unit summary	4-59
Checkpoint questions	4-60
Checkpoint answers	4-61
Exercise 3	4-62
Exercise objectives	4-63
Exercise 4	4-64
Exercise objectives	4-65

Unit 5. Orchestrating ruleset execution	5-1
Unit objectives	5-2
Topics	5-3
5.1. Controlling rule execution: Overview	5-5
Controlling rule execution: Overview	5-6
What is orchestration?	5-7
Key elements for orchestration	5-8
5.2. Designing ruleflows	5-9
Designing ruleflows	5-10
Ruleflows	5-11
Ruleflow elements	5-13
Rule tasks	5-14
Initial and final actions	5-15
Transitions	5-16
Forks and joins	5-17
Expression of initial or final actions, and conditions	5-18
Main ruleflow	5-19
5.3. Controlling rule selection for execution	5-21
Controlling rule selection for execution	5-22
Rule selection pipe	5-23
Ruleflow scope selection	5-24
Runtime rule selection	5-25
Rule hierarchies	5-26
Rule overriding	5-27
Rule condition evaluation	5-28
5.4. Controlling rule order during rule execution	5-29
Controlling rule order during rule execution	5-30
Introduction	5-31
Rule priority	5-32
Rule task execution order (1 of 2)	5-33
Rule task execution order (2 of 2)	5-34
Execution modes	5-35
Unit summary	5-36
Checkpoint questions	5-37
Checkpoint answers	5-38
Exercise 5	5-39
Exercise objectives	5-40
Unit 6. Authoring rules	6-1
Unit objectives	6-2
Topics	6-3
6.1. From business policy to business rules	6-5
From business policy to business rules	6-6
Rule language evolves during a project (1 of 2)	6-7
Rule language evolves during a project (2 of 2)	6-9
When rule authoring occurs	6-10
Who writes rules?	6-11
Where are rules written?	6-12
6.2. Authoring action rules with BAL	6-13
Authoring action rules with BAL	6-14
Business Action Language (BAL)	6-15

Rule structure	6-16
Example of action rule	6-17
Rule definitions: Overview	6-19
Rule definitions: Values	6-20
Rule definitions: Constraints	6-21
Multiple variables in BAL	6-22
Rule conditions: Introduction	6-23
Rule conditions: Comparison test	6-24
Rule conditions: Membership test	6-25
Rule conditions: Existence test	6-26
Rule conditions: Count test	6-27
Multiple conditions	6-28
Avoiding ambiguity with parentheses (1 of 2)	6-29
Avoiding ambiguity with parentheses (2 of 2)	6-30
Use of commas (,) in rule conditions	6-31
Rule actions	6-32
Examples of rule actions	6-34
BAL number operators	6-35
BAL arithmetic operators	6-36
6.3. Authoring decision tables and trees	6-37
Authoring decision tables and trees	6-38
Decision tables and trees	6-39
Preconditions	6-40
Error checking in the editors	6-41
About decision tables	6-42
Example: Symmetrical rules	6-43
Example: Symmetrical rules in a decision table	6-44
Structure of a decision table	6-45
Columns for conditions and actions	6-46
Locking facilities	6-47
Decision table size	6-48
About decision trees	6-49
Decision tree (pseudocode)	6-50
Corresponding decision tree	6-51
6.4. Working with templates	6-53
Working with templates	6-54
Creating templates for business users	6-55
6.5. Introducing ILOG Rule Language	6-57
Introducing ILOG Rule Language	6-58
ILOG Rule Language (IRL)	6-59
Translation of business rule artifacts	6-60
Language for technical rule artifacts	6-61
IRL syntax	6-62
Multiple variables in IRL	6-64
Technical rules	6-66
6.6. Defining objects that are used in rules	6-67
Defining objects that are used in rules	6-68
Introduction	6-69
Ruleset parameters	6-70
Ruleset variables	6-71
Objects in working memory	6-72

Working with objects in rule artifacts	6-73
Using IRL	6-74
Using verbalized elements in BAL	6-75
Rule variables (1 of 2)	6-76
Rule variables (2 of 2)	6-77
Using automatic variables	6-78
Unit summary	6-79
Checkpoint questions	6-80
Checkpoint answers	6-81
Exercise 6	6-82
Exercise objectives	6-83
Exercise 7	6-84
Exercise objectives	6-85
Exercise 8	6-86
Exercise objectives	6-87
Unit 7. Customizing rule vocabulary with categories and domains	7-1
Unit objectives	7-2
Topics	7-3
7.1. Simplifying vocabulary with categories	7-5
Simplifying vocabulary with categories	7-6
Categories in the BOM editor	7-7
Categories	7-8
Category semantics	7-9
7.2. Defining domains	7-11
Defining domains	7-12
Domains section	7-13
7.3. Static domains	7-15
Static domains	7-16
Domains: Literals	7-17
Domains: Static references	7-18
Domains: Bounded	7-19
Domains: Collection	7-20
Domains: Other	7-21
7.4. Dynamic domains	7-23
Dynamic domains	7-24
Domains: Dynamic (1 of 2)	7-25
Domains: Dynamic (2 of 2)	7-26
Dynamic domains: Microsoft Excel source (1 of 4)	7-27
Dynamic domains: Microsoft Excel source (2 of 4)	7-29
Dynamic domains: Microsoft Excel source (3 of 4)	7-30
Dynamic domains: Microsoft Excel source (4 of 4)	7-31
Enumerated versus complex domains	7-32
Automatic creation of domains	7-33
Unit summary	7-34
Checkpoint questions	7-35
Checkpoint answers	7-36
Exercise 9	7-37
Exercise objectives	7-38
Exercise 10	7-39
Exercise objectives	7-40

Unit 8. Working with queries	8-1
Unit objectives	8-2
Topics	8-3
8.1. Searching for rule artifacts	8-5
Searching for rule artifacts	8-6
Search	8-7
Search for rule dependencies	8-8
8.2. Querying rules	8-9
Querying rules	8-10
Introduction	8-11
When queries are useful (1 of 2)	8-12
When queries are useful (2 of 2)	8-13
Business Query Language (BQL)	8-14
Query conditions	8-15
Example of rule query conditions	8-16
Filter on properties	8-17
Filter on definitions	8-18
Filter on behavior	8-20
Query actions	8-21
Queries and ruleset extractor	8-22
8.3. Extracting rulesets	8-23
Extracting rulesets	8-24
Ruleset archive	8-25
Unit summary	8-26
Checkpoint questions	8-27
Checkpoint answers	8-28
Exercise 11	8-29
Exercise objectives	8-30
Unit 9. Debugging rules	9-1
Unit objectives	9-2
Topics	9-3
9.1. Working with launch configurations	9-5
Working with launch configurations	9-6
Running and debugging decision services	9-7
Defining launch configuration (1 of 2)	9-8
Defining launch configuration (2 of 2)	9-9
9.2. Using Rule Designer debug tools	9-11
Using Rule Designer debug tools	9-12
Rule Designer debugging tools	9-13
Rule Designer debug views	9-14
Unit summary	9-16
Checkpoint questions	9-17
Checkpoint answers	9-18
Exercise 12	9-19
Exercise objectives	9-20
Exercise 13	9-21
Exercise objectives	9-22
Unit 10. Enabling tests and simulations	10-1
Unit objectives	10-2

Topics	10-3
10.1. Overview of testing and simulation	10-5
Overview of testing and simulation	10-6
What is testing and simulation?	10-7
Decision Runner and Scenario Service Provider (SSP) (1 of 2)	10-8
Decision Runner and Scenario Service Provider (SSP) (2 of 2)	10-10
What are scenarios?	10-11
Example scenario: Loan application	10-12
Reports	10-13
KPI results	10-15
Validation services in Operational Decision Manager	10-16
Testing and simulation in the decision lifecycle	10-18
How does it work?	10-19
10.2. Setting up testing and simulation	10-21
Setting up testing and simulation	10-22
Setting up testing and simulation functionality	10-23
Generating scenario file templates	10-25
10.3. Working with scenarios	10-27
Working with scenarios	10-28
Excel scenario files and the BOM	10-29
Relationships in the BOM (1 of 2)	10-30
Relationships in the BOM (2 of 2)	10-31
Structure of the scenario files in Microsoft Excel (1 of 3)	10-32
Structure of the scenario files in Microsoft Excel (2 of 3)	10-33
Structure of the scenario files in Microsoft Excel (3 of 3)	10-35
Expected results (1 of 2)	10-36
Expected results (2 of 2)	10-37
Adding and removing columns in the Microsoft Excel file	10-38
Adding columns with virtual attributes	10-39
Removing columns in the Microsoft Excel file	10-40
Unit summary	10-41
Checkpoint questions	10-42
Checkpoint answers	10-43
Exercise 14	10-44
Exercise objectives	10-45
Unit 11. Managing deployment	11-1
Unit objectives	11-2
Topics	11-3
11.1. Introducing managed execution	11-5
Introducing managed execution	11-6
Introduction to Rule Execution Server	11-7
Key functions of Rule Execution Server	11-8
Ruleset execution with Rule Execution Server	11-9
Setting up a Rule Execution Server	11-11
11.2. Deploying decision services	11-13
Deploying decision services	11-14
Deployment principles	11-15
RuleApp	11-16
Ruleset path	11-17
RuleApp archive	11-18

RuleApp properties and ruleset properties	11-19
Deployment configurations (1 of 4)	11-21
Deployment configurations (2 of 4)	11-22
Deployment configurations (3 of 4)	11-23
Deployment configurations (4 of 4)	11-24
Deployment from Decision Center	11-25
Deployment from Rule Execution Server console	11-26
Deployment with Ant scripts	11-27
Deployed RuleApp and ruleset names	11-28
11.3. Managing XOMs	11-31
Managing XOMs	11-32
XOM deployment with decision services	11-33
Modifying the XOM	11-34
Managed Java XOM elements	11-35
Link from a ruleset to a managed Java XOM element	11-36
Java XOM deployment	11-37
XOM deployment in Decision Center	11-38
Managed XOM and Decision Center	11-39
Managed XOM, Rule Designer, and Decision Center	11-40
Managed XOM storage	11-42
Unit summary	11-43
Checkpoint questions	11-44
Checkpoint answers	11-45
Exercise 15	11-46
Exercise objectives	11-47
Unit 12. Executing rules with Rule Execution Server	12-1
Unit objectives	12-2
Topics	12-3
12.1. Managed execution with Rule Execution Server	12-5
Managed execution with Rule Execution Server	12-6
Rule Execution Server in a production enterprise application	12-7
Introducing managed execution with Rule Execution Server	12-8
Example of possible platforms	12-9
12.2. Rule Execution Server modular architecture	12-11
Rule Execution Server modular architecture	12-12
Rule Execution Server architecture	12-13
Execution stack: Execution components	12-15
Execution Unit	12-17
JMX management and execution model	12-18
Management and monitoring stack: Console	12-20
Persistence layer	12-21
Ant tasks	12-23
12.3. Managed execution in action	12-25
Managed execution in action	12-26
Elements of a rule-based solution	12-27
Ruleset parsing	12-28
Ruleset execution	12-30
Ruleset update at run time	12-31
Runtime class loading	12-32
12.4. Platforms for Rule Execution Server	12-33

Platforms for Rule Execution Server	12-34
Architecture questions	12-35
Possible choices: Introduction	12-36
Java SE	12-38
Java EE	12-39
Java SE and Java EE: A summary	12-41
Transparent decision services	12-42
12.5. Introducing the Rule Execution Server API	12-45
Introducing the Rule Execution Server API	12-46
Introduction	12-47
Execution patterns: Synchronous	12-48
Execution patterns: Asynchronous	12-49
Rule session API overview (1 of 2)	12-50
Rule session API overview (2 of 2)	12-51
Rule session factories	12-52
Rule sessions	12-53
Stateless rule sessions	12-54
Stateful rule sessions	12-55
Rule session requests	12-56
Rule session requests: Decision ID	12-57
Rule session responses	12-58
Ruleset path for execution (1 of 2)	12-59
Ruleset path for execution (2 of 2)	12-60
Traces	12-62
Decision traces: How to	12-64
Decision traces: Complement	12-66
12.6. Executing rules in Java SE	12-67
Executing rules in Java SE	12-68
Rule execution in Java SE	12-69
Deployed Rule Execution Server artifacts in Java SE	12-70
Required Rule Execution Server API in Java SE	12-71
Example with a stateless rule session in Java SE	12-72
12.7. Executing rules in Java EE	12-75
Executing rules in Java EE	12-76
Rule execution in Java EE	12-77
Deployed Rule Execution Server artifacts in Java EE	12-78
Required API in Java EE	12-79
Java EE POJO rule session	12-80
Java EE EJB rule session	12-81
Java EE message-driven bean (MDB)	12-82
12.8. Executing rules as transparent decision services	12-83
Executing rules as transparent decision services	12-84
Rules as transparent decision services	12-85
Hosted transparent decision service	12-86
Monitored transparent decision service	12-87
Unit summary	12-88
Checkpoint questions (1 of 2)	12-89
Checkpoint questions (2 of 2)	12-90
Checkpoint answers	12-91
Exercise 16	12-92
Exercise objectives	12-93

Exercise 17	12-94
Exercise objectives	12-95
Exercise 18	12-96
Exercise objectives	12-97
Unit 13. Auditing and monitoring ruleset execution	13-1
Unit objectives	13-2
Topics	13-3
13.1. Auditing ruleset execution	13-5
Auditing ruleset execution	13-6
Auditing ruleset execution	13-7
Decision Warehouse	13-8
Default use of Decision Warehouse	13-9
Step 1: Install and create database resource	13-10
Step 2: Enable rule execution monitoring	13-11
Step 3: Execute the ruleset	13-13
Step 4-a: View stored decision traces	13-14
Step 4-b: Filter stored decision traces	13-15
Step 4-c: View decision details and fired rules	13-16
13.2. Monitoring ruleset execution	13-17
Monitoring ruleset execution	13-18
Monitoring ruleset execution	13-19
Debug mode	13-20
Ruleset monitoring options (1 of 5)	13-21
Ruleset monitoring options (2 of 5)	13-22
Ruleset monitoring options (3 of 5)	13-23
Ruleset monitoring options (4 of 5)	13-24
Ruleset monitoring options (5 of 5)	13-26
Ruleset statistics (1 of 3)	13-27
Ruleset statistics (2 of 3)	13-29
Ruleset statistics (3 of 3)	13-30
Test of ruleset execution	13-31
Logged events on Execution Units	13-32
Unit summary	13-33
Checkpoint questions	13-34
Checkpoint answers	13-35
Exercise 19	13-36
Exercise objectives	13-37
Unit 14. Working with the REST API	14-1
Unit objectives	14-2
Topics	14-3
14.1. Overview of the ruleset execution REST API	14-5
Overview of the ruleset execution REST API	14-6
REST API for ruleset execution	14-7
Endpoint URLs	14-8
HTTP methods	14-9
Request and response schema	14-10
XML serialization of the XOM	14-12
WADL representation	14-13
Using the REST execution API for ruleset execution	14-14

14.2. Testing ruleset execution with the REST API	14-15
Testing ruleset execution with the REST API	14-16
Executing a ruleset (1 of 2)	14-17
Executing a ruleset (2 of 2)	14-18
Validating the XML execution request	14-19
Validation response in JSON	14-20
Execution response	14-21
14.3. Managing resources through the REST API	14-23
Managing resources through the REST API	14-24
REST API tool in Rule Execution Server console	14-25
Unit summary	14-26
Checkpoint questions	14-27
Checkpoint answers	14-28
Exercise 20	14-29
Exercise objectives	14-30
Unit 15. Introducing decision governance.....	15-1
Unit objectives	15-2
Topics	15-3
15.1. What is decision governance?.....	15-5
What is decision governance?	15-6
What is decision governance?	15-7
Why decision governance is required	15-8
Traditional approach to maintenance	15-9
Decision management increases communication	15-10
Decision governance goal	15-12
Definition of project governance	15-13
Business Decision Management group	15-14
Governance and agile development	15-16
Implementing governance	15-17
15.2. Operational Decision Manager support for governance	15-19
Operational Decision Manager support for governance	15-20
Decision lifecycle in Operational Decision Manager (1 of 2)	15-21
Decision lifecycle in Operational Decision Manager (2 of 2)	15-22
Discussion	15-23
15.3. Decision governance framework	15-25
Decision governance framework	15-26
Decision governance framework overview	15-27
Recall: Operational Decision Manager tools	15-28
Decision service (1 of 2)	15-30
Decision service (2 of 2)	15-31
Decision service properties	15-32
Decision operations (1 of 2)	15-33
Decision operations (2 of 2)	15-34
Run configurations for a decision operation	15-35
Publishing decision services from Rule Designer	15-36
Decision Center: Governance in Business console	15-37
States and user roles	15-39
Release governance	15-40
Release sequence	15-42
Change activity governance	15-43

Validation activity governance	15-44
Release deployment	15-45
Unit summary	15-46
Checkpoint questions	15-47
Checkpoint answers	15-48
Unit 16. Course summary	16-1
Unit objectives	16-2
Course learning objectives (1 of 2)	16-3
Course learning objectives (2 of 2)	16-4
To learn more on the subject	16-5
Earn an IBM Badge	16-6
Enhance your learning with IBM resources	16-7
Unit summary	16-8
Appendix A. List of abbreviations	A-1
Appendix B. Appendix: IBM ODM on Cloud	B-1
Appendix C. Resource guide.....	C-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®	developerWorks®	Express®
ILOG®	Insight™	Insights™
Orchestrate®	PartnerWorld®	Redbooks®
SPSS®	Tivoli®	WebSphere®
z/OS®		

Intel is a trademark or registered trademark 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 and the VMware "boxes" logo and design, Virtual SMP and VMotion are registered trademarks or trademarks (the "Marks") of VMware, Inc. in the United States and/or other jurisdictions.

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

For IBM ODM on Cloud students

This course is designed for developers who work with IBM Operational Decision Manager.

This course was created with IBM Operational Decision Manager Standard V8.8, but the tools and concepts that are covered in this class also apply to IBM ODM on Cloud. All of the tools that you work with in this course (Rule Designer, the Decision Center Business console, the Decision Center Enterprise console, and the Rule Execution Server console) are also available in ODM on Cloud.

Any 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 V8.8

Duration: 5 days

Purpose

This course introduces developers to IBM Operational Decision Manager V8.8. You learn the concepts and skills that you need to design, develop, and integrate a business rule solution with Operational Decision Manager. The course begins with an overview of Operational Decision Manager, which is composed of two main environments: Decision Server for technical users and Decision Center for business users. The course outlines the collaboration between development and business teams during project development. Through instructor-led presentations and hands-on lab exercises, you learn about the core features of Decision Server, which is the primary working environment for developers. You design a business rule application, and work with the object models that are required to author and execute rule artifacts. You gain experience with rule deployment and execution within various types of client applications. You work extensively with Rule Execution Server and learn how you can integrate decision services within an enterprise environment. In addition, you become familiar with rule authoring so that you can support business users to set up and customize the rule authoring and validation environments. You also learn how to use Operational Decision Manager features to support decision governance. The lab environment for this course uses Windows Server 2008 R2 Standard Edition.

Audience

This course is designed for application developers.

Prerequisites

Before taking this course, you should have:

- 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)

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

After completing this course, you should be able to:

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

Agenda

Day 1

- Course introduction
- Unit 1. Introducing IBM Operational Decision Manager V8.8
- Exercise 1. Operational Decision Manager in action
- Unit 2. Operational Decision Manager: Business rules
- Exercise 2. Setting up decision services
- Unit 3. Programming with business rules

Day 2

- Unit 4. Developing object models
- Exercise 3. Working with the BOM
- Exercise 4. Refactoring
- Unit 5. Orchestrating ruleset execution
- Exercise 5. Working with ruleflows
- Unit 6. Authoring rules
- Exercise 6. Exploring action rules
- Exercise 7. Authoring action rules

Day 3

- Exercise 8. Authoring decision tables and decision trees
- Unit 7. Customizing rule vocabulary with categories and domains
- Exercise 9. Working with static domains
- Exercise 10. Working with dynamic domains
- Unit 8. Working with queries
- Exercise 11. Working with queries
- Unit 9. Debugging rules
- Exercise 12. Executing rules locally
- Exercise 13. Debugging a ruleset

Day 4

- Unit 10. Enabling tests and simulations
- Exercise 14. Enabling tests and simulations
- Unit 11. Managing deployment
- Exercise 15. Managing deployment
- Unit 12. Executing rules with Rule Execution Server
- Exercise 16. Exploring the Rule Execution Server console

Day 5

- Exercise 17. Executing rules with Rule Execution Server in Java EE
- Exercise 18. Executing rules as a hosted transparent decision service (HTDS)
- Unit 13. Auditing and monitoring ruleset execution
- Exercise 19. Auditing ruleset execution through Decision Warehouse
- Unit 14. Working with the REST API
- Exercise 20. Working with the REST API
- Unit 15. Introducing decision governance
- Unit 16. Course summary

Unit 1. Introducing IBM Operational Decision Manager V8.8

What this unit is about

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

What you should be able to do

After completing this unit, you should be able to:

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

How you will check your progress

- Checkpoint
- Exercise



Unit objectives

After completing this unit, you should be able to:

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

© Copyright IBM Corporation 2016

Figure 1-1. Unit objectives

WB3951.2

Notes:



How to check online for course material updates



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

Instructions

1. Enter this URL in your browser:
ibm.com/developerworks/connect/middleware_edu
2. On the wiki page, locate and click the **Course Information** category.
3. Find your course in the list and then click the link.
4. The wiki page displays information for the course. If the course has a course corrections document, you can download it from this page.
5. If you want to download an attachment, such as a course corrections document, click the **Attachments** tab at the bottom of the page.
6. To save the file to your computer, click the document link, and follow the prompts.

[Comments \(0\)](#) [Versions \(1\)](#) **Attachments (1)** [About](#)

© Copyright IBM Corporation 2016

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

WB3951.2

Notes:



Topics

- What is a business rule?
- Why the need for decision management?
- What is operational decision management?
- Operational decision management: How events and rules work together
- Introducing IBM Operational Decision Manager
- IBM ODM on Cloud
- Integration with IBM product family
- Operational Decision Manager roles
- Governance and decision management
- Working with Decision Server Rules and Decision Center

© Copyright IBM Corporation 2016

Figure 1-3. Topics

WB3951.2

Notes:

1.1. What is a business rule?

What is a business rule?



© Copyright IBM Corporation 2016

Course materials may not be reproduced in whole or in part without the prior written permission of IBM.

10.1

Figure 1-4. What is a business rule?

WB3951.2

Notes:

What is a business rule?

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

© Copyright IBM Corporation 2016

Figure 1-5. What is a business rule?

WB3951.2

Notes:

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

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

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

From business policy to business rules

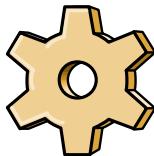
- Rules formalize business policy as “if-then” statements
- Example:

Business policy



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

Formal rule



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

© Copyright IBM Corporation 2016

Figure 1-6. From business policy to business rules

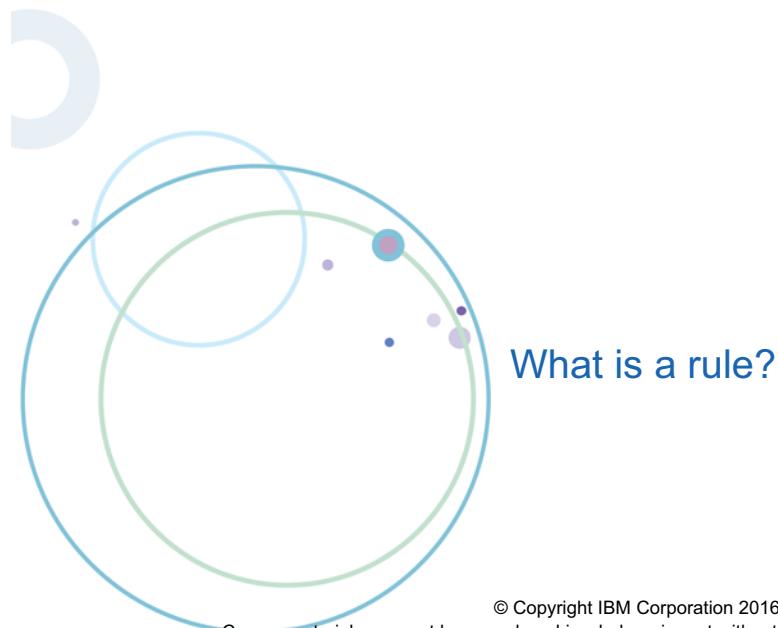
WB3951.2

Notes:

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

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

Discussion



© Copyright IBM Corporation 2016

Course materials may not be reproduced in whole or in part without the prior written permission of IBM.

10.1

Figure 1-7. Discussion

WB3951.2

Notes:

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

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



Discussion: What is a rule?

1. Write your definition of a business rule.

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

© Copyright IBM Corporation 2016

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

WB3951.2

Notes:

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

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

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

1.2. Why the need for decision management?

Why the need for decision management?



© Copyright IBM Corporation 2016

Course materials may not be reproduced in whole or in part without the prior written permission of IBM.

10.1

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

WB3951.2

Notes:

Why is there a need for decision management?

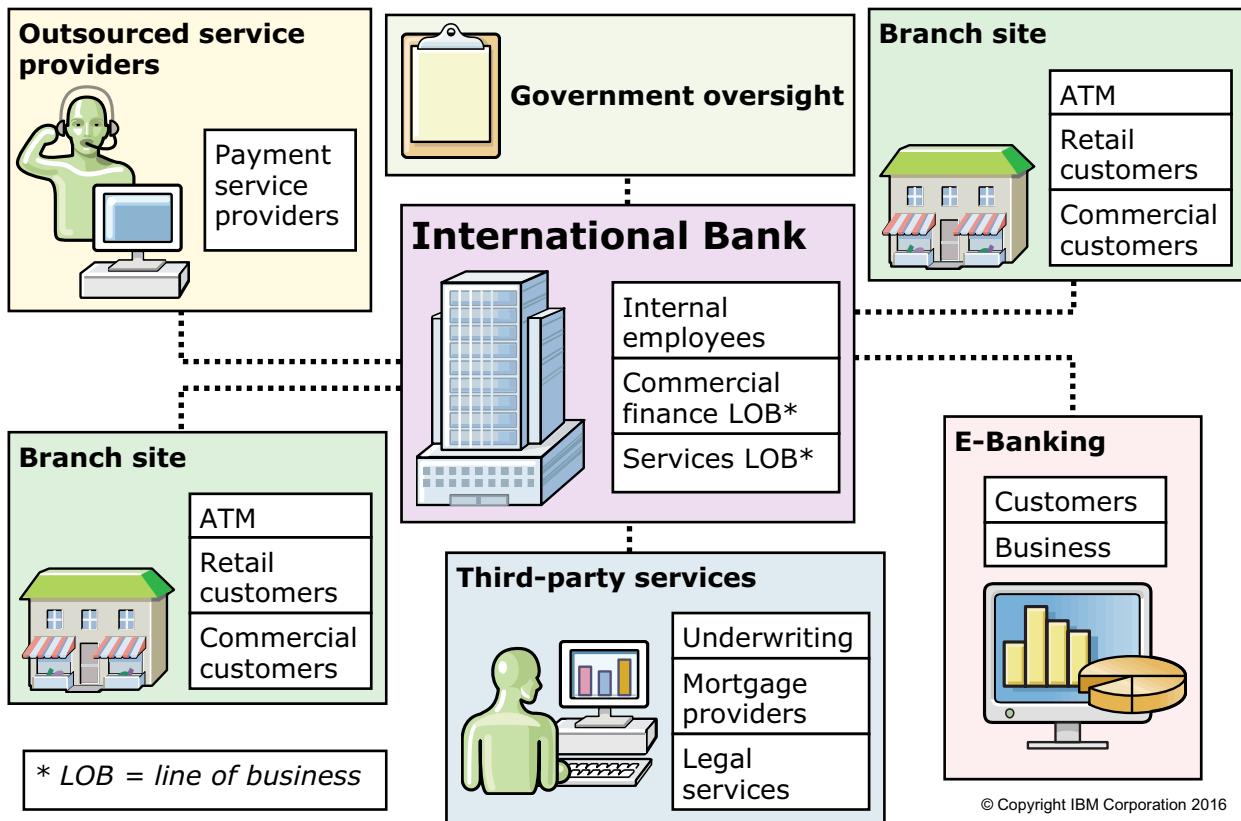


Figure 1-10. Why is there a need for decision management?

WB3951.2

Notes:

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

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

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

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

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

Within this network, businesses make thousands, even millions of *repeated and repeatable* decisions. Those decisions need to be consistent across the organization, but might vary according

to the context. The dynamic nature of this environment requires business agility to respond to change by making better decisions in real time.

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

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

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



Questions

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

What is decision management?

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

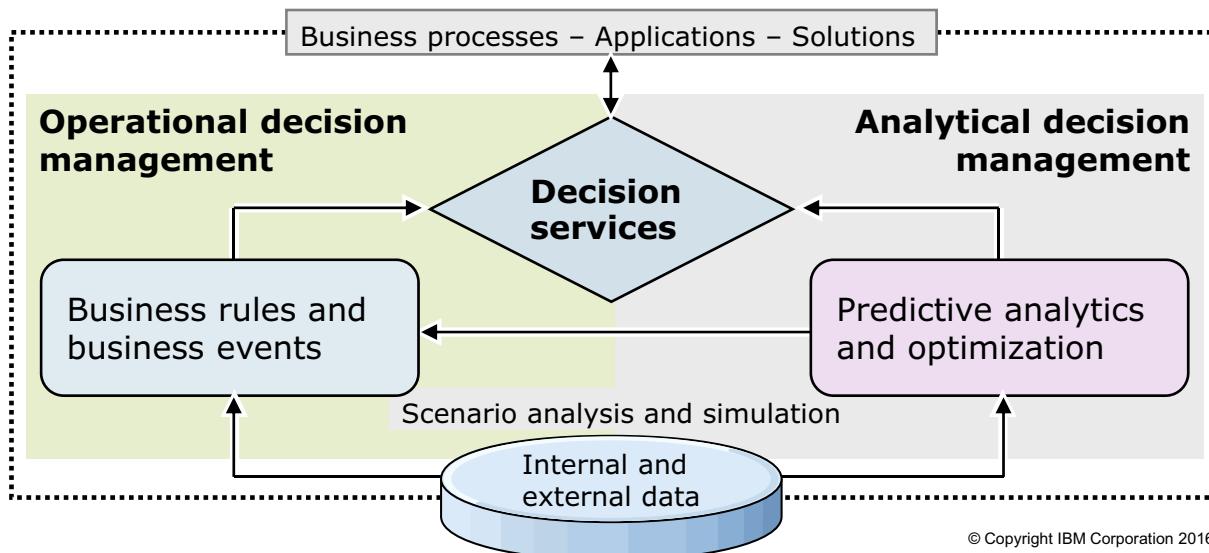


Figure 1-11. What is decision management?

WB3951.2

Notes:

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

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

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

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

- (Left side of screen) In *operational decision management*, policy, suggested practices, and business experience are used to write rules that describe decision making and identify situations that require a response. In many cases, the results from analytical decision management can enhance the operational decisions.

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



Note

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

1.3. What is operational decision management?

What is operational decision management?



© Copyright IBM Corporation 2016

Course materials may not be reproduced in whole or in part without the prior written permission of IBM.

10.1

Figure 1-12. What is operational decision management?

WB3951.2

Notes:

Challenges to decision automation challenges

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

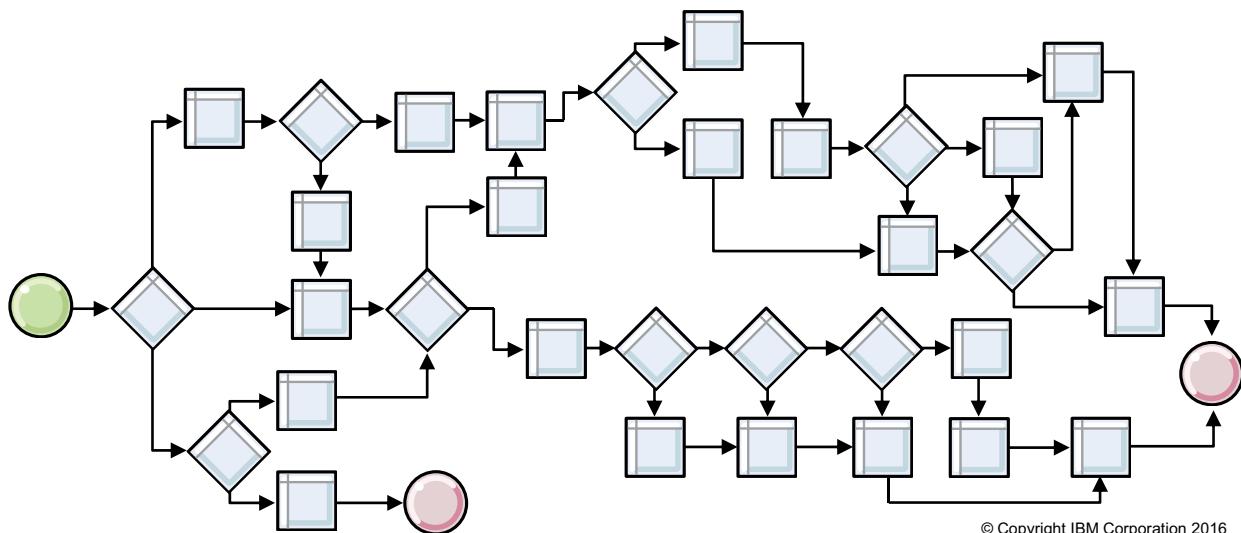


Figure 1-13. Challenges to decision automation challenges

WB3951.2

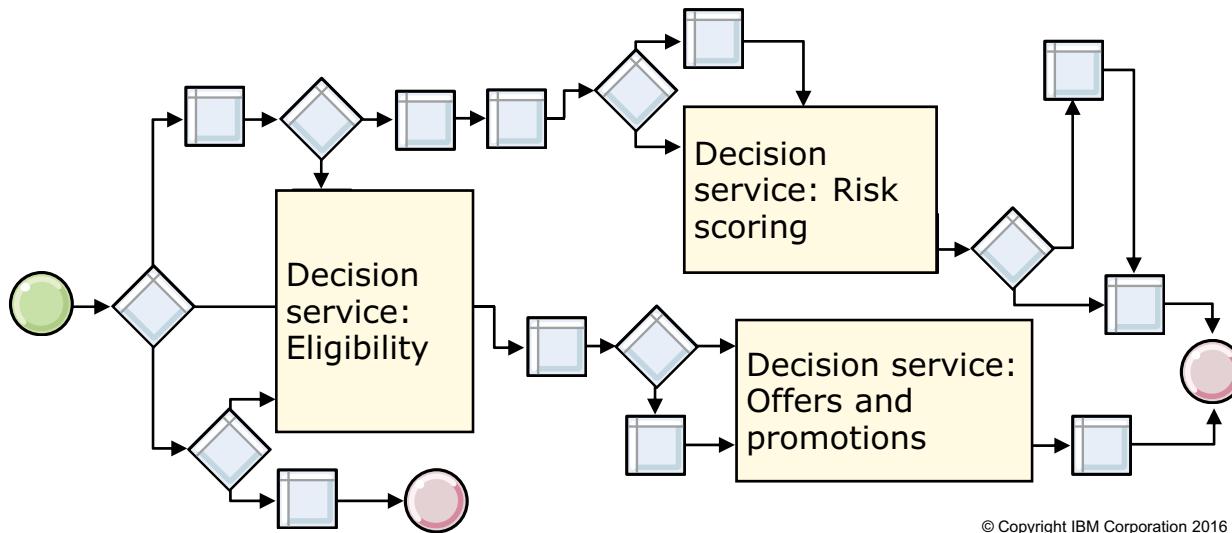
Notes:

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

Also, millions of events that potentially require a business response might be flowing freely through the IT infrastructure. Without decision management tools, business users cannot identify and respond to the volume and complexity of these situations by themselves.

Using operational decision management

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



© Copyright IBM Corporation 2016

Figure 1-14. Using operational decision management

WB3951.2

Notes:

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

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

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

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

To manage decisions, you must first recognize the context in which decisions are made. Modeling is really the key. Decisions are always made within the context of a business process, so you start

with process modeling and management. Within the process model, you can then identify the decision points and capture the individual rules that support the decision.

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

What is operational decision management?

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

© Copyright IBM Corporation 2016

Figure 1-15. What is operational decision management?

WB3951.2

Notes:

Operational decision management typically involves high-volume, repeatable decisions that are based on business rules. It also incorporates business events. This style of decision management is generally used for high-volume, repeatable decisions, which are often based on hundreds or thousands of business rules.

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

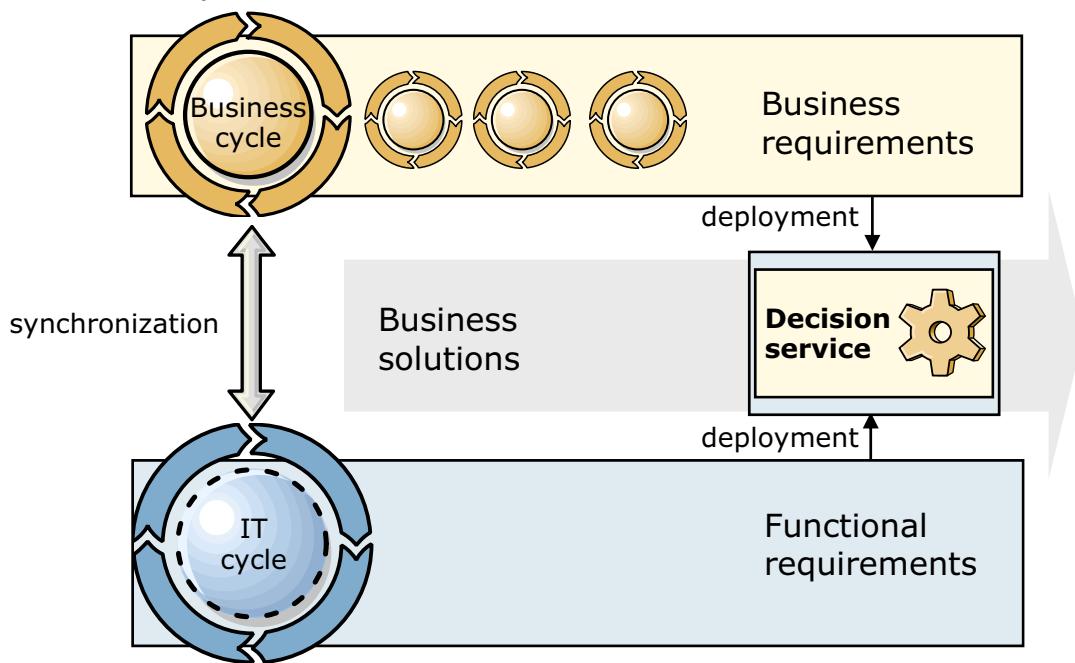
- Corporate suggested practices (explicit expertise that defines how the business is run)
- “Know-how” (implicit knowledge that is used in running the business that is not codified)

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

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

Synchronized business and IT cycles

- Decision management and application development lifecycles can evolve in parallel



© Copyright IBM Corporation 2016

Figure 1-16. Synchronized business and IT cycles

WB3951.2

Notes:

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

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

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

1.4. Operational decision management: How events and rules work together

Operational decision management: How events and rules work together



© Copyright IBM Corporation 2016

Course materials may not be reproduced in whole or in part without the prior written permission of IBM.

10.1

Figure 1-17. Operational decision management: How events and rules work together

WB3951.2

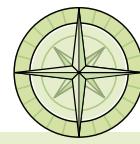
Notes:

Contextual data: Use business rules

- Business rule examples:

**if the Passenger is a gold frequent traveler
and the flight distance is more than 40000 miles
and the flight destination is in Europe or Asia
then**

Add 10,000 points to the fidelity card of the **Passenger**



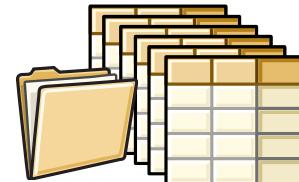
Contextual reasoning

**if the Vehicle is not an SUV
and all the following conditions are true:
• the vehicle is equipped with dual passenger airbag
• the driver has a good driver certificate
• the driver age is between 30 and 50
• the number of accidents the driver was responsible for is 0
then**

Apply a **5% discount** on the premium coverage price

- Business context:

- **Passenger** (age, address, gender, frequent travel status, company)
- **Vehicle** (VIN, manufacturer, data, type, brand)
- **Order** (amount, items)
- **Promotion** (code, amount, type, article)
- **Flight** (airline, depart, destination, distance, date)
- **Plant** (location, production, profitability)



© Copyright IBM Corporation 2016

Figure 1-18. Contextual data: Use business rules

WB3951.2

Notes:

As shown in these examples, the business rules look at fixed sets of data, which might be stored in databases or provided from other sources. The rules are evaluated against that data to produce a decision.

Situational triggers: Use event rules

- Event pattern examples:

if

the **low temperature has been detected more than 3 times in the last 5 minutes**
and the number of occurrences the **outside air temperature has increased in the past 5 minutes** is zero

then

Raise a severity alarm
Notify the maintenance team
Reduce activity by 15%



Time-based reasoning

if more than 2 **customer withdraws in an ATM** are done **in the same day**

and the 2 ATMs are from 2 foreign countries

then

Investigate possible fraud
Reduce to cash redraw max amount to \$100

- Triggers:



Incoming call at the call center



Detected RFID badge at the room entrance



Detected low temperature



Application form submission



Application rejected



Customer withdrawal in an ATM



Debts threshold exceeded

© Copyright IBM Corporation 2016

Figure 1-19. Situational triggers: Use event rules

WB3951.2

Notes:

Events are detected and correlated over time and across channels. The event rules are evaluated against the event patterns that are detected to produce a response.

Look at the example that is shown here about two customer withdrawals in an ATM done on the same day but in different countries. You can see that those two events are seemingly unrelated, over a specific time period, but they are being measured. And because of that pattern, the fact that those two events happened within that time period might be a trigger that a theft or some fraud occurred.

The ability to detect that type of trigger helps the business to respond and protect their customers.



Complementary decision strategies

Usage pattern	Event rules	Business rules
Type of data	<ul style="list-style-type: none"> Data is collected in increments and processed over time 	<ul style="list-style-type: none"> Complete data context is available at processing time
Reasoning	<ul style="list-style-type: none"> Process multiple unrelated events from varied sources to identify a pattern and respond asynchronously when this pattern is identified Process enterprise events over extended periods of time, from milliseconds to months 	<ul style="list-style-type: none"> Execute rules by using a defined order of rule execution or an order that the inference-based algorithm of the rule engine determines
Execution	<ul style="list-style-type: none"> Asynchronous Receive individual events from the network If the event completes a pattern of interest, react by initiating actions on other systems 	<ul style="list-style-type: none"> Synchronous Respond to each call as a unique transaction Called through an API, for example as a web service, or from a business process
Outcome	<ul style="list-style-type: none"> Initiate actions when patterns of business activity do not occur as expected Actions can be alerts or follow-on processing 	<ul style="list-style-type: none"> Calculate one or more values to decide and respond

© Copyright IBM Corporation 2016

Figure 1-20. Complementary decision strategies

WB3951.2

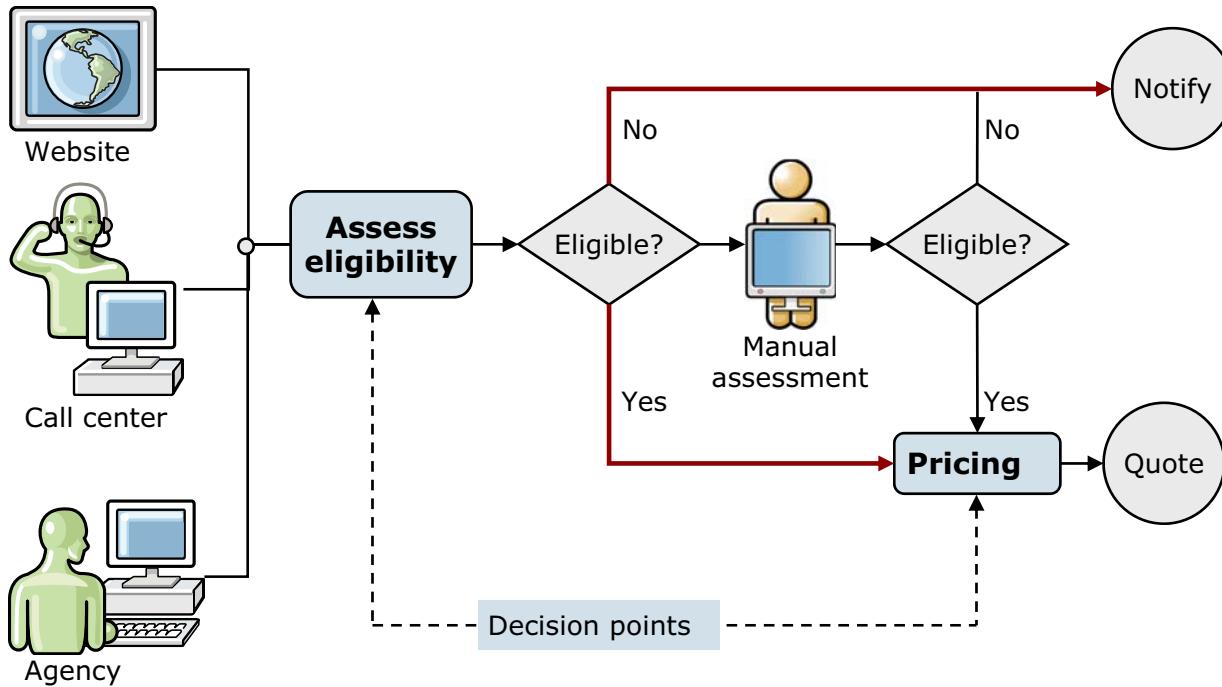
Notes:

Depending on the type of data that you are dealing with and the type of reasoning that you want to apply to the data, one of the two strategies prevails.

Business rule processing is a point-in-time analysis of related pieces of data that is based on contextual data. The rule engine evaluates the rules against a specific set of data, and returns a decision to whatever system requested the decision service.

Business event processing provides situational awareness by capturing events as they occur across various systems, tracking time stamps, and recognizing event patterns that require a response from the business. When a significant event or event pattern is detected, the event runtime triggers the appropriate system to respond.

Combining business rules and events (1 of 3)



© Copyright IBM Corporation 2016

Figure 1-21. Combining business rules and events (1 of 3)

WB3951.2

Notes:

Consider this scenario.

An insurance company provides quotations to customers through multiple channels, including a website, a call center, and an insurance agency.

To reduce the time and cost for an underwriter to manually review insurance requests, the company has two basic requirements:

- Use business rules to automate decision points and increase straight-through processing with no manual intervention.
- Based on customer interaction with the system, determine which types of promotions match customer needs. Then, provide notification on how best to follow up with the customer to finalize the sale.

How can these requirements be met by using the combined capability of rules and events?

First, by reviewing the business process model, you can determine where the decision points are.

- Within the context of the process, the process has two decision points. The process can request a decision service to provide a decision.

- When the decision is received, the process continues to the next step based on the decision results.

The decision itself is not what triggers the next task in the process because the decision service is not aware of what's happening before or after it. However, it is the process flow that orchestrates the next step to take after a decision result is received.

Combining business rules and events (2 of 3)

Event rules	<ul style="list-style-type: none"> Monitor customer behavior and identify the right timing for a promotion Detect multiple requests for insurance quotations over a short time period Detect quotation requests that are not accepted within a time period Follow up on customer promotional offers
Business rules	<ul style="list-style-type: none"> Determine whether to make an offer, and if so, for what Assess customer risk Evaluate a combination of factors to tailor the price

© Copyright IBM Corporation 2016

Figure 1-22. Combining business rules and events (2 of 3)

WB3951.2

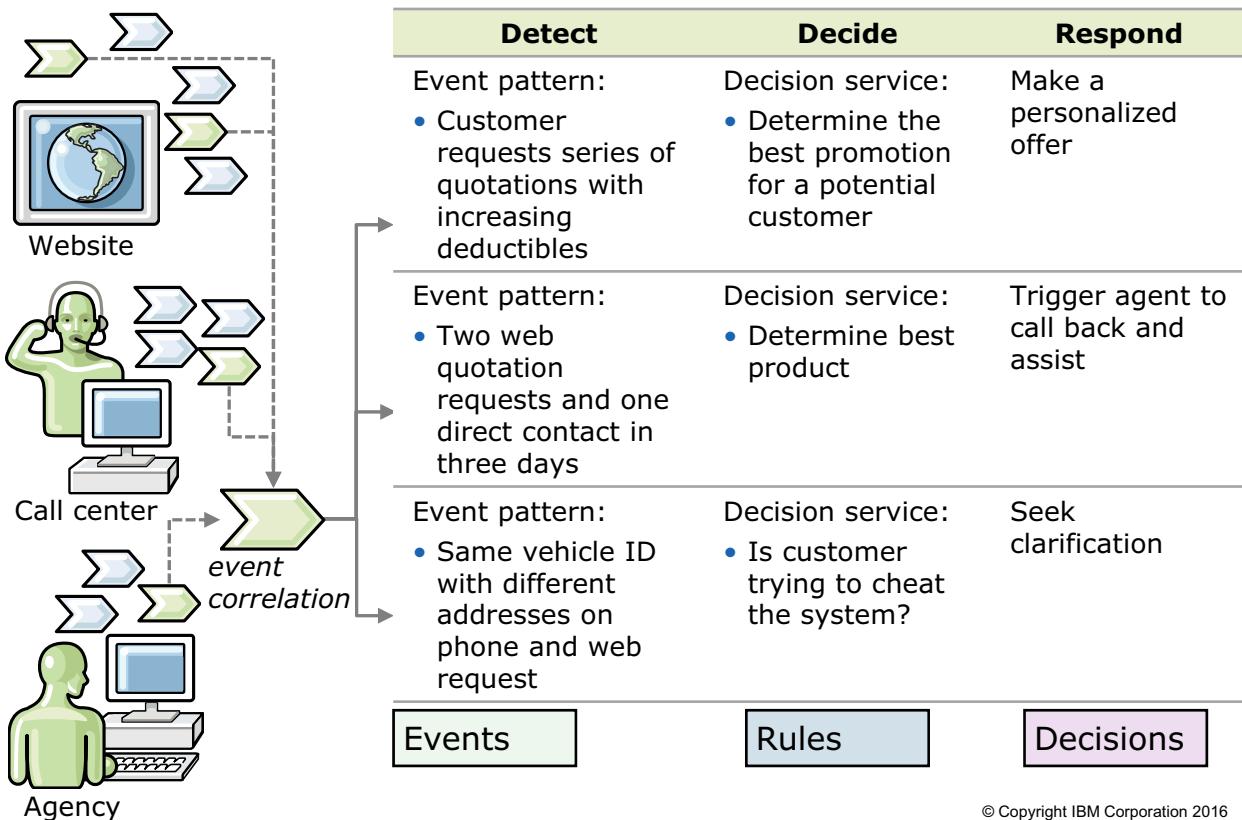
Notes:

Review the process again. This time, consider the types of events or event patterns you want to detect and what actions to take in response.

- What are the triggers that can start this process?
- What happens when the process interrupts before completion?
- What are the rules that describe the business event processing?
- What business rules are evaluated?

Events processing detects a pattern of activity, while the business rules decide which promotion is the best match for the customer. Based on the decision, the event rules determine how to respond in real time.

Combining business rules and events (3 of 3)



© Copyright IBM Corporation 2016

Figure 1-23. Combining business rules and events (3 of 3)

WB3951.2

Notes:

When you look again at this business process and see how events and rules can be applied, you can see that across the channels, the website, call center, and agency, events are being monitored. The blue and green arrows were detected, and the green arrows are showing a pattern. The event correlation box indicates that based on the pattern, a response is required from the business. In the Detect column of the table, you see what patterns can be handled. In the Decide column, you see how these patterns trigger a decision service. And in the Respond column, you see what is the best way for the business to respond to this situation according to what the rules showed.

Starting from this same business process, you can handle various use cases. The combination of rules and events can support fully automated responses, where the system itself makes an offer by either using an email or just returning a message through a web interface. A semiautomated response can be made where some message or guidance is provided to a call center agent. Or a fully manual response can be initiated where some alert is provided to the personnel and someone must investigate a transaction or issue.

Legend

- Blue arrows: Indicate business events. In this case, they indicate an insurance quotation request.

- Green arrows: Indicate insurance quotation requests that match a predefined pattern.
- Large green arrow: Indicate event correlation. In this case, they indicate insurance quotation patterns that are detected by using the business events capability.

Starting from the same business process, you can handle various use cases.

As shown in these examples, the combination of business events and business rules can support:

- Fully automated responses:

Example: Make a special offer to a customer who is looking at pricing by making multiple requests with slightly different criteria.

- Semiautomated responses:

Example: Provide guidance to a call center agent for a prospect who initiated multiple interactions with the company.

- Initiation of a manual action:

Example: Alert personnel to investigate a transaction or issue.

Scenario 1: Increasing deductibles

- Detect: A customer requests a series of quotations with increasing deductibles. The system detects that the customer is trying to get a lower monthly payment.
- Decide: Based on the detected activity pattern for this customer, the system triggers the correct set of business rules to decide:
 - Is the customer a good prospect?
 - Can a promotion be offered to the customer?
 - Which promotion would be a best match for this customer?
- Respond: A personalized offer is made to the customer in real time.

Scenario 2: Multiple forms of contact

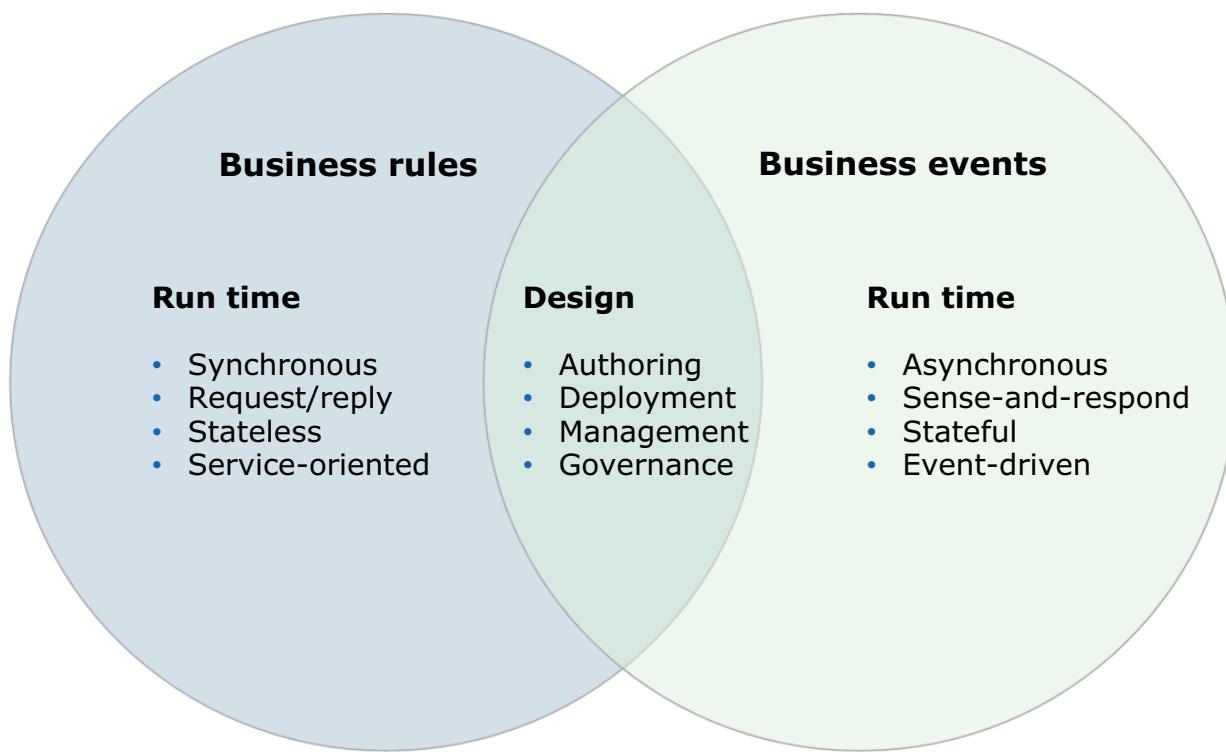
- Detect: Within the last three days, a customer made two quotation requests online and also contacted an agency. The system recognizes this pattern as an indication that the customer might be trying to find the best level of coverage for the customer's personal situation.
- Decide: Knowing that the customer is seeking clarity on the offering, the system triggers the set of business rules to run a product recommendation analysis by using the customer's specific characteristics.
- Respond: An agent can call back to assist the customer with the product recommendation and to help close the sale.

Scenario 3: Postal code experiments

- Detect: The customer requested multiple insurance quotations for the same vehicle, but each request used different postal codes. The address of where the car is garaged often determines the premium. The varying postal codes indicate that the customer is trying to work out which address results in a lower premium.

- Decide: The event pattern triggers the set of business rules that determine whether the customer is providing different addresses and postal codes to trick the system and get a lower quotation.
- Respond: An agent can either call back or send an email to ask where the customer intends to park the car and determine which is the primary address. Based on the risk level at the primary address, the correct premium can be quoted.

Runtime and design time activities



© Copyright IBM Corporation 2016

Figure 1-24. Runtime and design time activities

WB3951.2

Notes:

Runtime activities are managed separately.

As shown in this slide, business rules and events share some design-time activities. For example:

- The business process model is the starting point for discovering the business rules and determining which business events to look for.
- Use cases describe the specific interaction between the business process and business rules or business events.
- Business rules and event rules are both formulated as if-then statements and can be authored, managed, and governed in a shared environment.
- The Decision Center repository provides integrated storage for both business and event rule artifacts, which supports rule management and governance.

1.5. Introducing IBM Operational Decision Manager

Introducing IBM Operational Decision Manager



© Copyright IBM Corporation 2016

Course materials may not be reproduced in whole or in part without the prior written permission of IBM.

10.1

Figure 1-25. Introducing IBM Operational Decision Manager

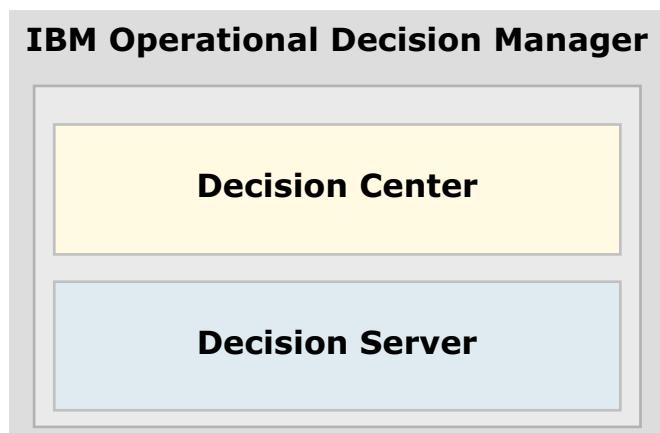
WB3951.2

Notes:



Operational Decision Manager (1 of 3)

- A combined business rules and events management platform that combines these technologies:
 - Business rule management system (BRMS)
 - Business event processing (BEP)
- Provides two main environments:
 - Decision Center for business users
 - Decision Server for technical users



© Copyright IBM Corporation 2016

Figure 1-26. Operational Decision Manager (1 of 3)

WB3951.2

Notes:

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

Operational Decision Manager combines these technologies:

- WebSphere JRules Business Rule Management System (BRMS)
 - A system for managing and storing business rules independently from the application source code
 - Improves the quality of automated decisions
- WebSphere Business Events
 - Detects significant events or event patterns that require a response from the business perspective, and provides an appropriate response
 - Improves situational awareness and response

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

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



Operational Decision Manager (2 of 3)

- Operational Decision Manager has three different editions to meet client needs: Standard, Express, and Advanced
- IBM Operational Decision Manager Standard
 - Adapt the decision logic of applications at the pace of business
- IBM Operational Decision Manager Express
 - Same features as Operational Decision Manager Standard but offered as an affordable entry point with license restrictions
- IBM Operational Decision Manager Advanced
 - Decision Server Insights included
 - Capture events, build contexts, and apply them to operational decisions in real time
 - Detect situations as they occur, whether they are risks or opportunities, and enable action

© Copyright IBM Corporation 2016

Figure 1-27. Operational Decision Manager (2 of 3)

WB3951.2

Notes:

IBM Operational Decision Manager (ODM) has three editions to meet client needs: IBM Operational Decision Manager Standard, IBM Operational Decision Manager Express, and IBM Operational Decision Manager Advanced.

IBM Operational Decision Manager Standard gives organizations the ability to adapt applications in a business-compliant time frame. It also provides visibility into, control over, and automation of point-in-time business decisions.

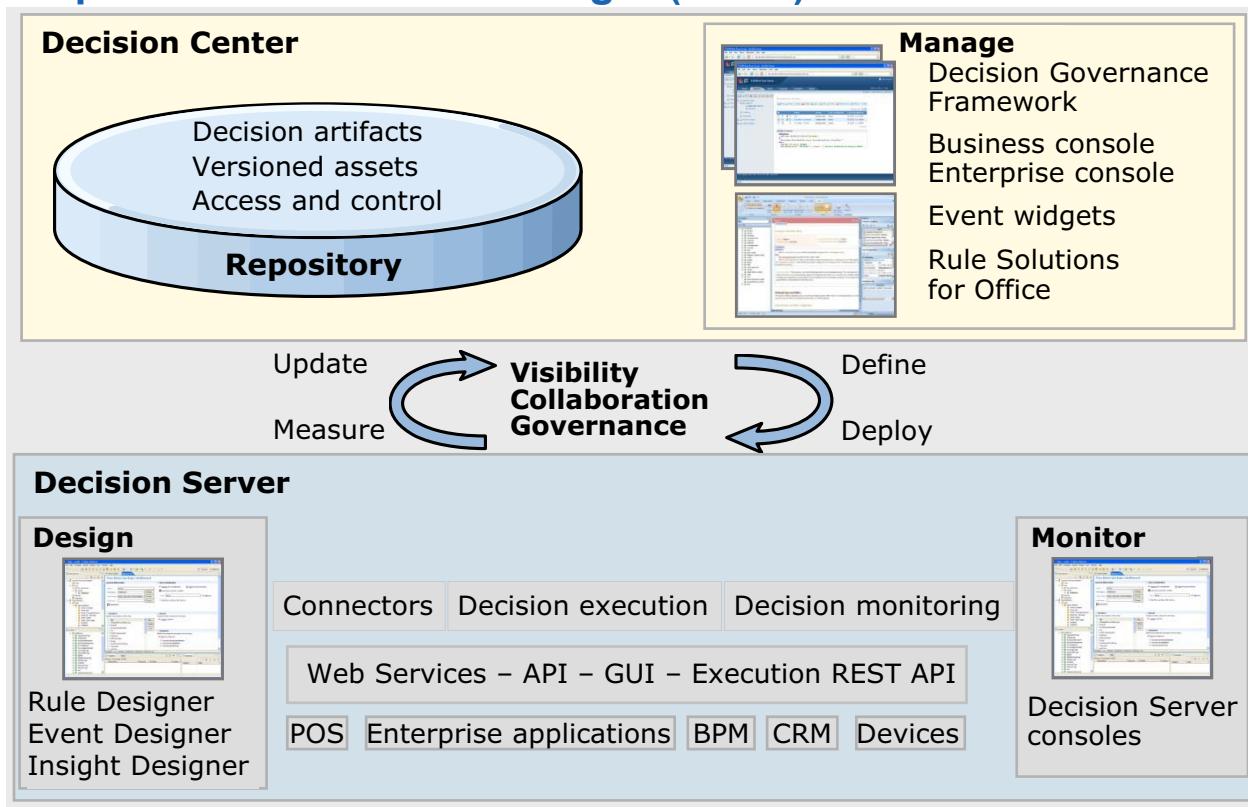
IBM Operational Decision Manager Express offers the same functionality as Standard, but with licensing restrictions that make it an affordable entry point into rules-based decision management capabilities.

IBM Operational Decision Manager Advanced, the newest addition to the Operational Decision Management portfolio, allows organizations to take advantage of the data available to them to enrich and improve their business decisions. With real-time, actionable insight capabilities, companies can now bring together data from multiple sources to identify meaningful patterns and trends that can be applied to operational decisions. As a result, an organization can create and shape business moments by automating decisions in context.

For more information about the IBM Operational Decision Manager packaging, see the product documentation.



Operational Decision Manager (3 of 3)



© Copyright IBM Corporation 2016

Figure 1-28. Operational Decision Manager (3 of 3)

WB3951.2

Notes:

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

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

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

Decision artifacts for (both rules and event logic) are stored in a centralized *repository* with version control, release management, and secure access.

Decision Center also comes with various user interfaces:

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

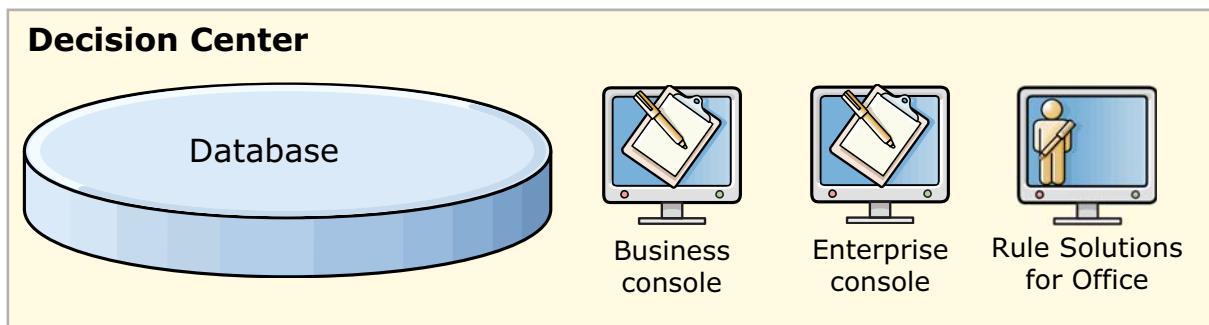
- Rule Solutions for Office, which provides offline authoring and review features in Microsoft Office documents.

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

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

Decision Center: Authoring and maintaining business rules

- Decision Center is the main authoring and management environment for business users
 - Decision Center Business console
 - Decision Center Enterprise console
 - Rule Solutions for Office
 - Decision Center database



© Copyright IBM Corporation 2016

Figure 1-29. Decision Center: Authoring and maintaining business rules

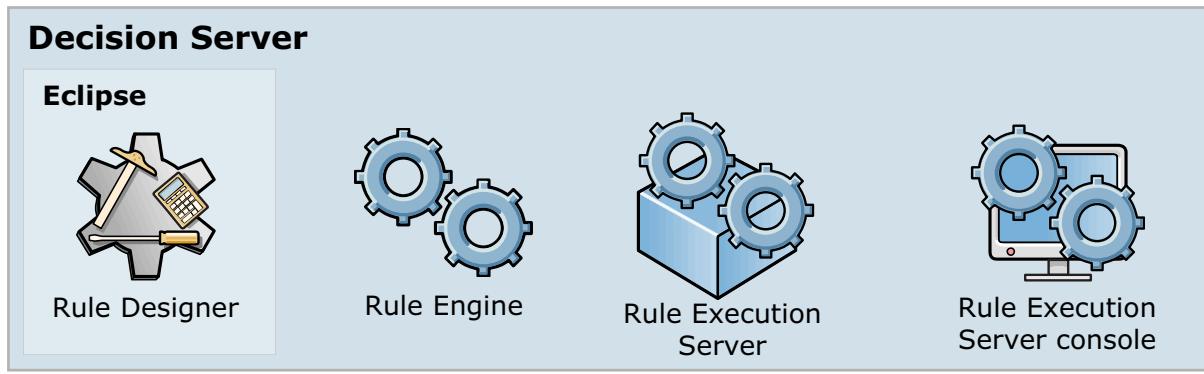
WB3951.2

Notes:

Decision Center encompasses the modules that are shown here.

Decision Server: Development and execution

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



© Copyright IBM Corporation 2016

Figure 1-30. Decision Server: Development and execution

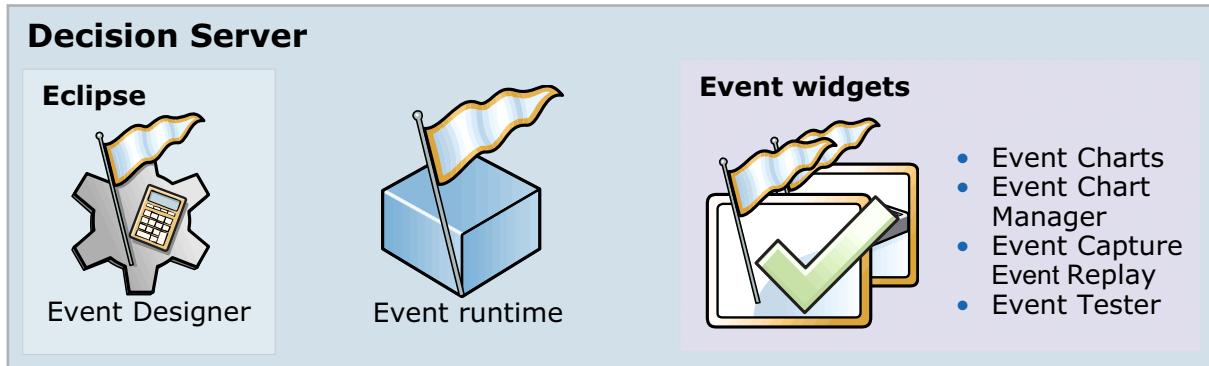
WB3951.2

Notes:

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

Decision Server Events

- Decision Server Events includes the following components:
 - Event Designer
 - Event runtime
 - Event widgets



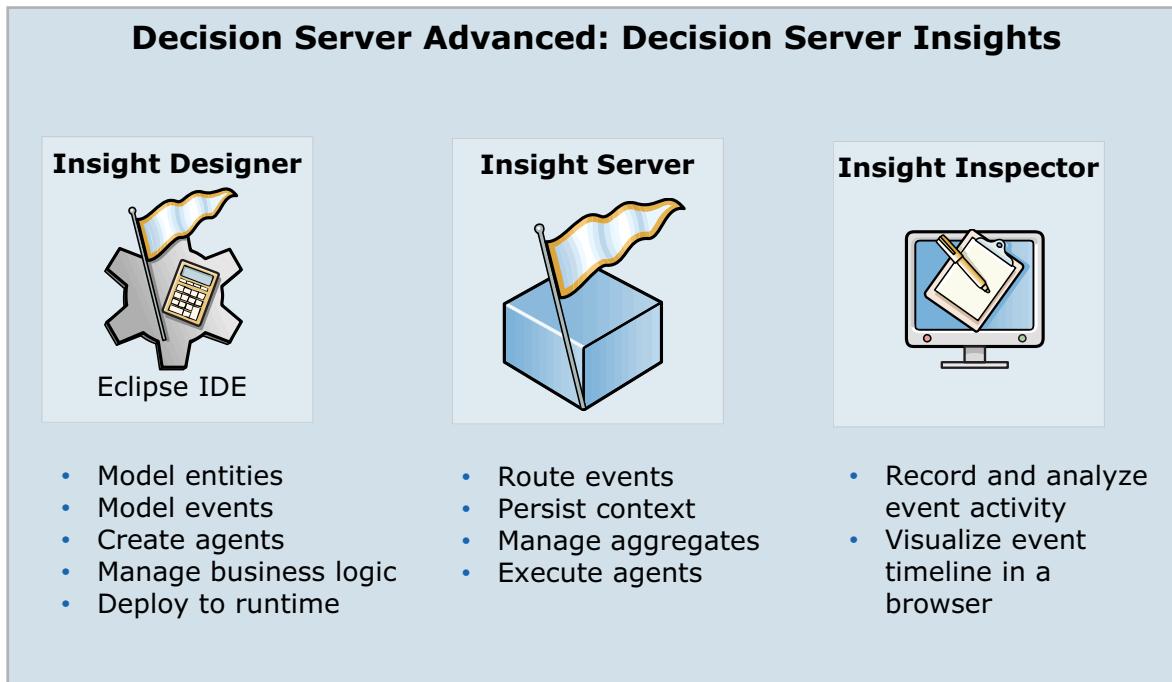
© Copyright IBM Corporation 2016

Figure 1-31. Decision Server Events

WB3951.2

Notes:

Decision Server Insights



© Copyright IBM Corporation 2016

Figure 1-32. Decision Server Insights

WB3951.2

Notes:

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

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

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

Insight Designer offers the following features:

- Eclipse interface to develop rule-based, event-driven solutions
- Develop solutions that capture business models and logic through natural-language editors
- Solutions route events to entities through agents or services and use business rules to process responses
- Solutions include model definition, business rules, and analytics

- Connectivity definitions determine inbound and outbound endpoints to receive and send events between solution and external systems

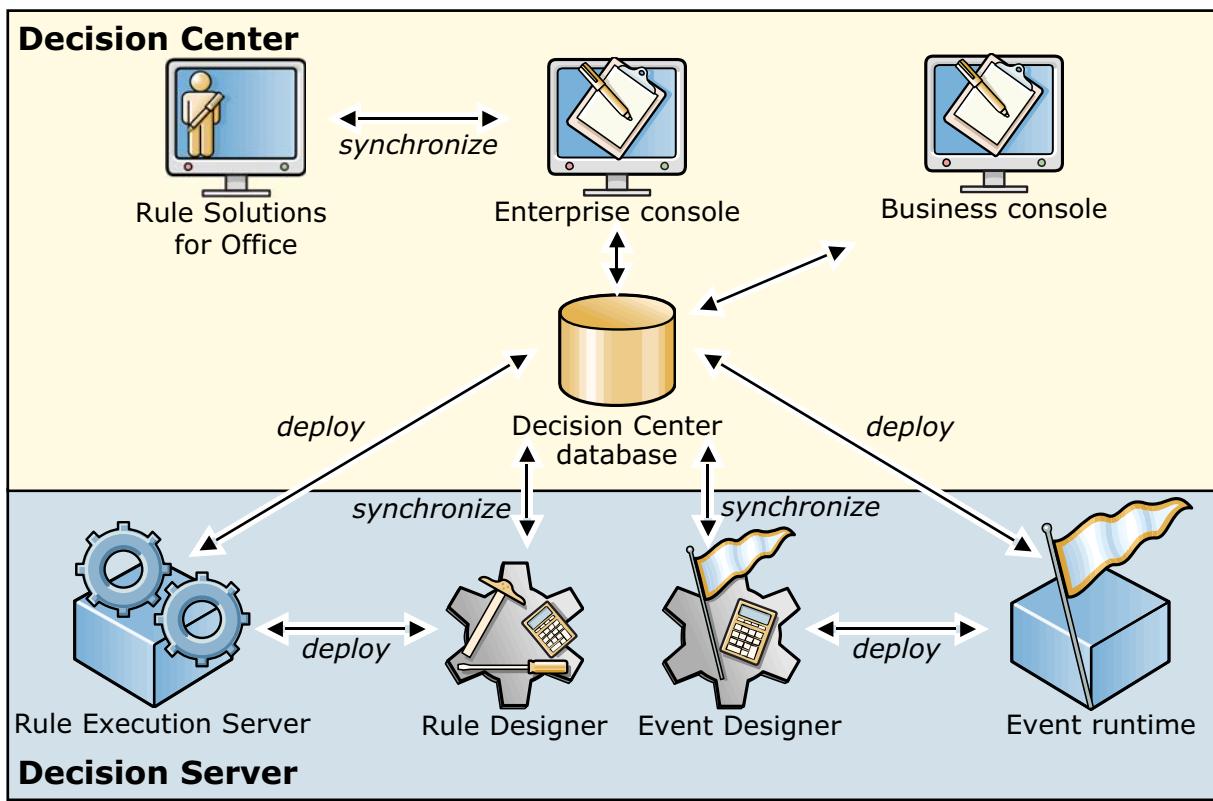
Insight Server Runtime is based on WebSphere Liberty and Extreme Scale:

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

Insights Inspector provides the capability to:

- Visualize timeline of event activity for entities

Operational Decision Manager components



© Copyright IBM Corporation 2016

Figure 1-33. Operational Decision Manager components

WB3951.2

Notes:

This diagram shows an overview of how the different modules work together through synchronization and deployment.

First look at Decision Server, where application development starts. You can work in either Rule Designer or Event Designer, which are Eclipse-based environments for technical users.

After preparing the rule authoring environment in Designer, developers use the synchronization mechanism in Designer to publish rules to Decision Center, where business users can access and maintain the rules through Decision Center consoles.

Decision Center includes permissions and access control, so multiple users can work on the same project. Business users who prefer to work offline can author or review rules in Rule Solutions for Office, which is a plug-in to Microsoft Office Word and Excel. A synchronization mechanism in Decision Center allows rules to be shared and synchronized between business users who are working in Decision Center consoles and users who are working in Rule Solutions for Office plug-ins.

To make the rules available for execution, the rules are deployed to the Rule Execution Server, which is a managed execution environment that houses the rule engine. Rules can be deployed to

Rule Execution Server from either Designer or Decision Center, which means that even business users can have direct control over deployment.

Business rule application development starts in Designer, which is an Eclipse-based environment for technical users. Based on the business requirements, developers work in Rule Designer or Event Designer to design the project and set up the authoring environment for business users.

This course does not cover Decision Server Insights. For more information about how Decision Server Insights works with the other Operational Decision Manager Advanced modules, see course WB398, *Developing Solutions with IBM Decision Server Insights V8.7.1*.



Operational Decision Manager editions

Advanced

- Decision Server Rules
- Designer Server Events
- Decision Server Insights
- Decision Center

Standard

- Decision Server Rules
- Decision Center

Express

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

© Copyright IBM Corporation 2016

Figure 1-34. Operational Decision Manager editions

WB3951.2

Notes:

Operational Decision Manager (ODM) has three editions to meet client needs:

- Operational Decision Manager Standard includes business rules only. You can use the complete business rules management system (BRMS) from IBM to develop business rule applications.
- Operational Decision Manager Express offers the same functionality as Standard, but with licensing restrictions that make it an affordable entry point into rules-based decision management capabilities.

This edition remains the low-cost-of-ownership BRMS solution of IBM. You can get started with small-sized to medium-sized applications. Operational Decision Manager Express entitles you to use only the business rules capabilities, but not the events capabilities, within a limited configuration. In particular, you cannot use Operational Decision Manager Express in a cluster environment.

- Operational Decision Manager Advanced, the newest addition to the Operational Decision Management portfolio, allows organizations to take advantage of the data available to them to enrich and improve their business decisions. With real-time, actionable insight capabilities,

companies can now bring together data from multiple sources to identify meaningful patterns and trends that can be applied to operational decisions. As a result, an organization can create and shape business moments by automating decisions in context.

This edition provides a full-featured decision management platform that includes Decision Server Advanced. You can address the entire decision automation scope, which includes business rules, business events, and decision insights capabilities.

This course uses Operational Decision Manager Standard.

For more information about the IBM Operational Decision Manager packaging, see the product documentation.



Installable components

- Operational Decision Manager comprises a set of components
 - Can be installed on both distributed operating systems or z/OS
- Each component includes a set of modules
 - Some modules are installed by default; others are optional

Decision Center Business user capability for rules and events
Decision Server Integrated development environments (IDEs) and runtime components for rules and events
Decision Server Rules (subset of Decision Server)
Decision Server Events (subset of Decision Server)
Decision Server Insights Capture events, build contexts, and apply them to operational decisions in real time
WebSphere Application Server profile templates Profile templates for Rules Profile templates for Events
Business Rules Embedded Development environment for adding business rules functionality to applications

© Copyright IBM Corporation 2016

Figure 1-35. Installable components

WB3951.2

Notes:

In addition to the Decision Center and Decision Server components, the Operational Decision Manager installer includes:

- WebSphere Application Server profile templates to facilitate installation for cluster topology.
- Business Rules Embedded, which provides authoring components that support the creating and editing of business rules in Eclipse and Dojo environments. The SDK also provides an API for exporting the resulting rule projects to a full Operational Decision Manager system.

1.6. IBM ODM on Cloud



IBM ODM on Cloud



© Copyright IBM Corporation 2016

Course materials may not be reproduced in whole or in part without the prior written permission of IBM.

10.1

Figure 1-36. IBM ODM on Cloud

WB3951.2

Notes:



Introducing 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
 - As of 2015, over 25 data centers are available worldwide
- Managed by IBM
- Artifacts that are created with IBM ODM on Cloud are compatible with IBM ODM on-premises product

© Copyright IBM Corporation 2016

Figure 1-37. Introducing IBM ODM on Cloud

WB3951.2

Notes:

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

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

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

IBM ODM on Cloud: Three runtime environments

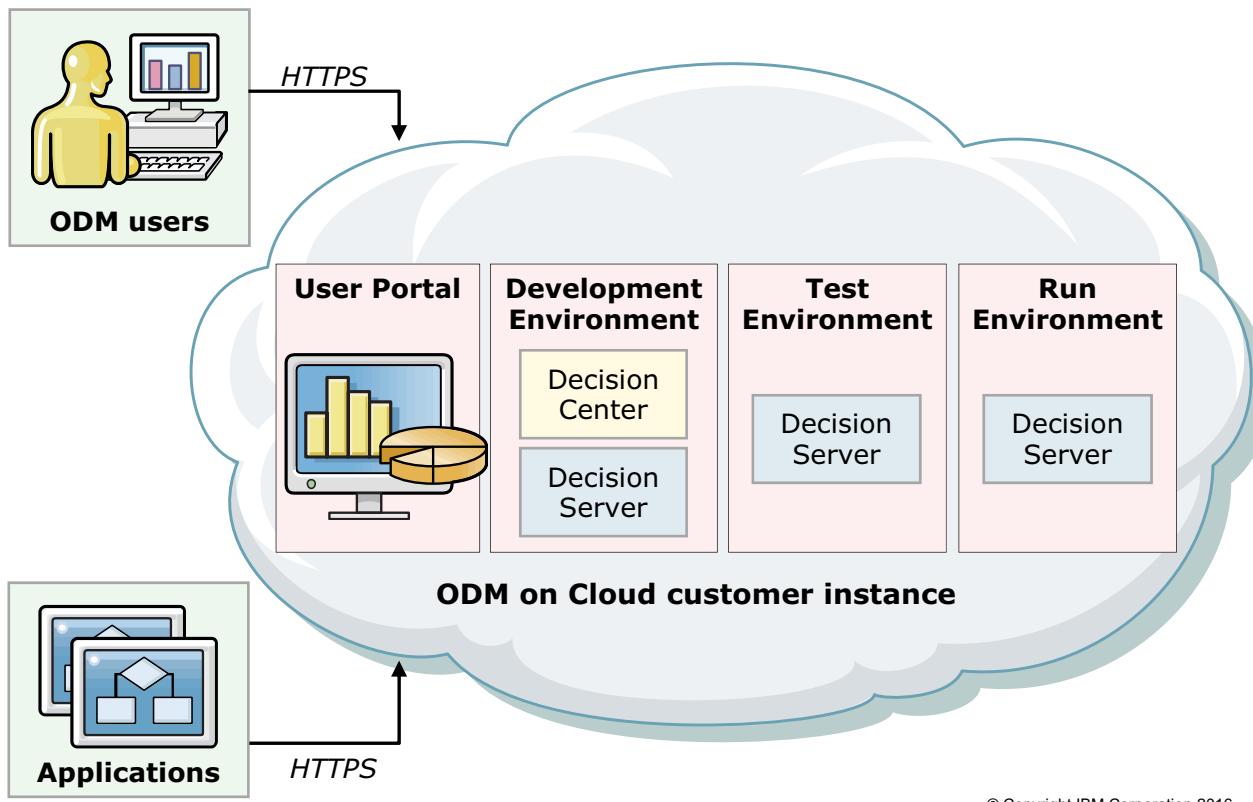


Figure 1-38. IBM ODM on Cloud: Three runtime environments

WB3951.2

Notes:

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

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

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



IBM ODM on Cloud user portal

The screenshot shows the IBM ODM on Cloud user portal interface. At the top, there's a navigation bar with the IBM logo and a search bar. Below the navigation bar, the title "IBM ODM on Cloud user portal" is displayed. The main content area is organized into three horizontal sections: "Development Environment", "Test Environment", and "Production Environment".

- Development Environment:** Contains four items:
 - Rule Designer**: Build the model and technical artifacts that are required for authoring and running a decision service. Includes "Download" and "More info" links.
 - Decision Center Business Console**: Author, test, govern and deploy decision services. Includes "Launch" and "More info" links.
 - Decision Center Enterprise Console**: Perform occasional administration or advanced management activities. Includes "Launch" and "More info" links.
 - Rule Execution Server console**: Manage and monitor decision services deployed from Rule Designer or Decision Center as part of your development activities. Includes "Launch" and "More info" links.
- Test Environment:** Contains one item:
 - Rule Execution Server console**: Monitor decision services deployed from Decision Center used for performance, system, and integration testing activities. Includes "Launch" and "More info" links.
- Production Environment:** Contains one item:
 - Rule Execution Server console**: Monitor decision services deployed from Decision Center used in the context of a production application. Includes "Launch" and "More info" links.

© Copyright IBM Corporation 2016

Figure 1-39. IBM ODM on Cloud user portal

WB3951.2

Notes:

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.

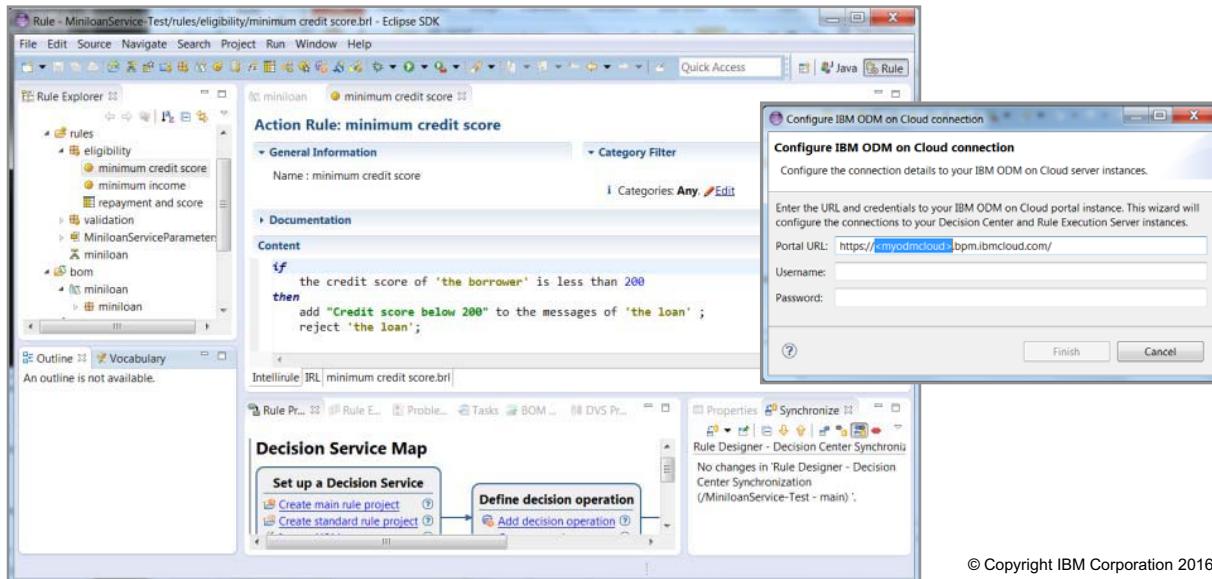
You can start the Decision Center consoles (Business console and Enterprise console) and the Rule Execution server console, from the user portal. These modules run in the cloud environment.

Rule Designer is available for download from the user portal, and is run locally on your own computer. This version of Rule Designer is configured to work specifically with ODM on Cloud.



ODM on Cloud: Rule Designer

- Download the installation file from the user portal
- Double-click the `eclipse.exe` file
- Configure Rule Designer to work with ODM on Cloud
- Has the same basic interface as the on-premises version



© Copyright IBM Corporation 2016

Figure 1-40. ODM on Cloud: Rule Designer

WB3951.2

Notes:

To work with Rule Designer in ODM on Cloud, first download the installation file. Rule Designer is run locally on your computer, instead of in the cloud environment.

After the download is complete, double-click the `eclipse.exe` file to start Rule Designer.

The next step is to configure Rule Designer to connect to your ODM on Cloud environment.

After you connect Rule Designer to ODM on Cloud, you can use it to import and create rule projects. You can publish your rule projects to the cloud for use with Decision Center and Rule Execution Server. The ODM on Cloud Rule Designer offers the same basic interface as the on-premises version, but some options might be different.



ODM on Cloud: Decision Center and Rule Execution Server

- In user portal, click **Launch** under the console that you want to open
- Console opens in the web browser
 - Has the same basic interface as the on-premises version of ODM

© Copyright IBM Corporation 2016

Figure 1-41. ODM on Cloud: Decision Center and Rule Execution Server

WB3951.2

Notes:

To work with the Decision Center consoles (Business console and Enterprise console) or the Rule Execution Server console, identify the console that you want to start in the user portal, and click **Launch**.

The next step is to log in with your ODM on Cloud user name and password.

After you log in, the console opens directly in the web browser window. The ODM on Cloud modules offer the same basic interface as the on-premises version, but some options might be different.

1.7. Integration with IBM product family

Integration with IBM product family



© Copyright IBM Corporation 2016

Course materials may not be reproduced in whole or in part without the prior written permission of IBM.

10.1

Figure 1-42. Integration with IBM product family

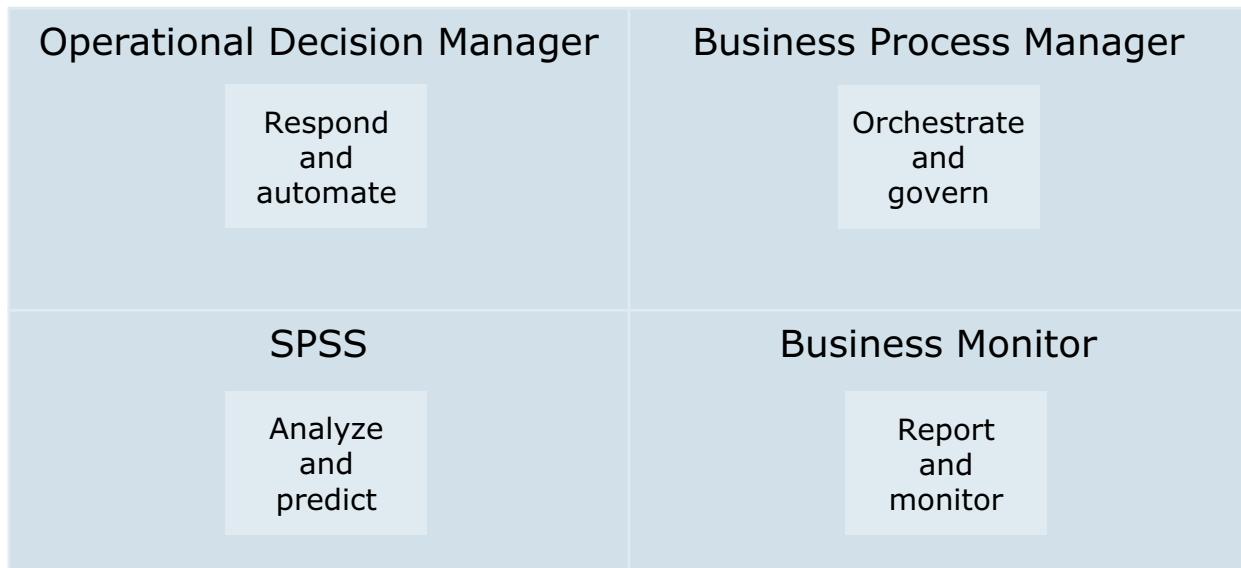
WB3951.2

Notes:



Synergies with other IBM products

- Operational Decision Manager complements other IBM products to empower businesses to automate, manage, and improve the decision cycle



© Copyright IBM Corporation 2016

Figure 1-43. Synergies with other IBM products

WB3951.2

Notes:

If you are already familiar with IBM offerings, you probably recognize these products that can be integrated with Operational Decision Management. For more information about how these products complement each other, see the IBM Knowledge Centers for these products.

IBM offerings include the following products.

- For operational decision management:
 - Operational Decision Manager
 - Operational Decision Manager for z/OS
- For analytic decision management:
 - SPSS Decision Management
 - Integration with IBM SPSS provides business predictive analytics for continuous decision improvement.
 - Allows rule-based decision services to use analytic models, for example, in determining fraud or risk, or best offers.

- Business Process Manager orchestrates decision services within processes.
 - IBM Process Designer and Integration Designer include wizards to easily integrate existing business rule applications into business processes.
- Integration with IBM Business Monitor provides real-time monitoring of business decisions for decision improvement.

1.8. Operational Decision Manager roles

Operational Decision Manager roles



© Copyright IBM Corporation 2016

Course materials may not be reproduced in whole or in part without the prior written permission of IBM.

10.1

Figure 1-44. Operational Decision Manager roles

WB3951.2

Notes:

Operational Decision Manager user roles

Business users	
	Business analyst Bridges between the business side and technical side of a business rule application
	Policy manager Business expert and owner of business policy
	Rule author Business domain expert who updates and reviews rules
Technical users	
	Architect Manages overall deployment, organization of rules, and optimization of rule execution
	Developer Develops, tests, and deploys business rule applications and event applications
	Administrator Installs and configures rule management and execution environments

© Copyright IBM Corporation 2016

Figure 1-45. Operational Decision Manager user roles

WB3951.2

Notes:

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

© Copyright IBM Corporation 2016

Figure 1-46. Interaction between roles

WB3951.2

Notes:

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 and event rules so that rule authors can maintain them
 - Validating that rule execution yields the expected results
- Tools: Designer and Decision Center



Business analyst

© Copyright IBM Corporation 2016

Figure 1-47. Business analyst

WB3951.2

Notes:

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

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

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

Policy manager

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



Policy manager

© Copyright IBM Corporation 2016

Figure 1-48. Policy manager

WB3951.2

Notes:

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

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

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

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

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



Rule author

- Responsibilities:
 - Updating and sometimes creating rules
 - Reviewing the business rules and event rules by using queries and reports
- Tools: Decision Center Business and Enterprise Consoles, and Rule Solutions for Office



Rule author

© Copyright IBM Corporation 2016

Figure 1-49. Rule author

WB3951.2

Notes:

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

Architect

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



Architect

© Copyright IBM Corporation 2016

Figure 1-50. Architect

WB3951.2

Notes:

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

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

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

Developer

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



Developer

© Copyright IBM Corporation 2016

Figure 1-51. Developer

WB3951.2

Notes:

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

Developers work in Designer to do these tasks:

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

Administrator

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



Administrator

© Copyright IBM Corporation 2016

Figure 1-52. Administrator

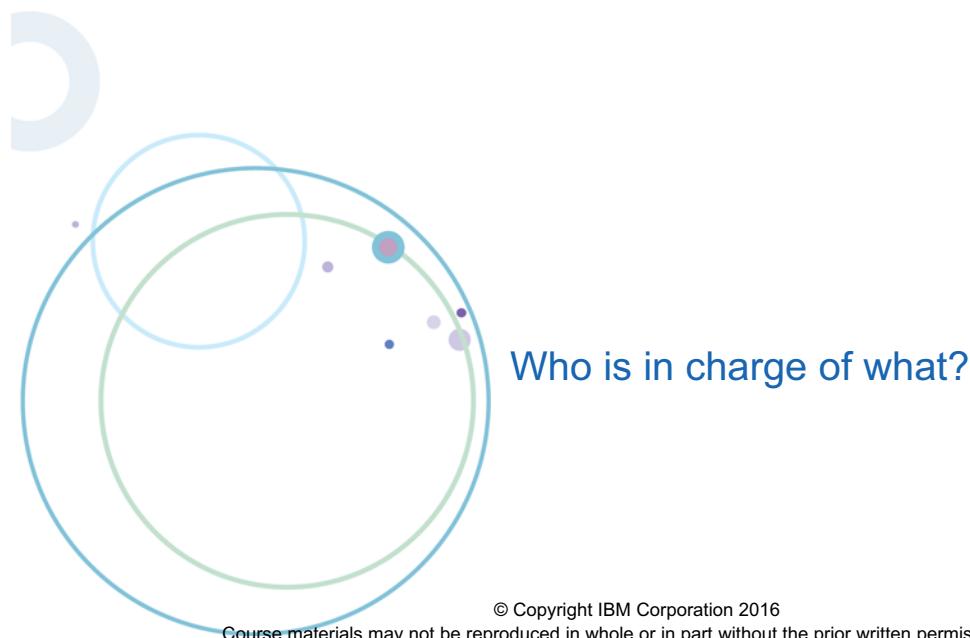
WB3951.2

Notes:

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



© Copyright IBM Corporation 2016

Course materials may not be reproduced in whole or in part without the prior written permission of IBM.

10.1

Figure 1-53. Discussion

WB3951.2

Notes:

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.

Be prepared to share your ideas with the class.

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

2. Who is in charge of authoring rules?

3. Who is in charge of deploying rules?

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

1.9. Governance and decision management

Governance and decision management



© Copyright IBM Corporation 2016

Course materials may not be reproduced in whole or in part without the prior written permission of IBM.

10.1

Figure 1-54. Governance and decision management

WB3951.2

Notes:

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

© Copyright IBM Corporation 2016

Figure 1-55. Applying governance to decision management

WB3951.2

Notes:

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 actions must be taken in response to the various business events, and which requirements form the basis of decisions. Modeling makes it easier for the business team to understand the business logic, and how it can be implemented, which makes the system more maintainable.

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

Applying governance to decision management encompasses people, processes, and goals across your organization. By defining expectations, assigning responsibilities, and verifying performance, governance provides an organizational framework that prevents potential problems. It facilitates

efficient collaboration between business and IT teams, and makes it possible for an organization to demonstrate that it accomplished what it said it would accomplish.

Governance in Operational Decision Manager

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

© Copyright IBM Corporation 2016

Figure 1-56. Governance in Operational Decision Manager

WB3951.2

Notes:

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

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

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

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

1.10. Working with Decision Server Rules and Decision Center

Working with Decision Server Rules and Decision Center



© Copyright IBM Corporation 2016

Course materials may not be reproduced in whole or in part without the prior written permission of IBM.

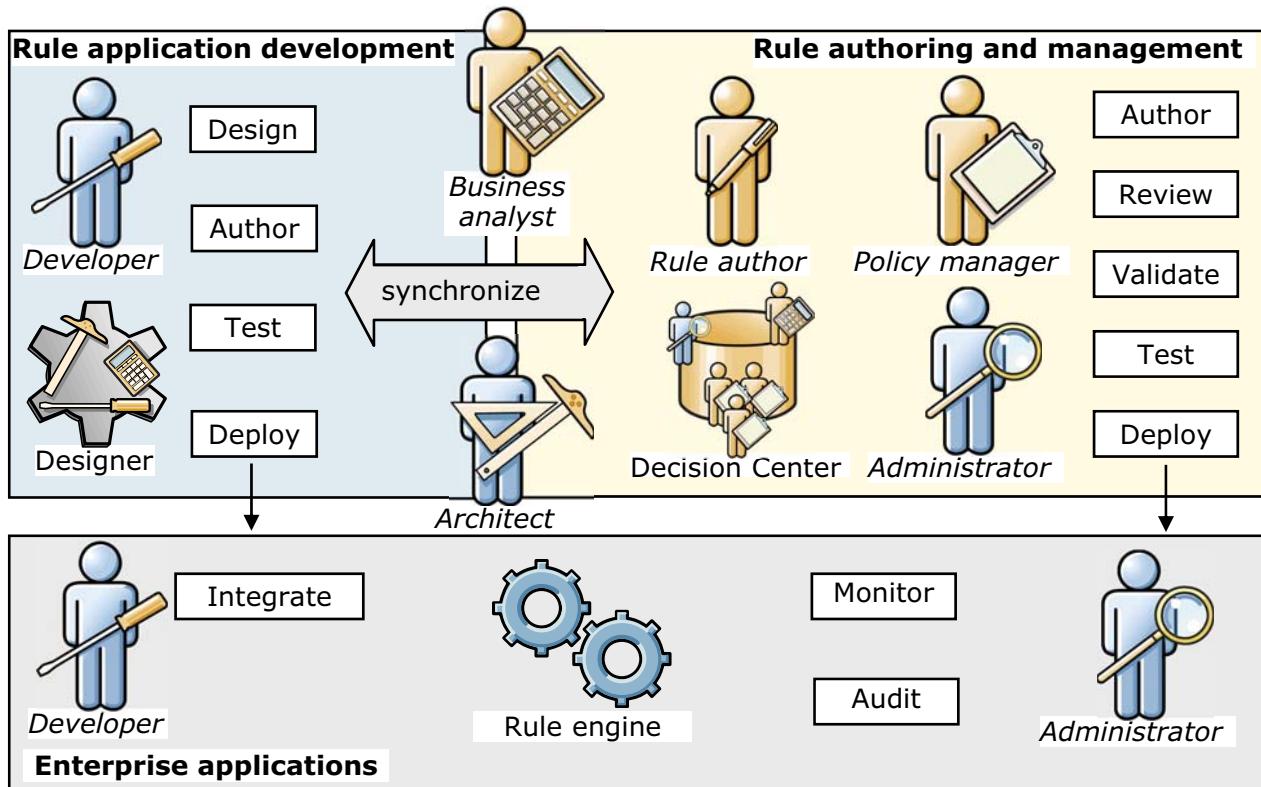
10.1

Figure 1-57. Working with Decision Server Rules and Decision Center

WB3951.2

Notes:

Operational Decision Manager roles and activities



© Copyright IBM Corporation 2016

Figure 1-58. Operational Decision Manager roles and activities

WB3951.2

Notes:

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

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

Operational Decision Manager has three main environments:

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

Next, you learn more about these modules.



Unit summary

Having completed this unit, you should be able to:

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

© Copyright IBM Corporation 2016

Figure 1-59. Unit summary

WB3951.2

Notes:

Checkpoint questions

1. **True or False:** Operational decision management combines business expertise with technology to automate repeated and repeatable decisions.
2. **True or False:** Because application lifecycles and decision lifecycles are usually about the same length of time, by using Operational Decision Manager, developers can update the rules every six months when the application infrastructure is updated.
3. Operational Decision Manager includes which components:
 - a. Decision Server Rules
 - b. Decision Server Insights
 - c. Decision Center
 - d. All of the above
4. Which of the following roles are involved during the early stages of a business rule application development project? Select all that apply.
 - a. Policy manager
 - b. Business analyst
 - c. Architect
 - d. Developer

© Copyright IBM Corporation 2016

Figure 1-60. Checkpoint questions

WB3951.2

Notes:

Write your answers here:

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



Checkpoint answers

1. **True.**
2. **False:** *Business policies evolve more rapidly than the application infrastructure. By using Operational Decision Manager, you can manage the decision lifecycle and the application infrastructure lifecycle asynchronously.*
3. Operational Decision Manager includes which components:
 - d. Decision Server Rules, Decision Server Insights, and Decision Center
4. Which of the following roles are involved during the early stages of a decision management solution development project?
 - a. True
 - b. True
 - c. True
 - d. True

© Copyright IBM Corporation 2016

Figure 1-61. Checkpoint answers

WB3951.2

Notes:

Exercise 1



Operational Decision Manager in action

© Copyright IBM Corporation 2016

Course materials may not be reproduced in whole or in part without the prior written permission of IBM.

10.1

Figure 1-62. Exercise 1

WB3951.2

Notes:



Exercise objectives

After completing this exercise, you should be able to:

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

© Copyright IBM Corporation 2016

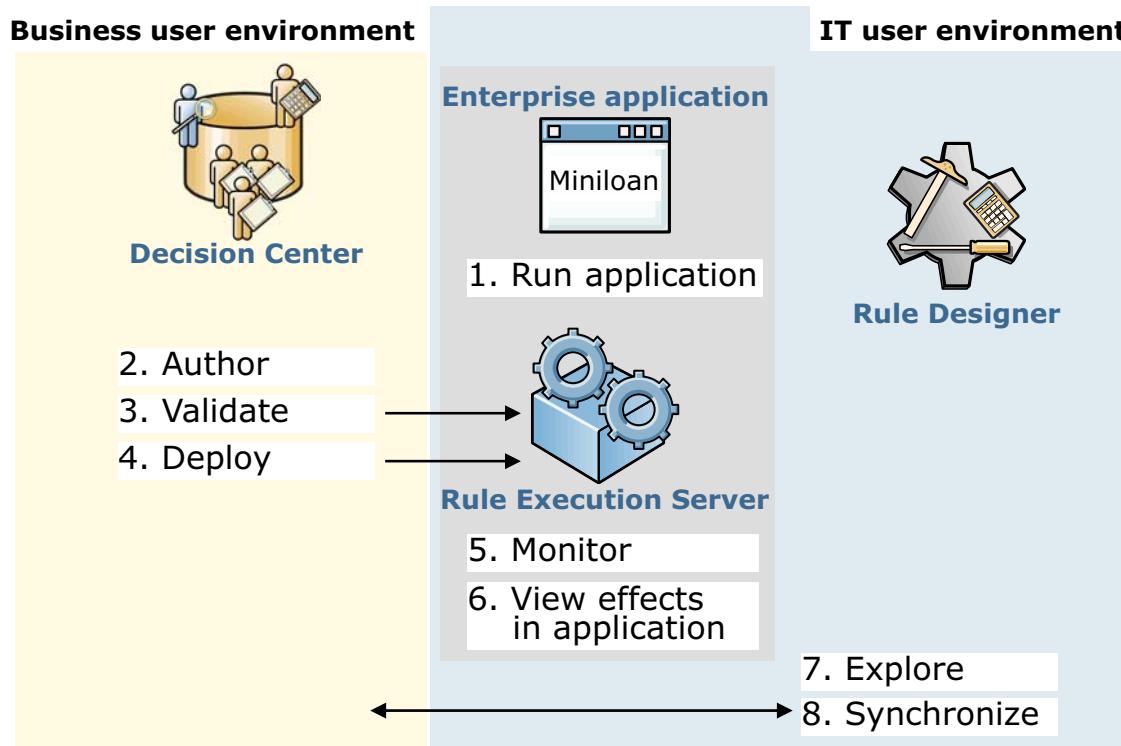
Figure 1-63. Exercise objectives

WB3951.2

Notes:



Exercise workflow



© Copyright IBM Corporation 2016

Figure 1-64. Exercise workflow

WB3951.2

Notes:

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. Operational Decision Manager: Business rules

What this unit is about

This unit teaches you how to develop a business rule application with Operational Decision Manager, and how to get started with decision services.

What you should be able to do

After completing this unit, you should be able to:

- 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

How you will check your progress

- Checkpoint
- Exercise



Unit objectives

After completing this unit, you should be able to:

- 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 2016

Figure 2-1. Unit objectives

WB3951.2

Notes:



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 2016

Figure 2-2. Topics

WB3951.2

Notes:

2.1. Using an agile development approach

Using an agile development approach



© Copyright IBM Corporation 2016

Course materials may not be reproduced in whole or in part without the prior written permission of IBM.

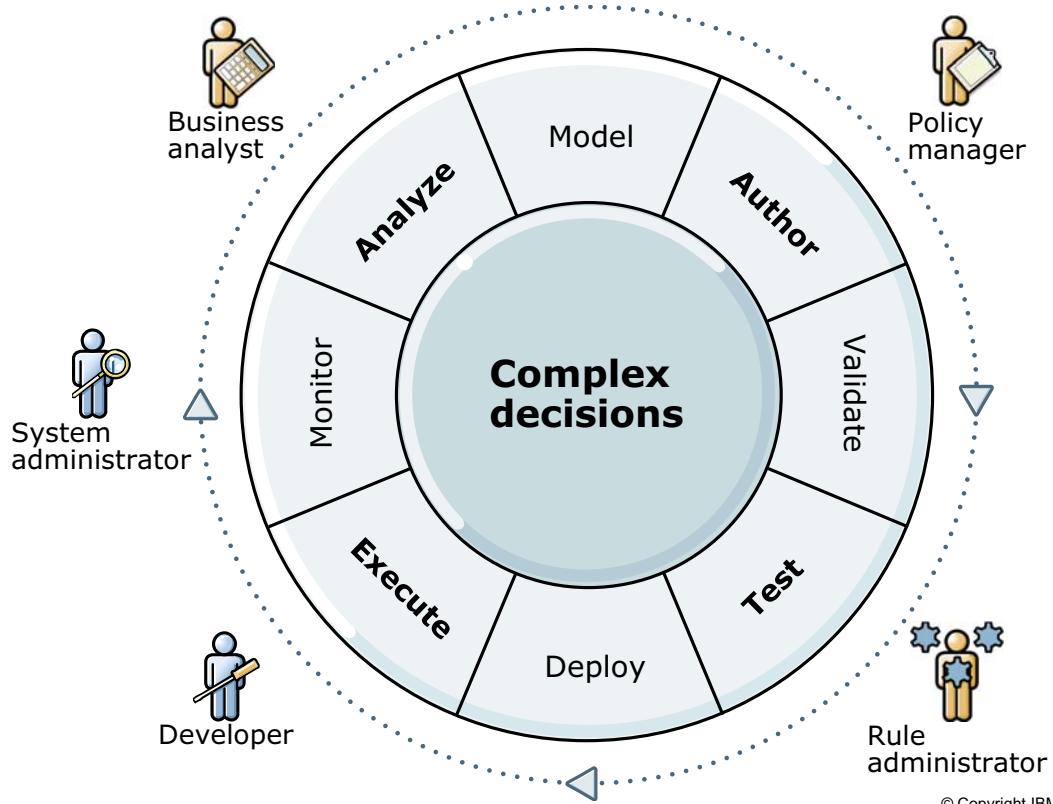
10.1

Figure 2-3. Using an agile development approach

WB3951.2

Notes:

Lifecycle of business decisions



© Copyright IBM Corporation 2016

Figure 2-4. Lifecycle of business decisions

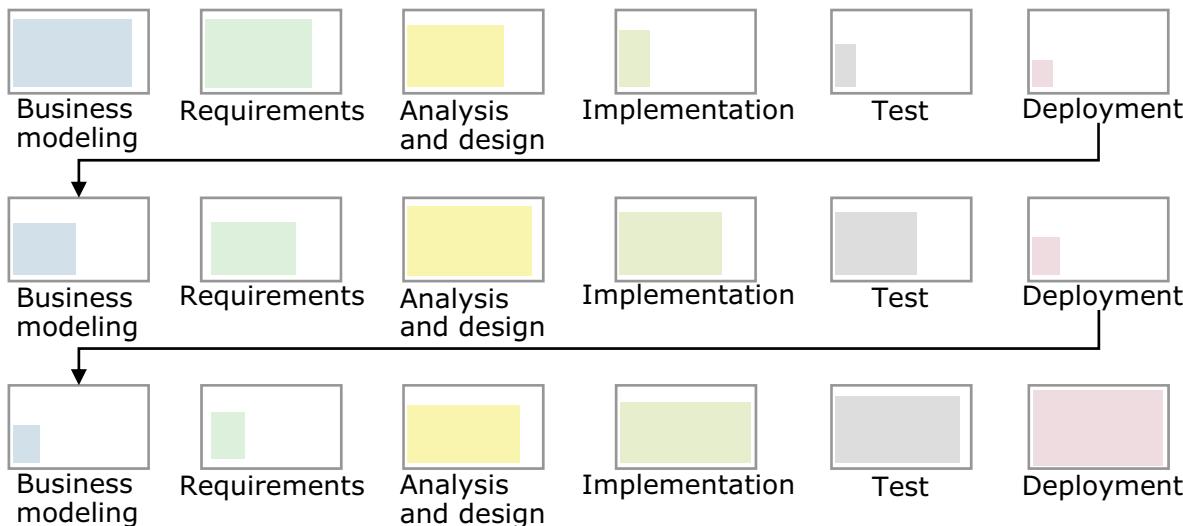
WB3951.2

Notes:

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

Agile Business Rule Development methodology

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



© Copyright IBM Corporation 2016

Figure 2-5. Agile Business Rule Development methodology

WB3951.2

Notes:

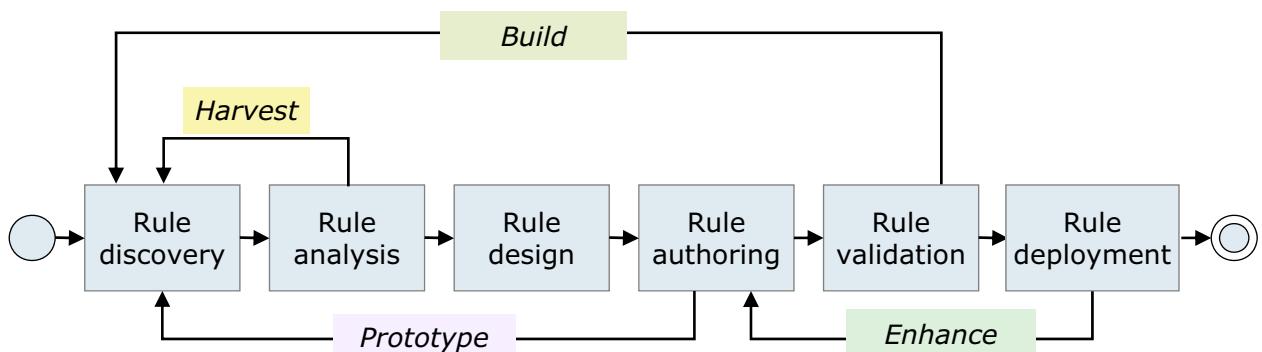
The overall implementation of decision management should follow an agile and iterative approach that responds easily to change.

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

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

Agile development approach

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



© Copyright IBM Corporation 2016

Figure 2-6. Agile development approach

WB3951 2

Notes:

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

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

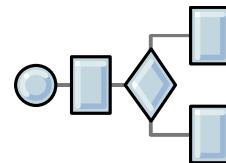
During the discovery and analysis phases, developers can already start working on the *design* of the application, which includes writing code to implement business object models. Developers start setting up the rule authoring environment in Rule Designer according to the early results of discovery,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

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

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).

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

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



- Roles: Architects, business analysts, and policy managers



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



© Copyright IBM Corporation 2016

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

WB3951.2

Notes:

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

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

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

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

© Copyright IBM Corporation 2016

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

WB3951.2

Notes:

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

© Copyright IBM Corporation 2016

Figure 2-9. Discovery: Identifying the rules

WB3951.2

Notes:

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:

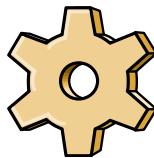
- Did the person provide all the required personal information?
- Is the information valid?
- How much is the client eligible to borrow?
- Is the client able to repay the loan?

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

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**

- Formalize vocabulary as a conceptual object model

Customer

-first name
-last name
-category {Gold, Platinum}
-shopping cart

1.. *

Shopping Cart

-items
-value

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

© Copyright IBM Corporation 2016

Figure 2-10. Discovery: Identifying vocabulary

WB3951.2

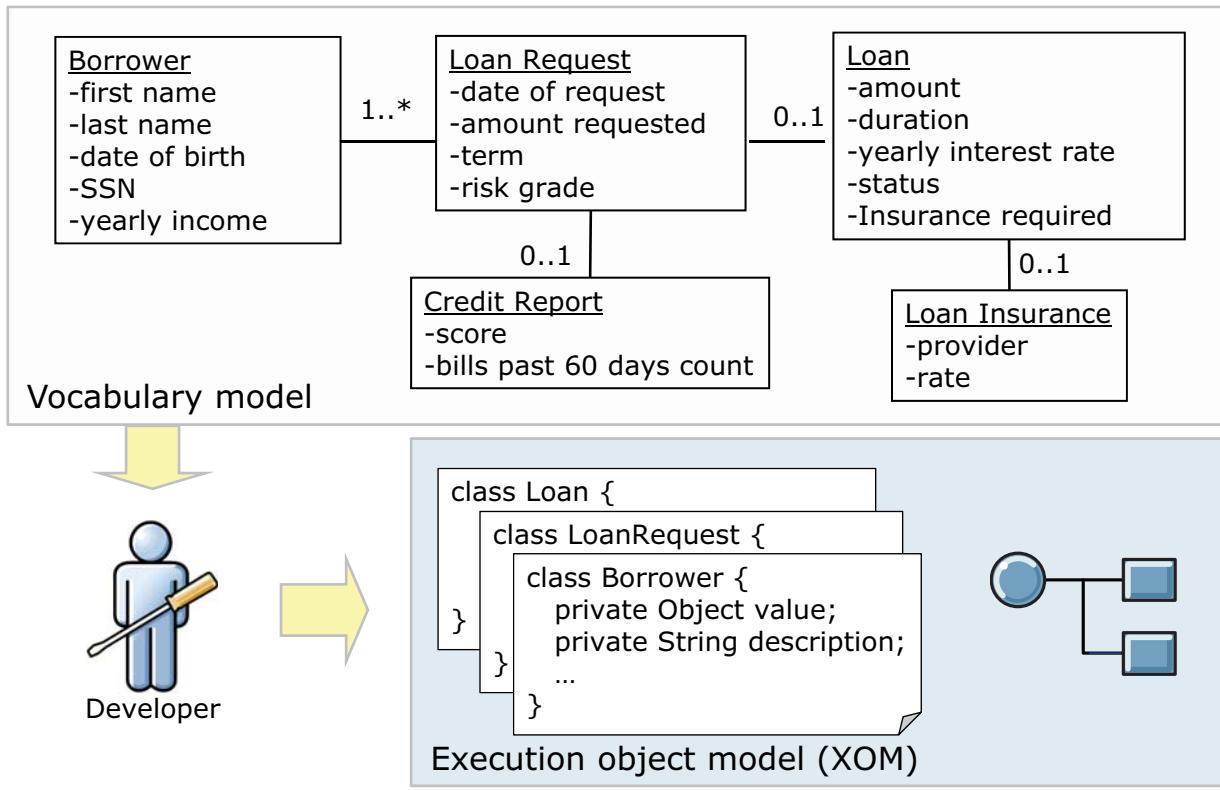
Notes:

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

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

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

Implementing the model



© Copyright IBM Corporation 2016

Figure 2-11. Implementing the model

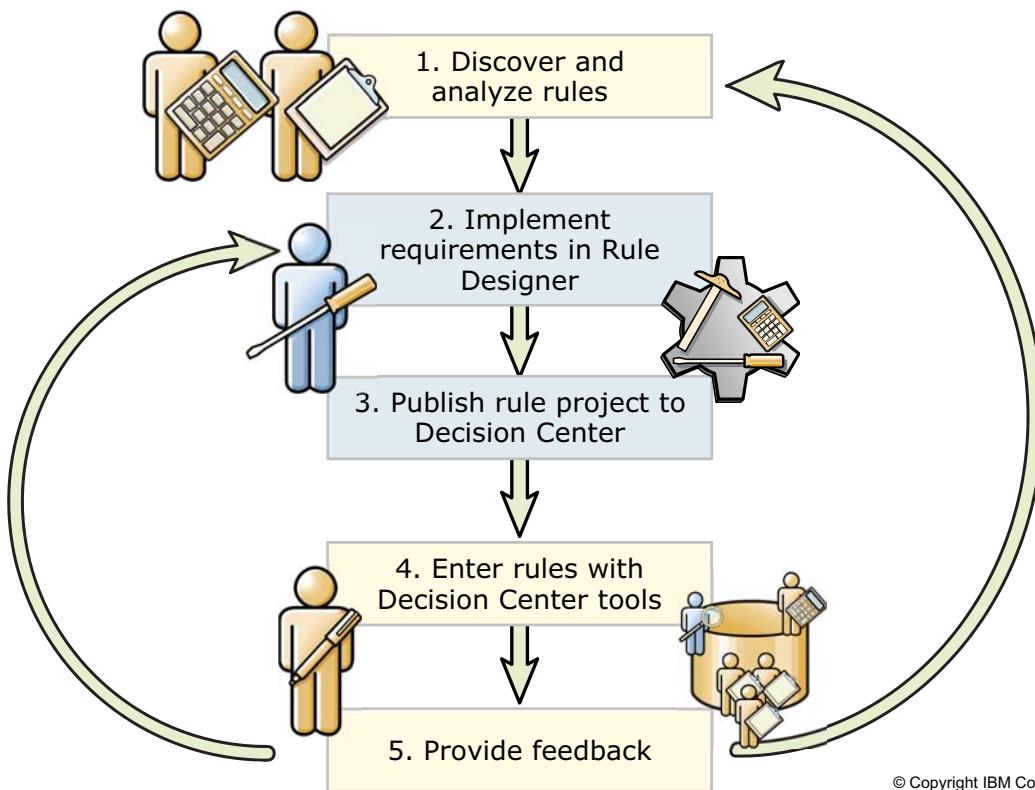
WB3951.2

Notes:

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

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

Using an agile development approach



© Copyright IBM Corporation 2016

Figure 2-12. Using an agile development approach

WB3951.2

Notes:

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

The following development tasks are illustrated on this slide:

1. Early results from rule discovery, including vocabulary models and initial rules, are passed to the developers as requirements.
2. Based on these requirements and discussion with the business analyst, developers work in Rule Designer to design and set up the rule authoring environment. As a first step, developers prepare a *rule project*, which is used as a container for the newly discovered rules and vocabulary.
3. When the rule project is ready, it is published to Decision Center so that business users can access it through the Decision Center consoles. Both developers and business users can

continue to work simultaneously on the same project but in their separate environments on separate tasks.

4. As rule authors begin entering rules in Decision Center, they might find problems or discover new requirements.
5. The rules might look good on paper, but trying to implement them in the tool can uncover possible design or analysis issues.

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

2.2. Designing the business rule application

Designing the business rule application



© Copyright IBM Corporation 2016

Course materials may not be reproduced in whole or in part without the prior written permission of IBM.

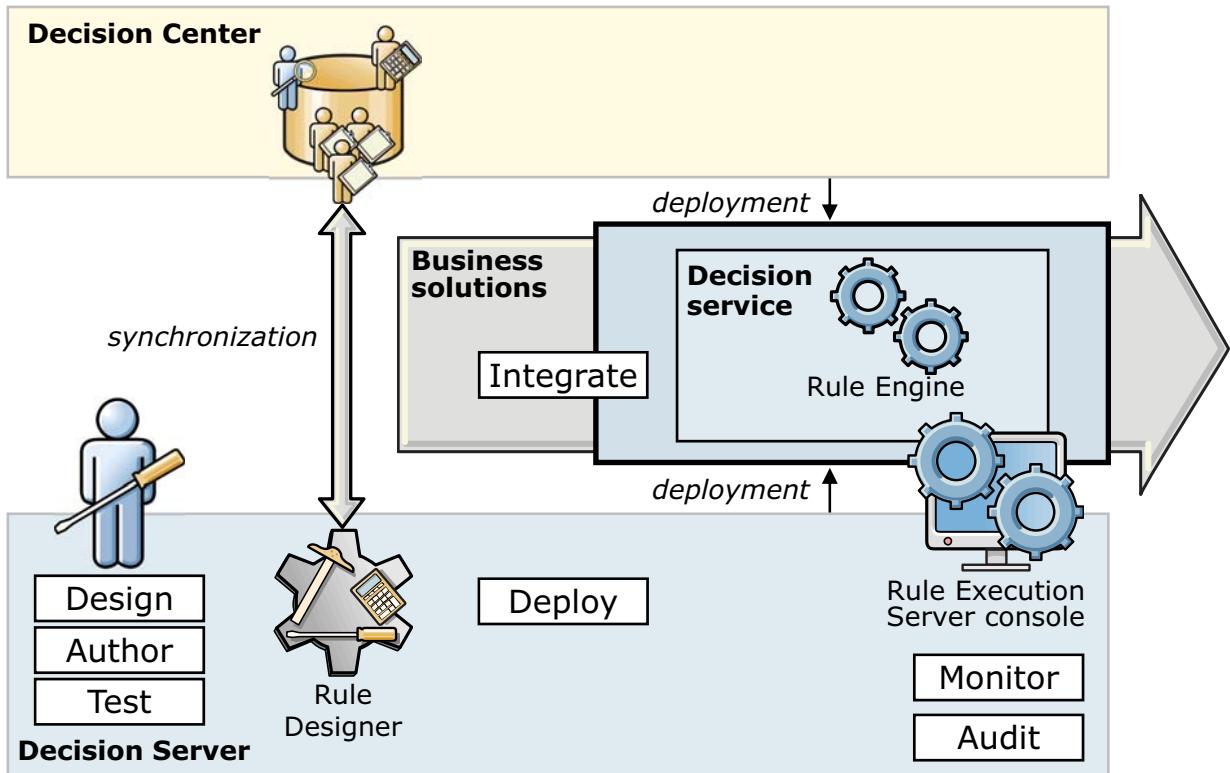
10.1

Figure 2-13. Designing the business rule application

WB3951.2

Notes:

Overview of developer tasks



© Copyright IBM Corporation 2016

Figure 2-14. Overview of developer tasks

WB3951.2

Notes:

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

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

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

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

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

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

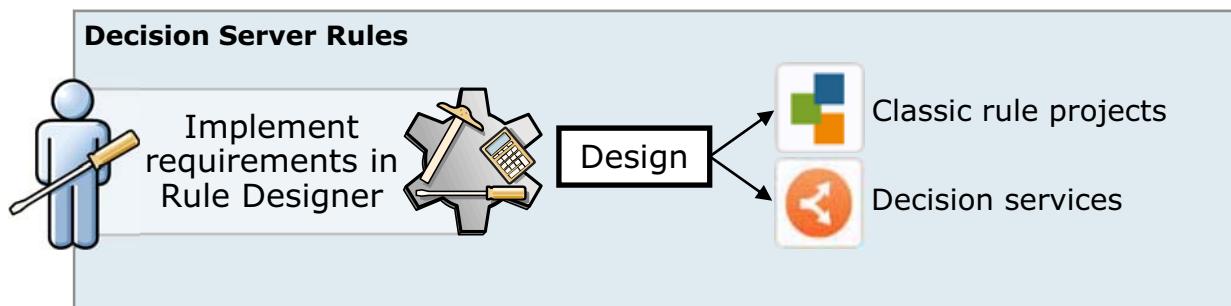
In Rule Designer, you design the granularity of your decision services and the contract between client applications and the decision service. You also design the model and vocabulary for authoring business rules.



Note

The remaining tasks are described later in this course.

Starting point for development: Design



- Design: Starts with rule projects and object models
- Two approaches:
 - Classic rule projects
 - Decision services
- **Note:** For this course, you focus on decision services

© Copyright IBM Corporation 2016

Figure 2-15. Starting point for development: Design

WB3951.2

Notes:

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

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



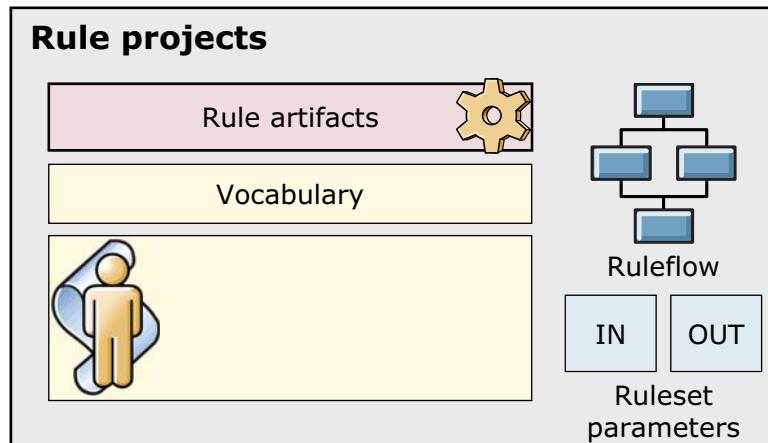
Note

IBM ODM on Cloud

ODM on Cloud does not support classic rule projects.

Rule projects (1 of 2)

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



© Copyright IBM Corporation 2016

Figure 2-16. Rule projects (1 of 2)

WB3951.2

Notes:

A decision service is a set of related *rule projects*. A rule project is a type of Eclipse project that is dedicated to the development of business rule applications.

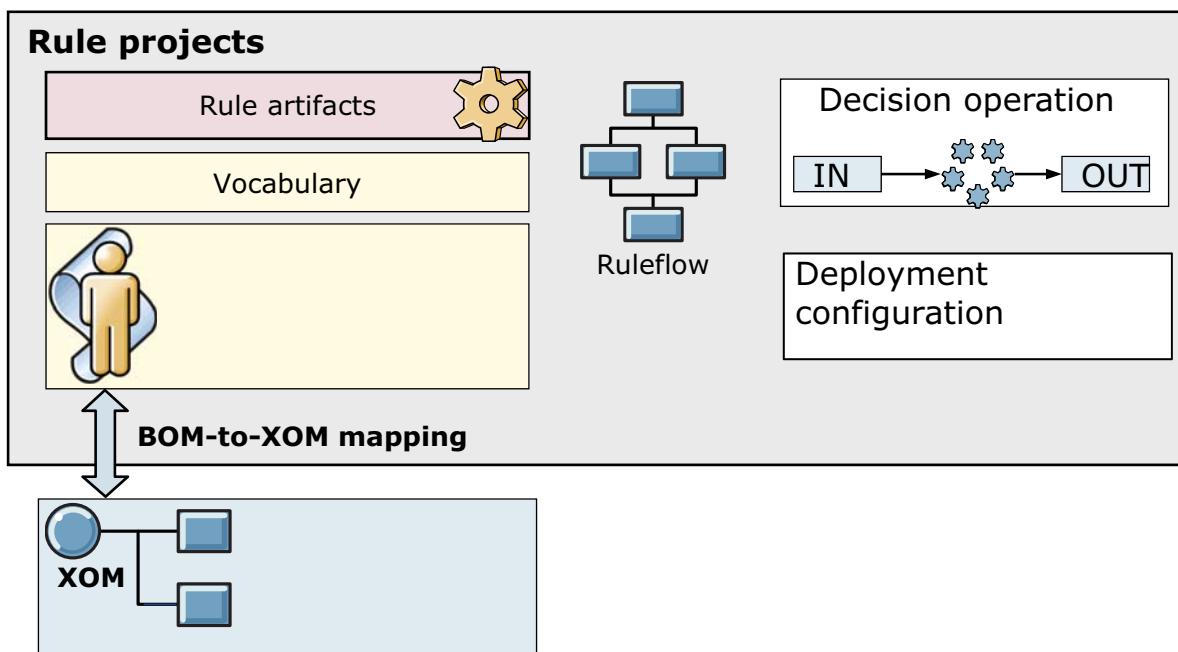
In a rule project, you define a structure for your rule artifacts and set up the business object model (BOM) and vocabulary for editing the rules.

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

You also define *ruleset parameters* to pass objects from the calling application that the rules are going to be evaluated against. The decision result is returned to the calling application by ruleset parameters. These ruleset parameters describe the format of input that is required to make a decision and how the decision results should be returned. These parameters are also called the *ruleset signature*.

Rule projects (2 of 2)

- A ruleset is extracted from the finalized rule project



© Copyright IBM Corporation 2016

Figure 2-17. Rule projects (2 of 2)

WB3951.2

Notes:

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

Each ruleset within the decision service is defined as a *decision operation* with a unique signature of input and output parameters.

- The decision operation also includes ruleset parameters to pass data from the calling application to the ruleset, and to return the decision to the application.
- Multiple decision operations can share a BOM and rule project hierarchy, and be managed and deployed as a single unit.

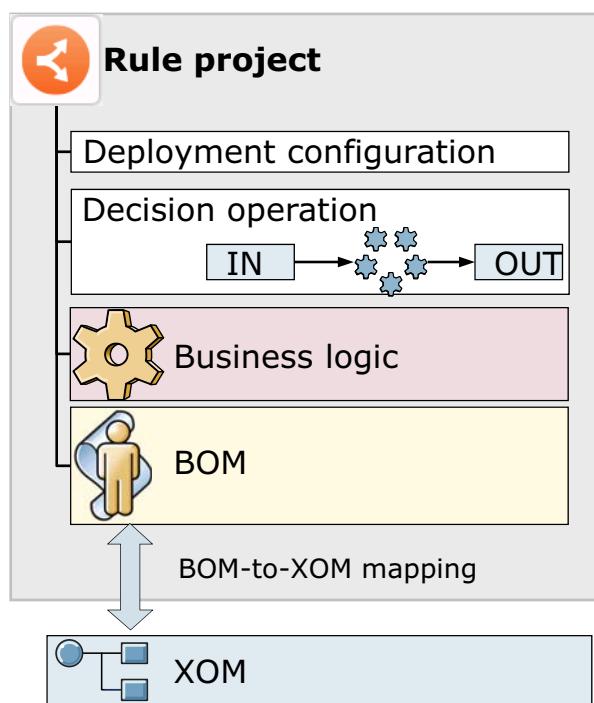
For your client application to use the ruleset, a contract must exist between the application and the ruleset. In the simplest case, this contract consists of two primary elements:

- A mapping from the BOM to the Java objects or XML schemas of your application environment
 - These objects or schemas are called the execution object model (XOM).

- The BOM-to-XOM mapping makes the ruleset (written in terms of the BOM) capable of handling the data that your application manipulates in the form of native Java objects or XML.
- A set of input and output parameters that describe the data that you pass to the ruleset and the data that you expect to receive back as the decision results
 - Through the BOM-to-XOM mapping, these data elements are provided to and from your application in the form of the XOM, but the rules reason on them in terms of the vocabulary and BOM.

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

Decision service rule projects



© Copyright IBM Corporation 2016

Figure 2-18. Decision service rule projects

WB3951.2

Notes:

Decision services incorporate governance features so that all related rule projects are managed and deployed as a single unit.

2.3. Setting up decision services

Setting up decision services



© Copyright IBM Corporation 2016

Course materials may not be reproduced in whole or in part without the prior written permission of IBM.

10.1

Figure 2-19. Setting up decision services

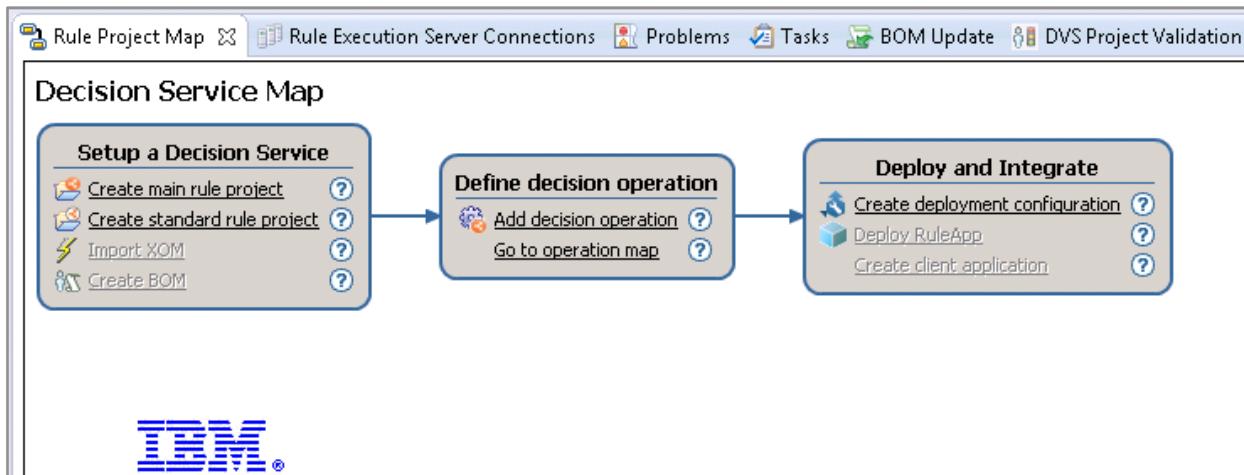
WB3951.2

Notes:



Decision service map

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



© Copyright IBM Corporation 2016

Figure 2-20. Decision service map

WB3951.2

Notes:

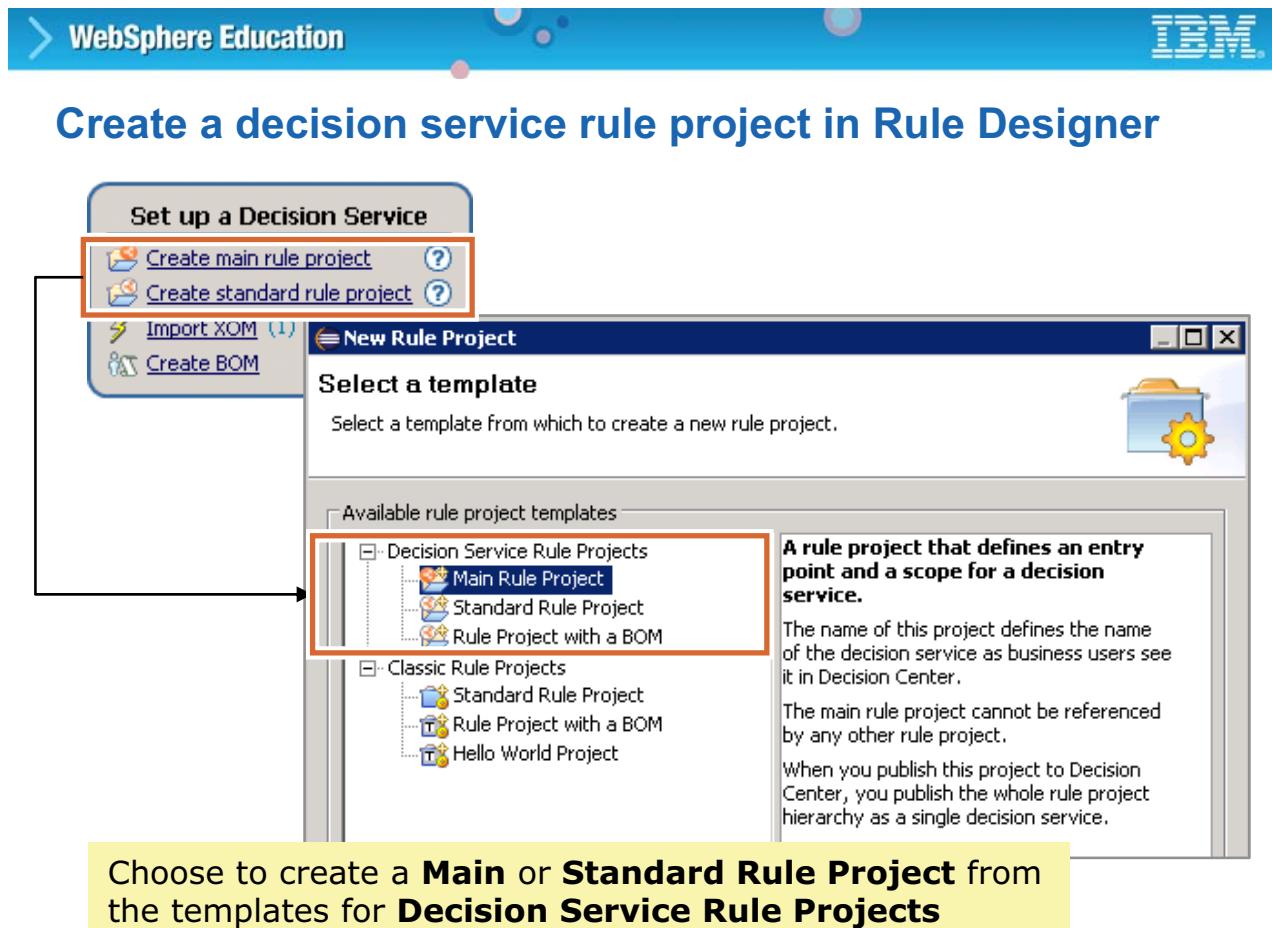
Rule Designer includes a Decision Service Map that helps set up a decision service. The Decision Service Map is available in the Rule Project Map view in your Rule Designer workspace.

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.



© Copyright IBM Corporation 2016

Figure 2-21. Create a decision service rule project in Rule Designer

WB3951.2

Notes:

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

© Copyright IBM Corporation 2016

Figure 2-22. Decision service rule project templates

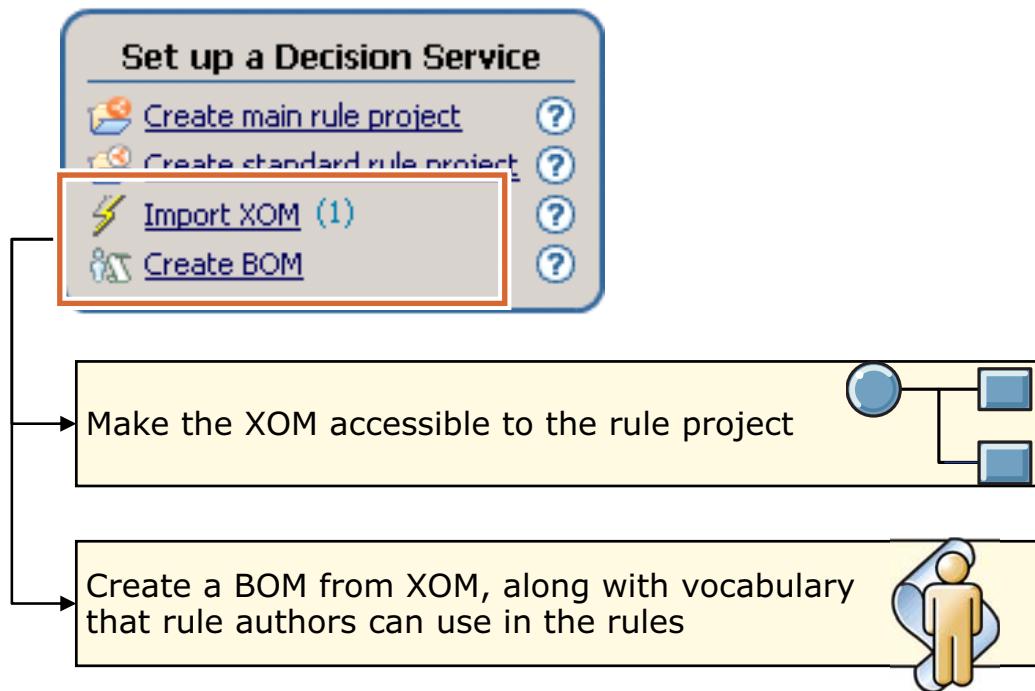
WB3951.2

Notes:

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.

WebSphere Education

Design: XOM and BOM



© Copyright IBM Corporation 2016

Figure 2-23. Design: XOM and BOM

WB3951.2

Notes:

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.

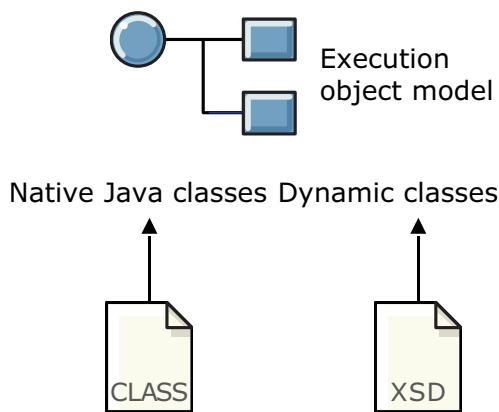
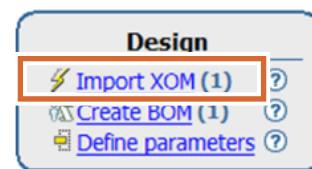
The XOM is stored in a separate project; it is not part of the rule project. However, when you create the rule project, you import the XOM into the rule project, which associates that XOM to the rule project.

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



© Copyright IBM Corporation 2016

Figure 2-24. Design: XOM

WB3951.2

Notes:

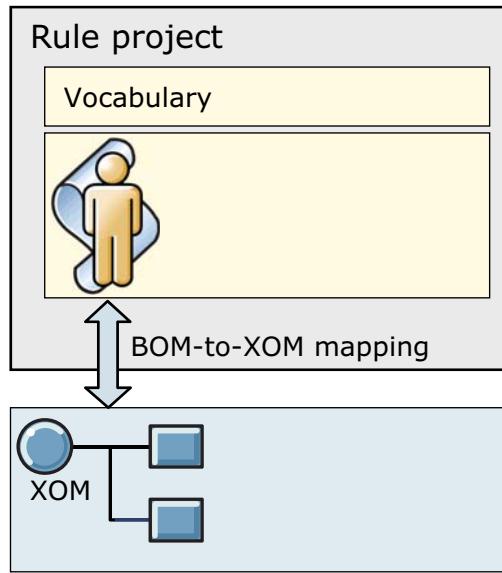
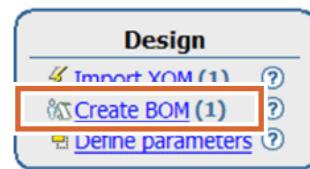
The business logic can be implemented by using either Java objects or an XML schema (XSD). As mentioned earlier, the implementation code is called the execution object model (XOM). The XOM makes rule execution possible. While business rules use the vocabulary from the BOM, each business element in a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

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



Design: BOM

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



© Copyright IBM Corporation 2016

Figure 2-25. Design: BOM

WB3951.2

Notes:

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

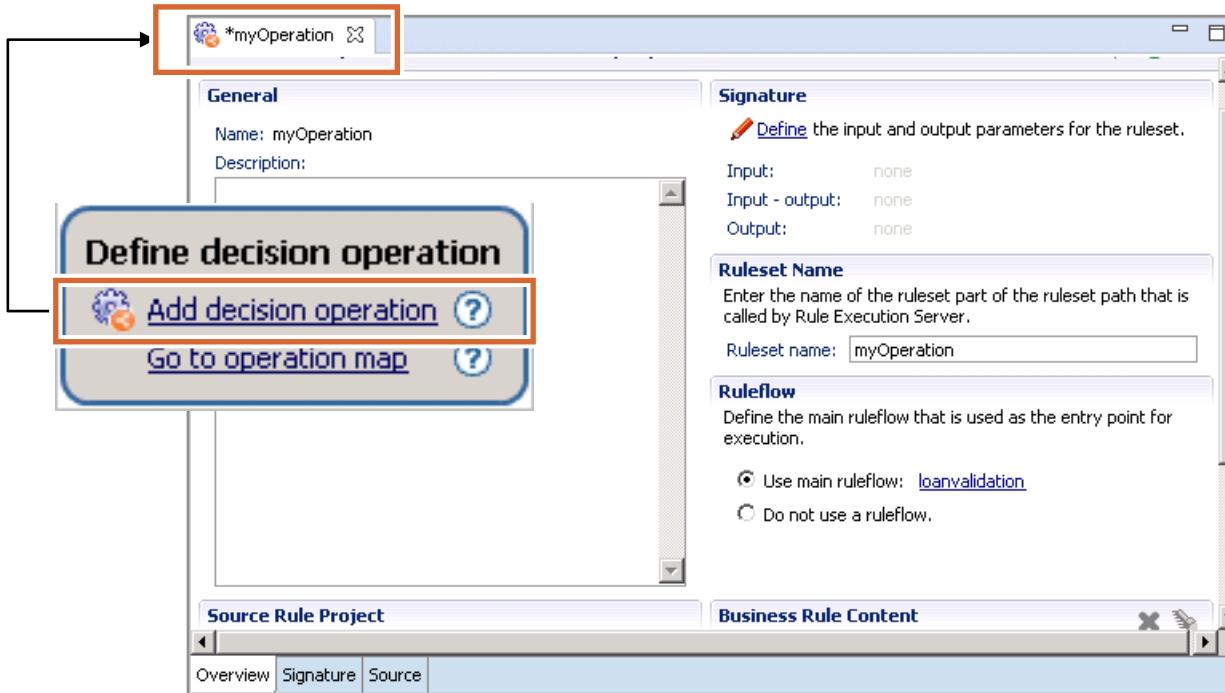
In some cases, you might choose to store the BOM in a separate rule project so that it can be shared across multiple rule projects.

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.



Defining the decision operation

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



© Copyright IBM Corporation 2016

Figure 2-26. Defining the decision operation

WB3951.2

Notes:

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



Using the Operation Map

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

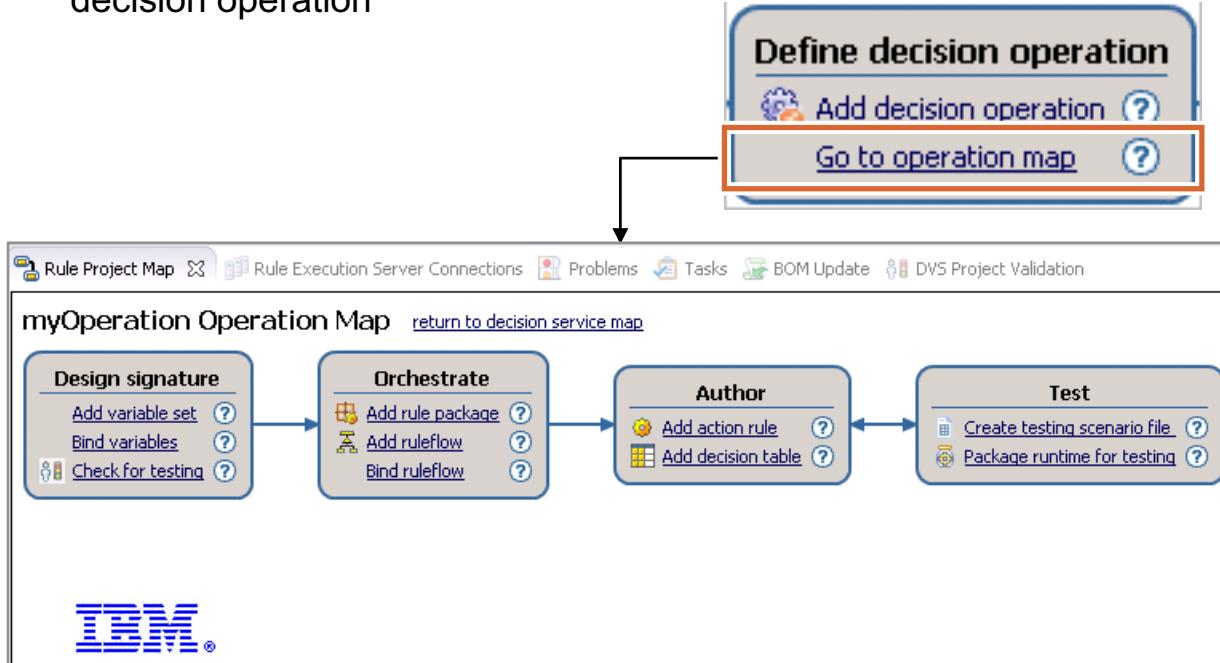


Figure 2-27. Using the Operation Map

WB3951.2

Notes:

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

Designing the signature

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

Design signature

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

Variable Set: loan-rulesParameters

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

© Copyright IBM Corporation 2016

Figure 2-28. Designing the signature

WB3951.2

Notes:

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

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

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

WebSphere Education

Designing the signature

- Bind the variables to the ruleset parameters

Decision Operation Signature - loan-rulesOperation

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	training_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	training_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	training_loan.Report	

© Copyright IBM Corporation 2016

Figure 2-29. Designing the signature

WB3951.2

Notes:

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 2016

Course materials may not be reproduced in whole or in part without the prior written permission of IBM.

10.1

Figure 2-30. Project properties

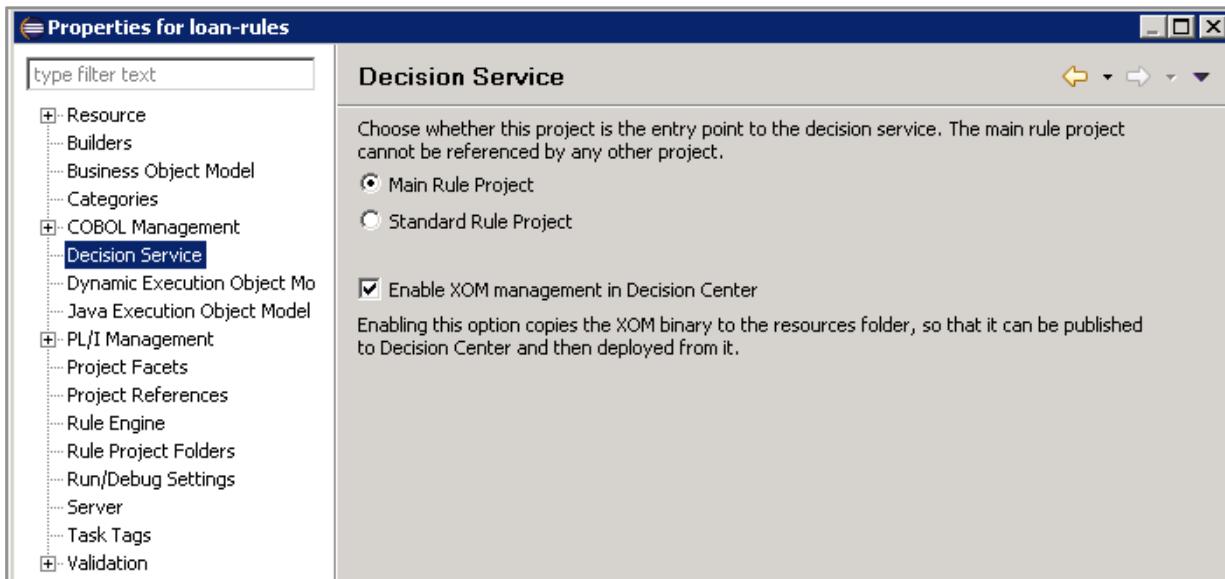
WB3951.2

Notes:



Properties: Ruleset parameters

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



© Copyright IBM Corporation 2016

Figure 2-31. Properties: Ruleset parameters

WB3951.2

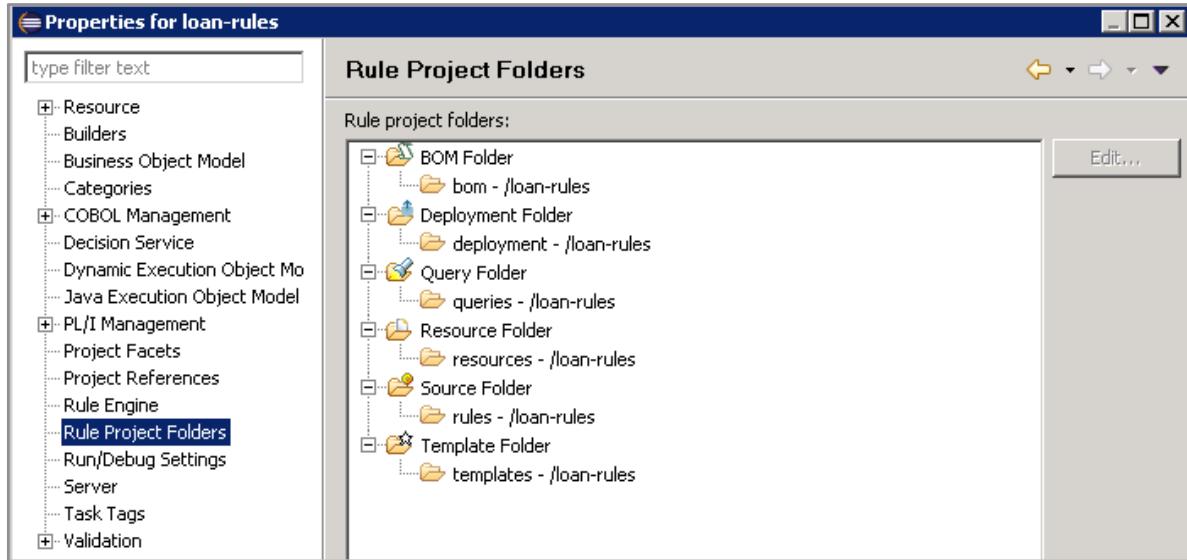
Notes:

Project properties are defined in the Properties dialog box.



Properties: Folders

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



© Copyright IBM Corporation 2016

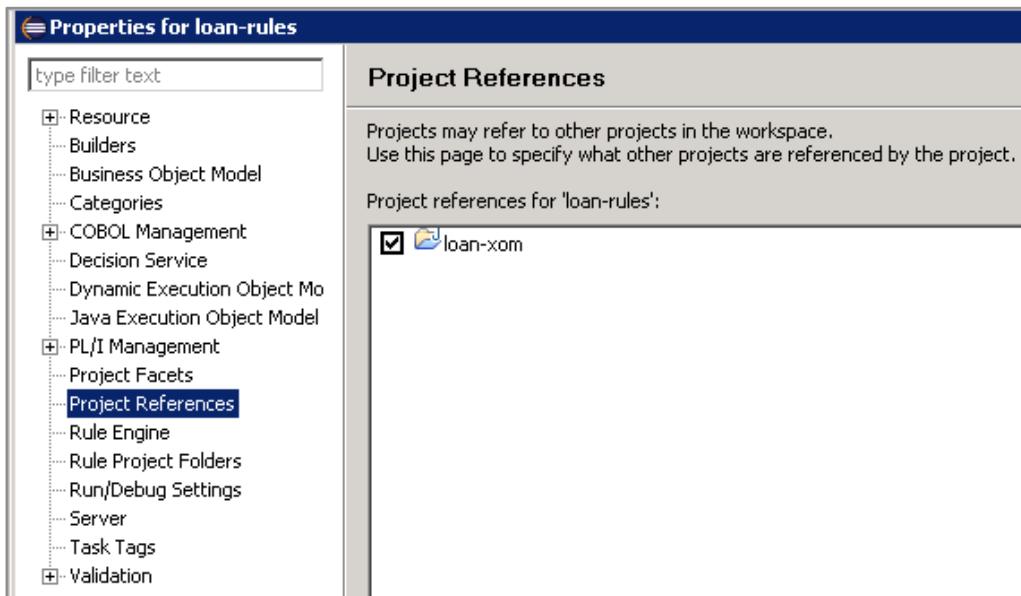
Figure 2-32. Properties: Folders

WB3951.2

Notes:

Properties: Project references

- Specify project references to define dependencies between projects



© Copyright IBM Corporation 2016

Figure 2-33. Properties: Project references

WB3951.2

Notes:

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
 - Classic rule engine
 - Decision engine

The screenshot shows the 'Properties for loan-rules' dialog box. On the left is a tree view of project components, with 'Rule Engine' selected. On the right is a configuration panel titled 'Rule Engine' containing two radio buttons: 'Classic rule engine' (selected) and 'Decision engine'. Below the radio buttons is the text 'Version: 1.30.0'. At the bottom right of the dialog is the copyright notice '© Copyright IBM Corporation 2016'.

© Copyright IBM Corporation 2016

Figure 2-34. Properties: Rule engine type

WB3951.2

Notes:

You learn more about the rule engine later in this course.

2.5. Using modular project organization

Using modular project organization



© Copyright IBM Corporation 2016

Course materials may not be reproduced in whole or in part without the prior written permission of IBM.

10.1

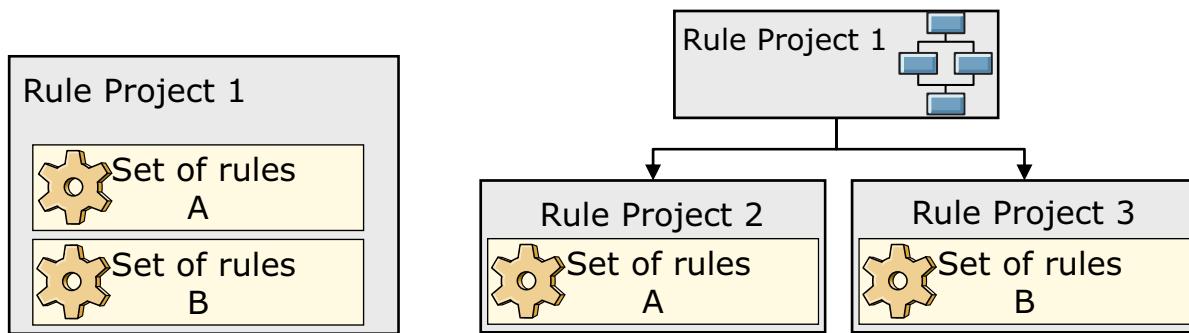
Figure 2-35. Using modular project organization

WB3951.2

Notes:

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



© Copyright IBM Corporation 2016

Figure 2-36. Modular structure (1 of 4)

WB3951.2

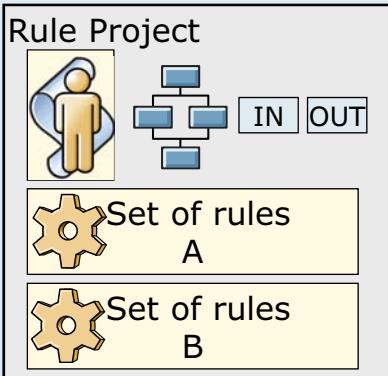
Notes:

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.

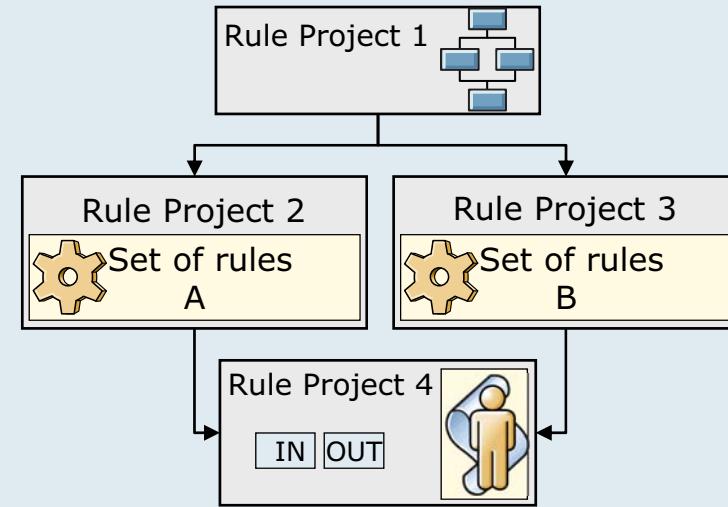
Modular structure (2 of 4)

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

- Avoid managing a large number of artifacts in a single project



- Modular architecture makes it easier to maintain rules and share work



© Copyright IBM Corporation 2016

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

WB3951.2

Notes:

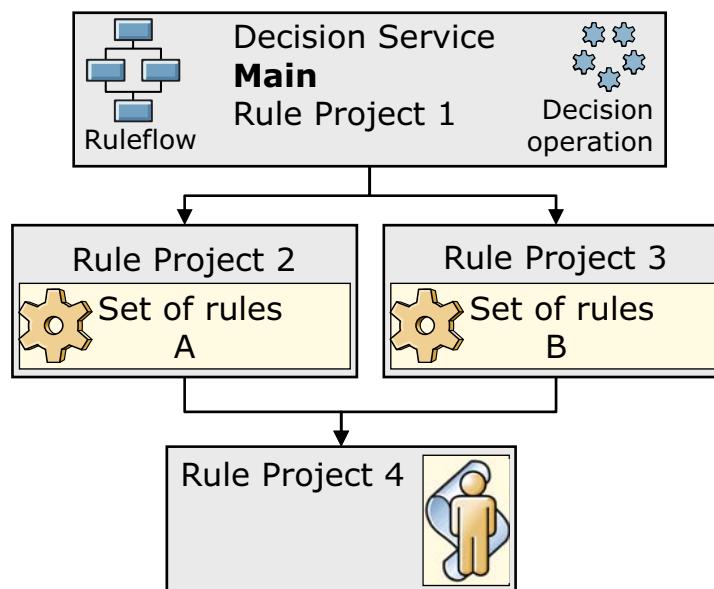
As the business rule application evolves, developers can set up multiple rule projects to handle large numbers of rules.

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



© Copyright IBM Corporation 2016

Figure 2-38. Modular structure (3 of 4)

WB3951.2

Notes:

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.

WebSphere Education

IBM

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

The screenshot shows the 'Properties for myDecisionService' dialog box and the 'Rule Explorer' view. The 'Project References' tab in the properties dialog lists 'ds project1' and 'ds project2' as referenced projects. The 'Rule Explorer' view shows a tree structure for 'myDecisionService' containing 'rules', 'bom', 'deployment', 'queries', 'resources', and 'templates'. A copyright notice at the bottom right reads '© Copyright IBM Corporation 2016'.

Figure 2-39. Modular structure (4 of 4)

WB3951.2

Notes:

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 2016

Course materials may not be reproduced in whole or in part without the prior written permission of IBM.

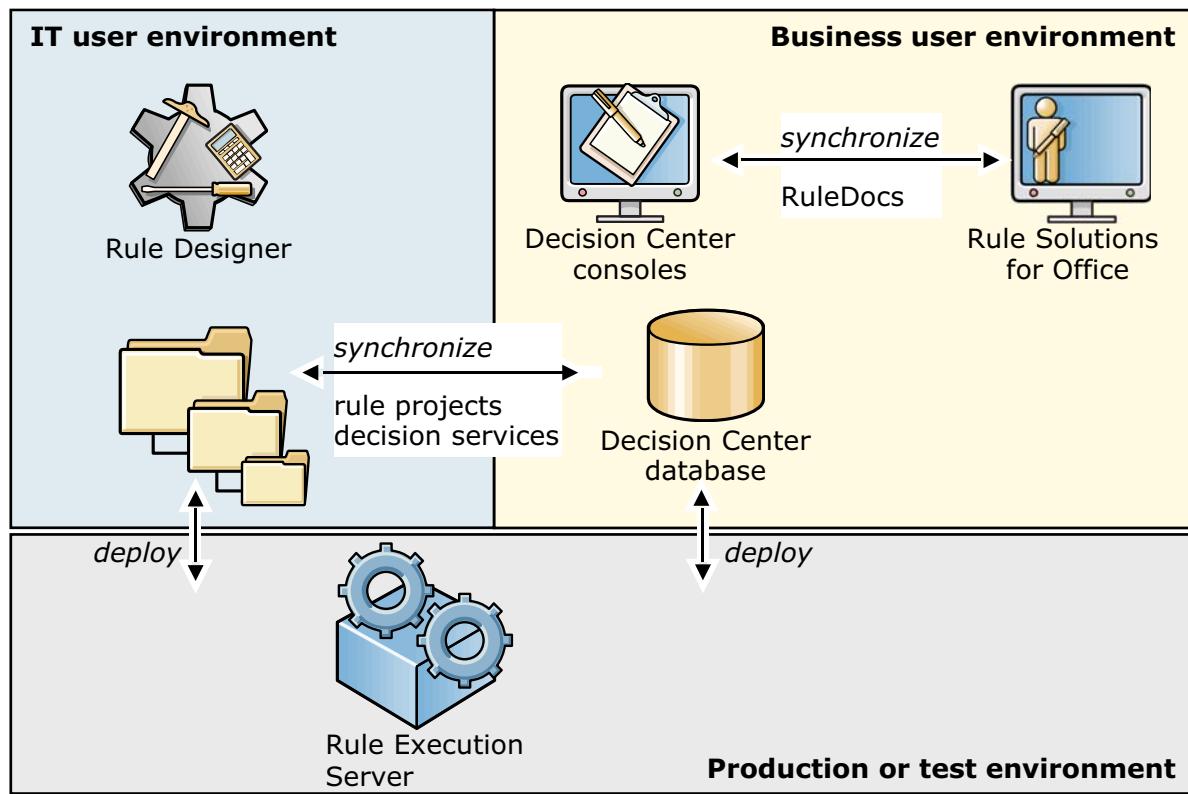
10.1

Figure 2-40. Sharing and synchronizing decision services

WB3951.2

Notes:

Synchronization



© Copyright IBM Corporation 2016

Figure 2-41. Synchronization

WB3951.2

Notes:

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.

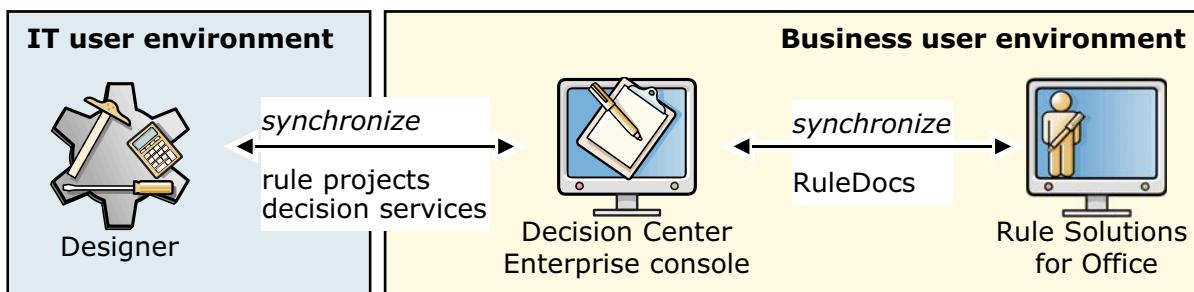
- Business users store rule projects in the Decision Center repository, and access them through the Decision Center Business console or Enterprise console. They can also edit rules through Rule Solutions for Office. The Decision Center repository handles multi-user concurrency and version control.
- 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 between users

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

- Synchronization between business users
 - Initiated and controlled from Decision Center Enterprise console
 - Compares rule artifacts that are stored in the Decision Center repository with the rules that are edited in the RuleDocs



© Copyright IBM Corporation 2016

Figure 2-42. Synchronization between users

WB3951.2

Notes:



Synchronization tools

- Publish an existing rule or event 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

© Copyright IBM Corporation 2016

Figure 2-43. Synchronization tools

WB3951.2

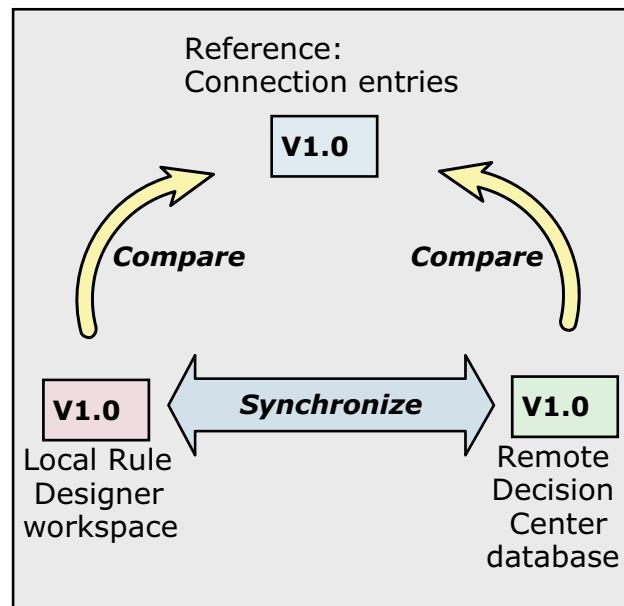
Notes:

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, and then compares them to the reference state



© Copyright IBM Corporation 2016

Figure 2-44. Synchronization architecture

WB3951.2

Notes:

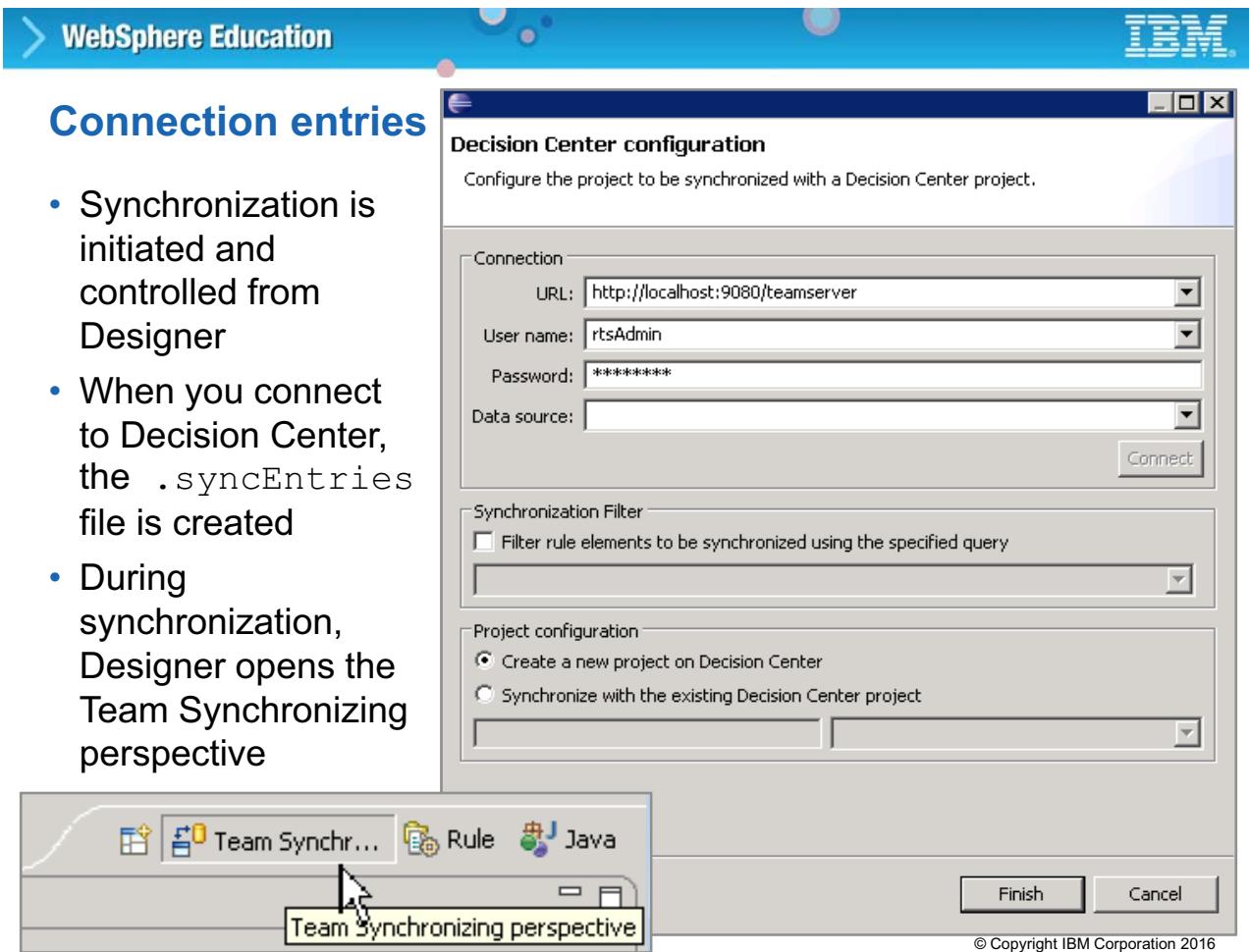


Figure 2-45. Connection entries

WB3951.2

Notes:

WebSphere Education

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

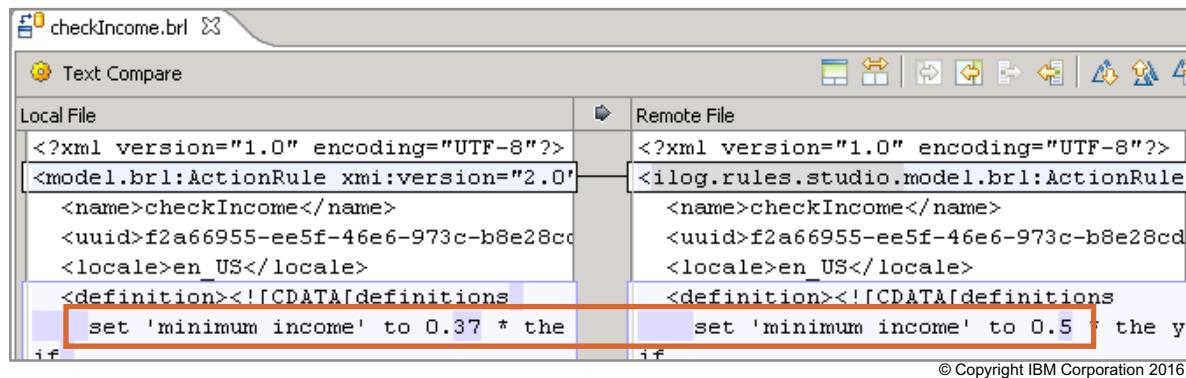
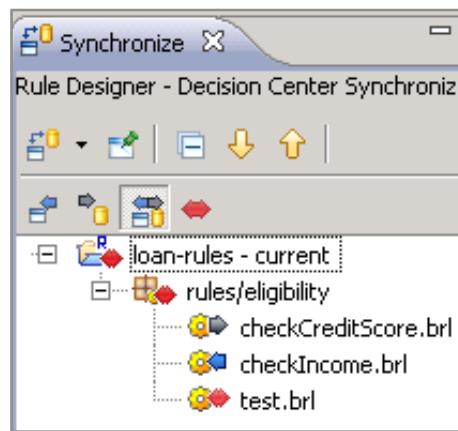


Figure 2-46. Choosing how to synchronize

WB3951.2

Notes:



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

© Copyright IBM Corporation 2016

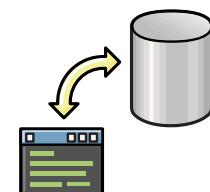
Figure 2-47. Synchronization commands

WB3951.2

Notes:

Choosing the master source

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



© Copyright IBM Corporation 2016

Figure 2-48. Choosing the master source

WB3951.2

Notes:

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

Having completed this unit, you should be able to:

- 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 2016

Figure 2-49. Unit summary

WB3951.2

Notes:



Checkpoint 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.

© Copyright IBM Corporation 2016

Figure 2-50. Checkpoint questions

WB3951.2

Notes:

Write your answers here:

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



Checkpoint answers

1. **False:** *Agile Business Rule Development involves collaboration between development and business teams throughout project development, and especially during the early phases.*
2. **True.**
3. **True.**
4. **False:** *Synchronization between Rule Designer and Decision Center is controlled from Rule Designer.*

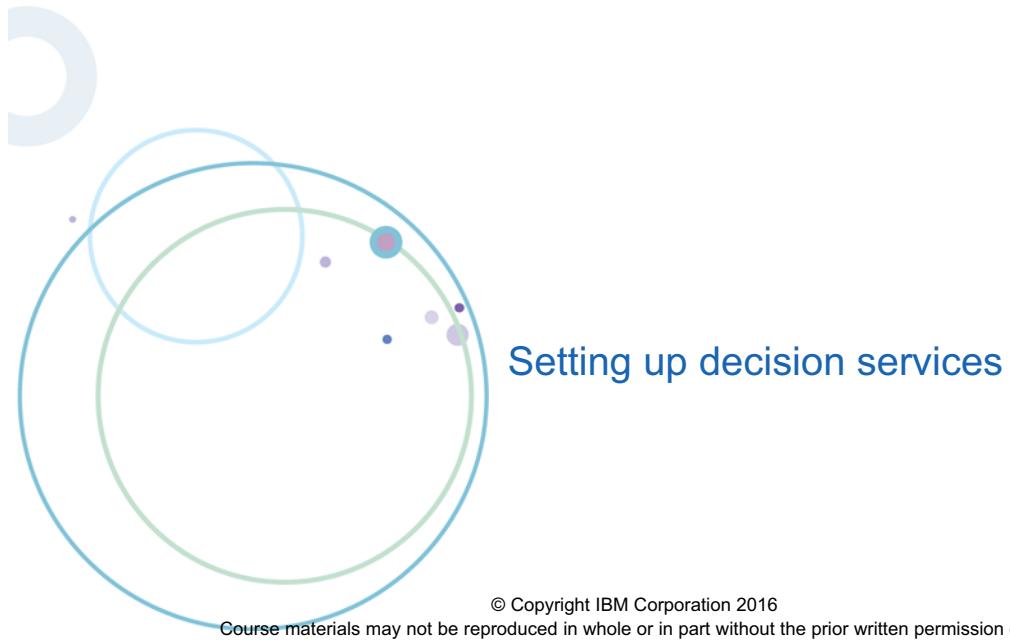
© Copyright IBM Corporation 2016

Figure 2-51. Checkpoint answers

WB3951.2

Notes:

Exercise 2



© Copyright IBM Corporation 2016

Course materials may not be reproduced in whole or in part without the prior written permission of IBM.

10.1

Figure 2-52. Exercise 2

WB3951.2

Notes:

Exercise objectives

After completing this exercise, you should be able to:

- 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 2016

Figure 2-53. Exercise objectives

WB3951.2

Notes:

Unit 3. Programming with business rules

What this unit is about

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

What you should be able to do

After completing this unit, you should be able to:

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

How you will check your progress

- Checkpoint



Unit objectives

After completing this unit, you should be able to:

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

© Copyright IBM Corporation 2016

Figure 3-1. Unit objectives

WB3951.2

Notes:



Topics

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

© Copyright IBM Corporation 2016

Figure 3-2. Topics

WB3951.2

Notes:

3.1. Introducing ruleset integration and execution

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

Introducing ruleset integration and execution



© Copyright IBM Corporation 2016

Course materials may not be reproduced in whole or in part without the prior written permission of IBM.

10.1

Figure 3-3. Introducing ruleset integration and execution

WB3951.2

Notes:

Integrating Operational Decision Manager

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

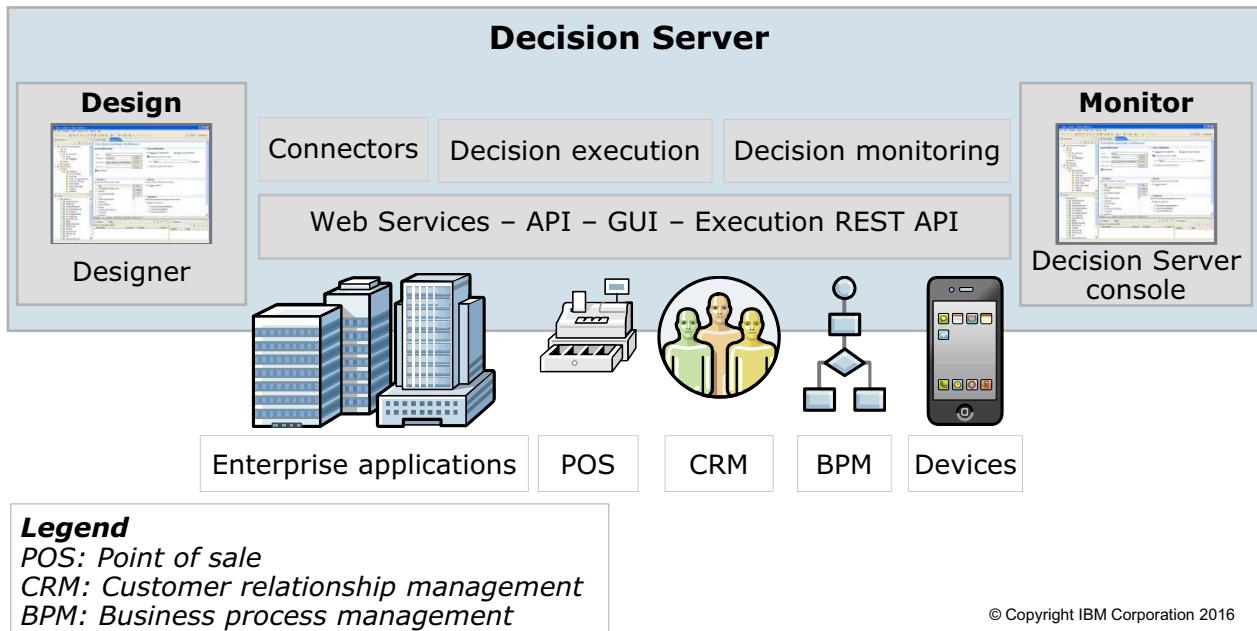


Figure 3-4. Integrating Operational Decision Manager

WB3951.2

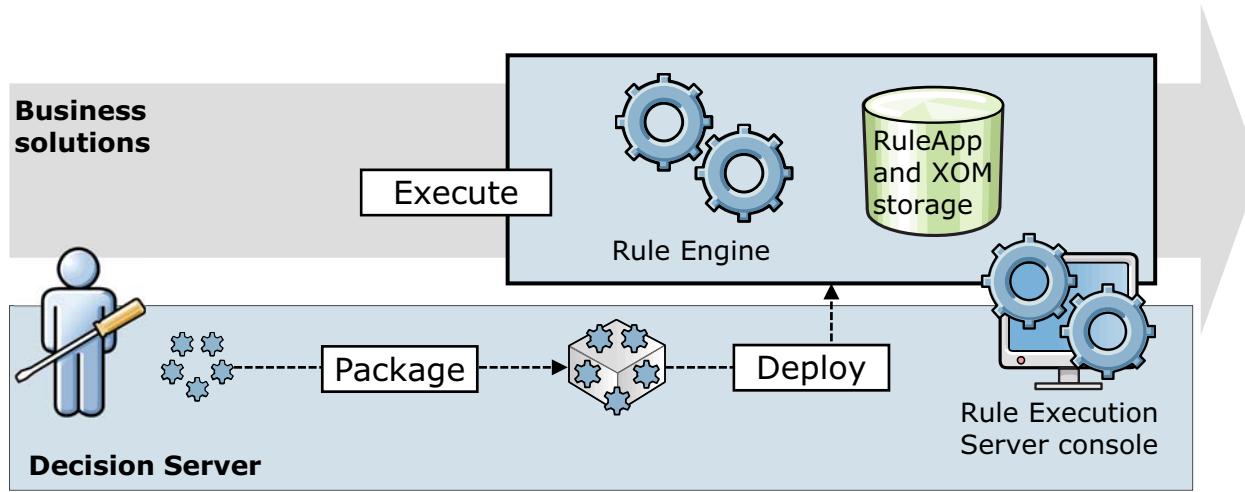
Notes:

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

Integration steps vary according to the execution environment.

Integration steps

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



© Copyright IBM Corporation 2016

Figure 3-5. Integration steps

WB3951.2

Notes:

Integration follows these basic steps:

1. Packaging

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

2. Deployment

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

3. Execution

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



Execution environments

- Ruleset execution can be embedded or managed
 - Embedded
 - The rule engine is integrated into a Java application
 - Used for test purposes
 - Managed
 - Rule execution is managed with Rule Execution Server

© Copyright IBM Corporation 2016

Figure 3-6. Execution environments

WB3951.2

Notes:

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

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

3.2. What is a rule engine?

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

What is a rule engine?



© Copyright IBM Corporation 2016

Course materials may not be reproduced in whole or in part without the prior written permission of IBM.

10.1

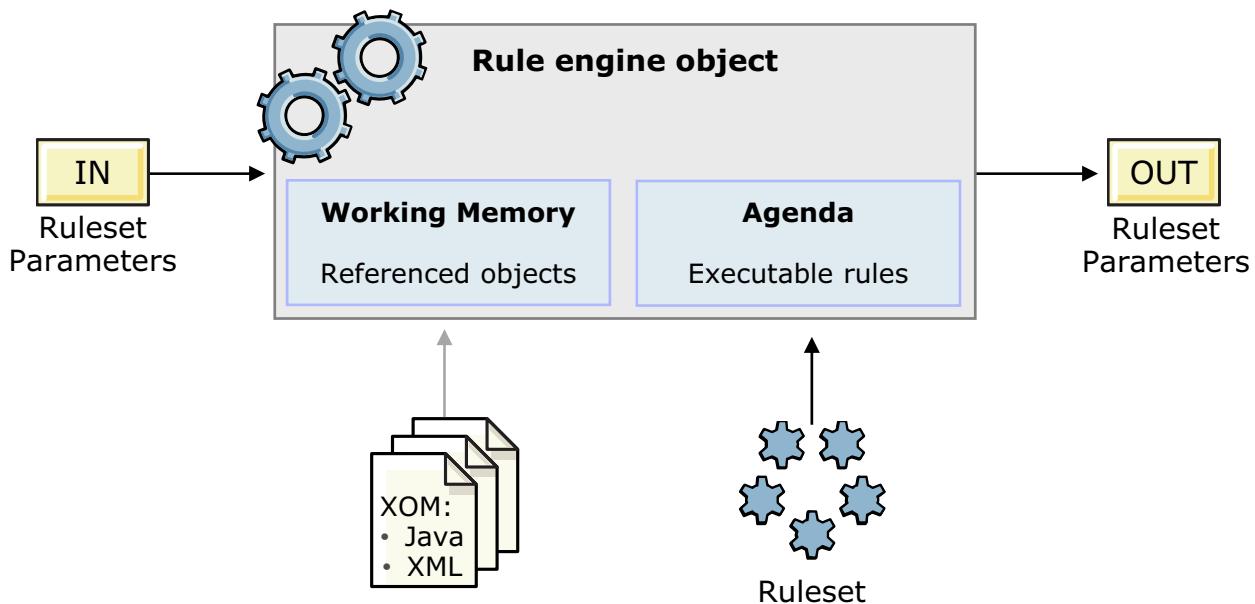
Figure 3-7. What is a rule engine?

WB3951.2

Notes:

Rule engine

- The rule engine is the module that executes the rules



© Copyright IBM Corporation 2016

Figure 3-8. Rule engine

WB3951.2

Notes:

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.



Note

In any rule, the engine accepts up to 20 object references. Otherwise, an error message is displayed. This feature corresponds to a rule with up to 20 conditions, excluding any evaluate conditions.

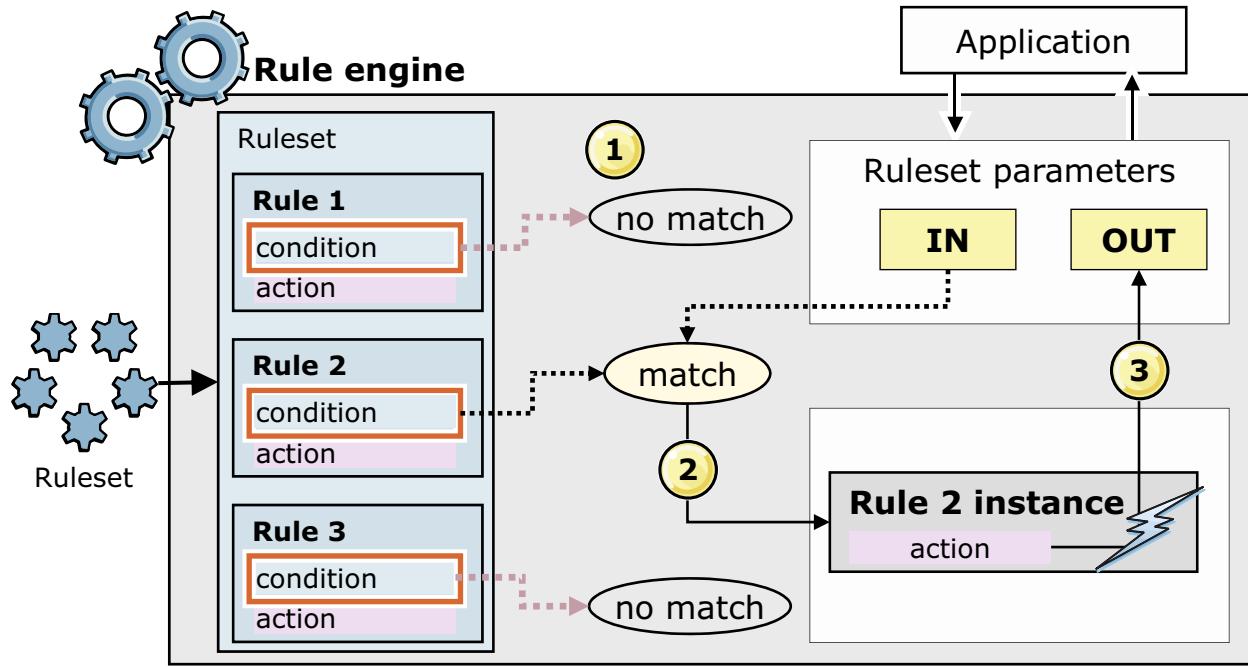
- The rule engine processes the rules that you provide. It evaluates the rules against the application objects and executes the rules when appropriate.

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 sequential execution modes.

Rule execution by the rule engine

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



© Copyright IBM Corporation 2016

Figure 3-9. Rule execution by the rule engine

WB3951.2

Notes:

This diagram outlines how the rule engine performs rule execution.

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

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



Execution modes

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

© Copyright IBM Corporation 2016

Figure 3-10. Execution modes

WB3951.2

Notes:

In Operational Decision Manager, the possible execution modes are RetePlus, Sequential, and Fastpath. You learn more about the principles of rule order and execution modes in Unit 5, "Orchestrating ruleset execution".

Rule engine types

- Classic rule engine
 - The default rule engine
- Decision engine
 - A high performance rule engine
- Both engines work in similar ways
 - Rules are passed to the decision engine already compiled as code
 - Ruleset loading in the decision engine is faster because no code is parsed or interpreted at run time

© Copyright IBM Corporation 2016

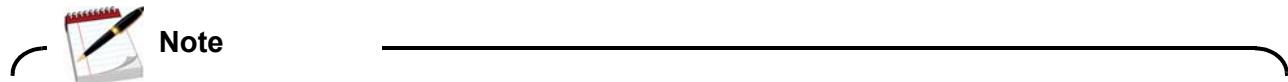
Figure 3-11. Rule engine types

WB3951.2

Notes:

ODM includes two types of rule engine:

- Classic rule engine
- Decision rule engine



IBM ODM on Cloud

ODM on Cloud does not support the classic rule engine.

Both engines work with the rules and application objects in the same way. However, the decision engine is designed to optimize the execution performance of your ruleset. The rules are compiled and loaded differently for the decision engine.

For more information about the differences between rule engines, and how to choose the appropriate engine for your execution environment, see the product documentation.

3.3. Understanding rule execution modes

Understanding rule execution modes



© Copyright IBM Corporation 2016

Course materials may not be reproduced in whole or in part without the prior written permission of IBM.

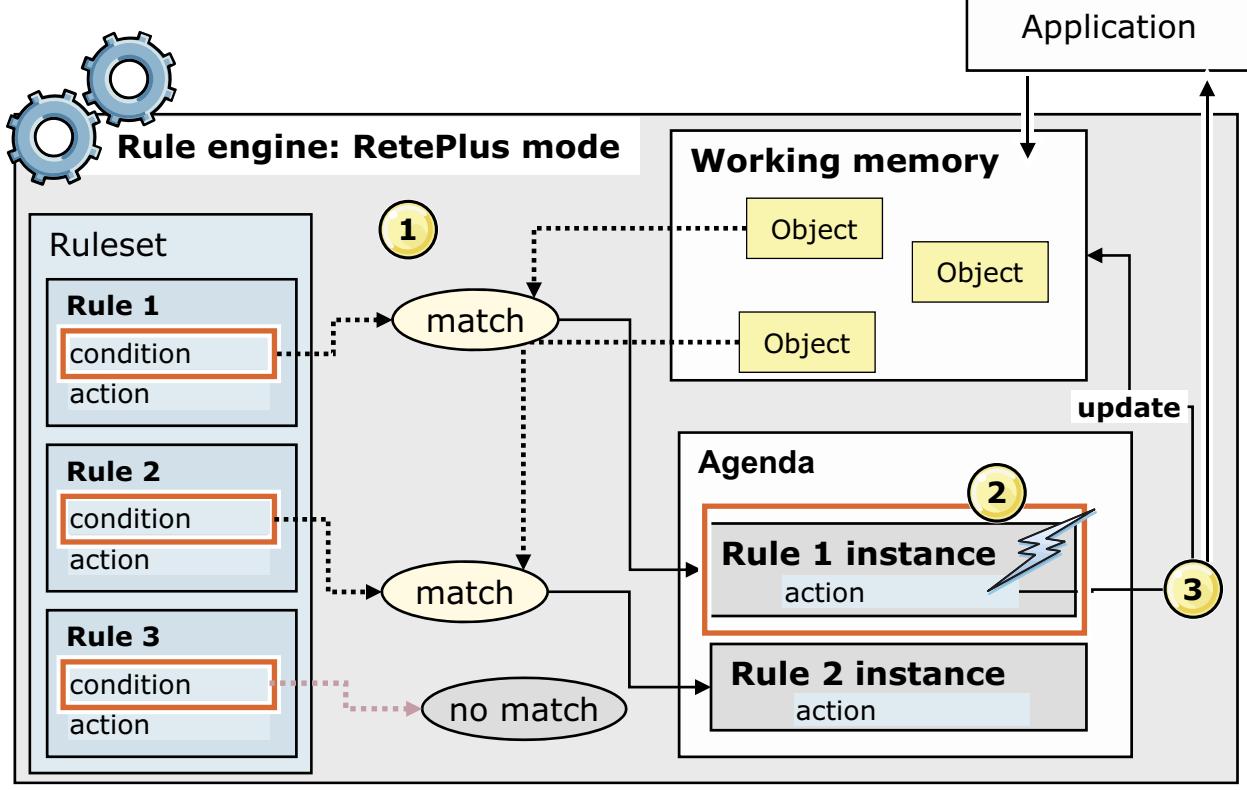
10.1

Figure 3-12. Understanding rule execution modes

WB3951.2

Notes:

Execution modes: RetePlus (1 of 3)



© Copyright IBM Corporation 2016

Figure 3-13. Execution modes: RetePlus (1 of 3)

WB3951.2

Notes:

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 ruleset 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)

© Copyright IBM Corporation 2016

Figure 3-14. Execution modes: RetePlus (2 of 3)

WB3951.2

Notes:

RetePlus is the default mode because it 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

© Copyright IBM Corporation 2016

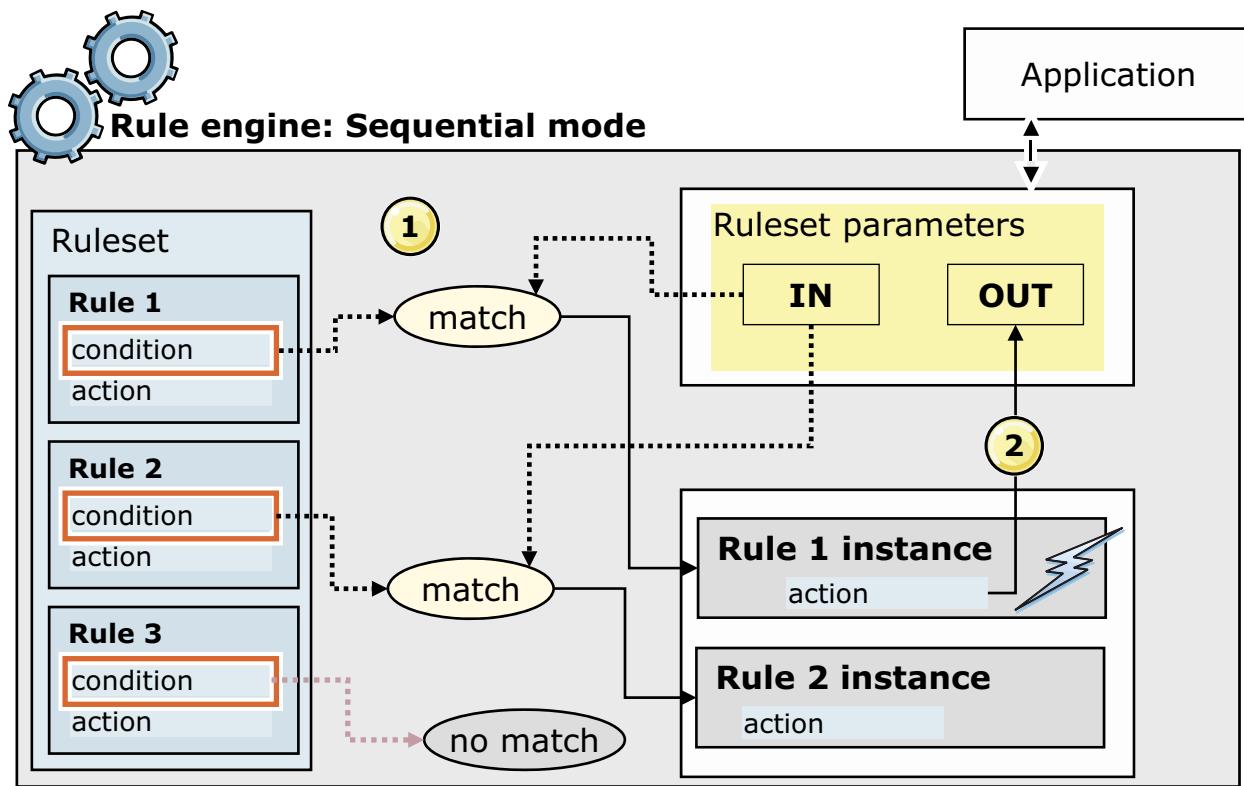
Figure 3-15. Execution modes: RetePlus (3 of 3)

WB3951.2

Notes:

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

Execution modes: Sequential (1 of 2)



© Copyright IBM Corporation 2016

Figure 3-16. Execution modes: Sequential (1 of 2)

WB3951.2

Notes:

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 ruleset 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 take that modification into account. 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

© Copyright IBM Corporation 2016

Figure 3-17. Execution modes: Sequential (2 of 2)

WB3951.2

Notes:

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

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

Execution modes: Fastpath (1 of 2)

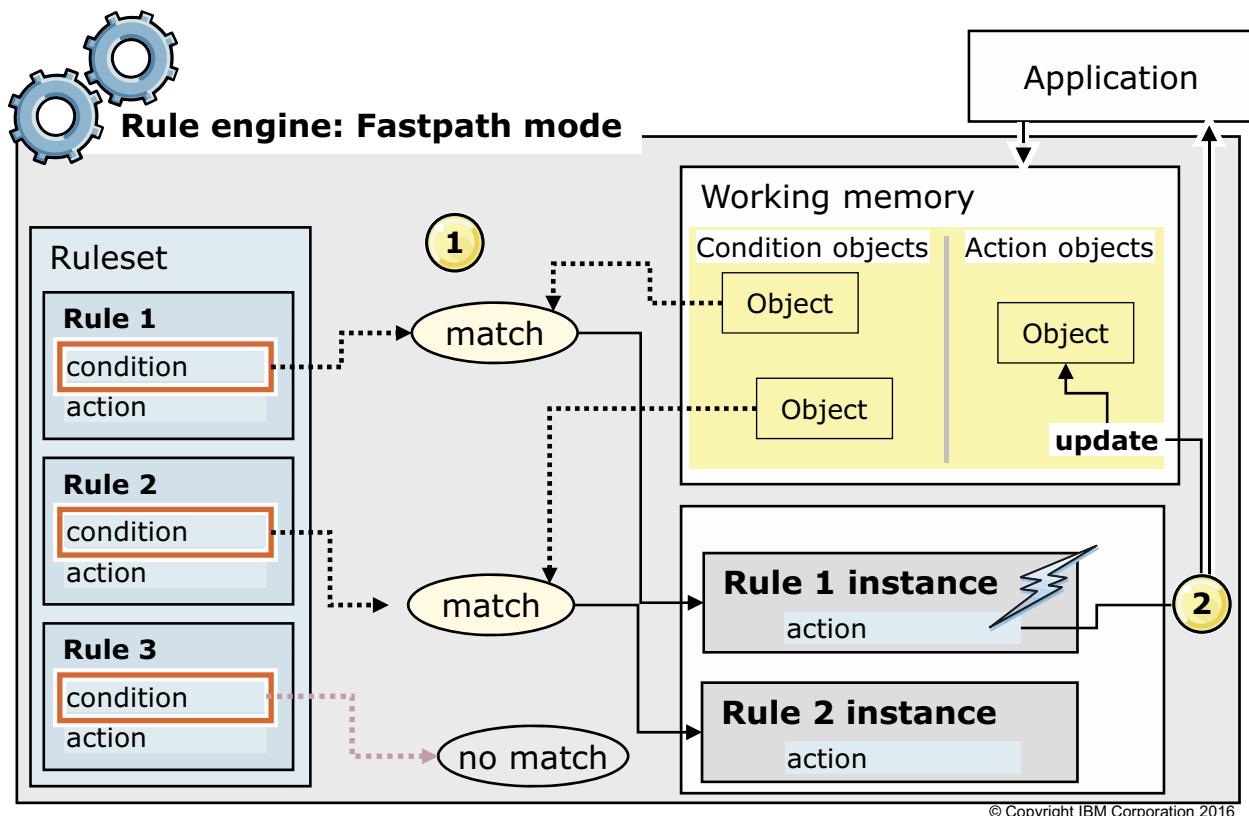


Figure 3-18. Execution modes: Fastpath (1 of 2)

WB3951.2

Notes:

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 ruleset 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 taken into account to redo the pattern matching process (**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

© Copyright IBM Corporation 2016

Figure 3-19. Execution modes: Fastpath (2 of 2)

WB3951.2

Notes:

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.



RetePlus: Rule order and selection

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

© Copyright IBM Corporation 2016

Figure 3-20. RetePlus: Rule order and selection

WB3951.2

Notes:

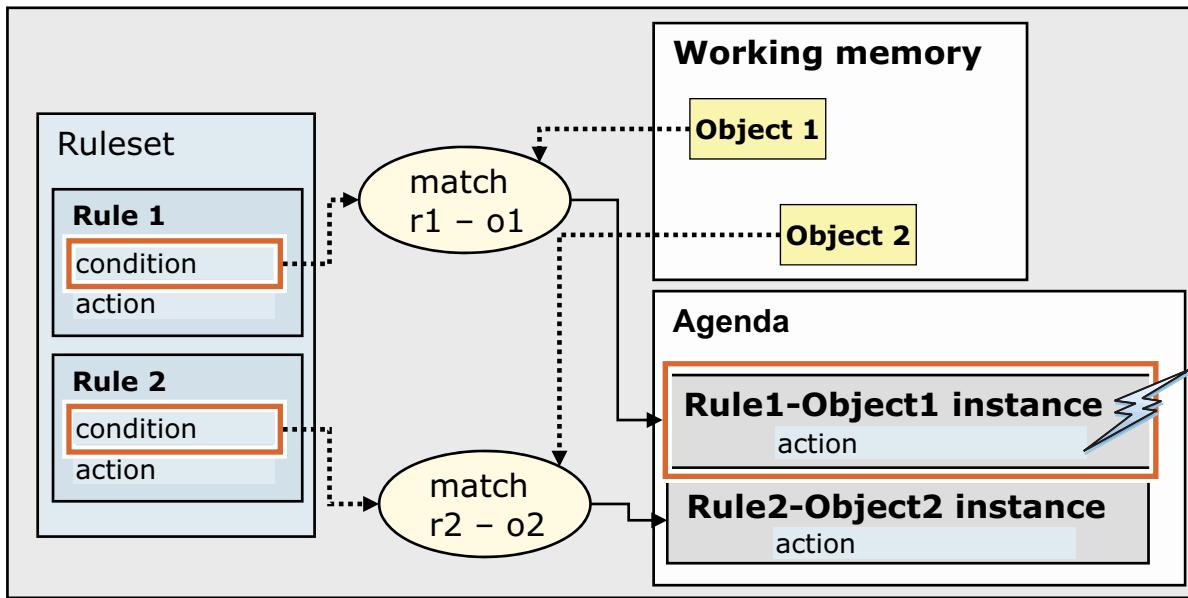
As previously explained, when a rule condition matches objects in working memory, a rule instance is placed into the agenda.

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

Refraction (1 of 2)

Example:

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



© Copyright IBM Corporation 2016

Figure 3-21. Refraction (1 of 2)

WB3951.2

Notes:

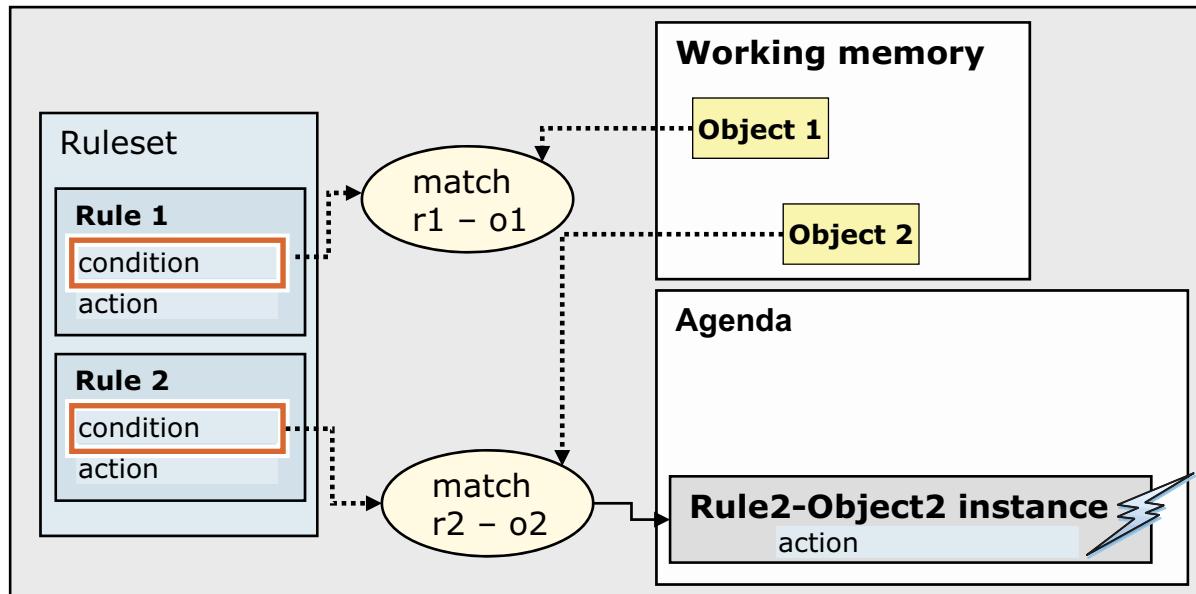
Refraction helps eliminate trivial loops. A rule instance cannot be put back in the agenda if:

- 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



© Copyright IBM Corporation 2016

Figure 3-22. Refraction (2 of 2)

WB3951.2

Notes:

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.

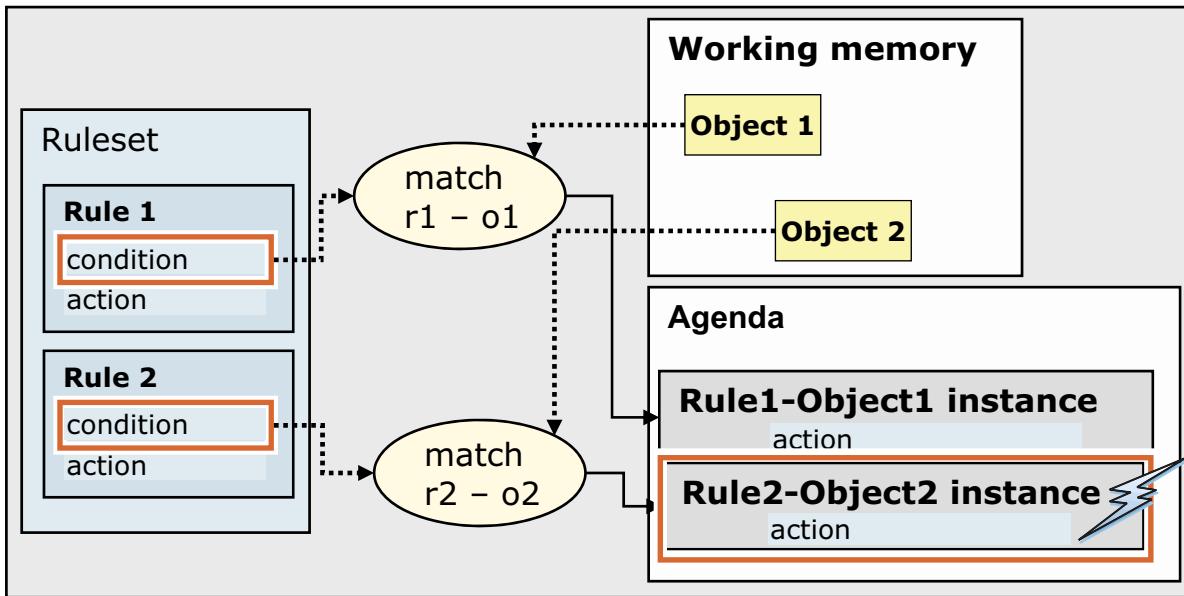


Note

You can use the rule engine API to override refraction.

Priority

- Rule instances with the highest priority value are fired first
 - Define a static priority with a constant
 - Define a dynamic priority with an expression that the rule engine evaluates after the rule instance is added to the agenda



© Copyright IBM Corporation 2016

Figure 3-23. Priority

WB3951.2

Notes:

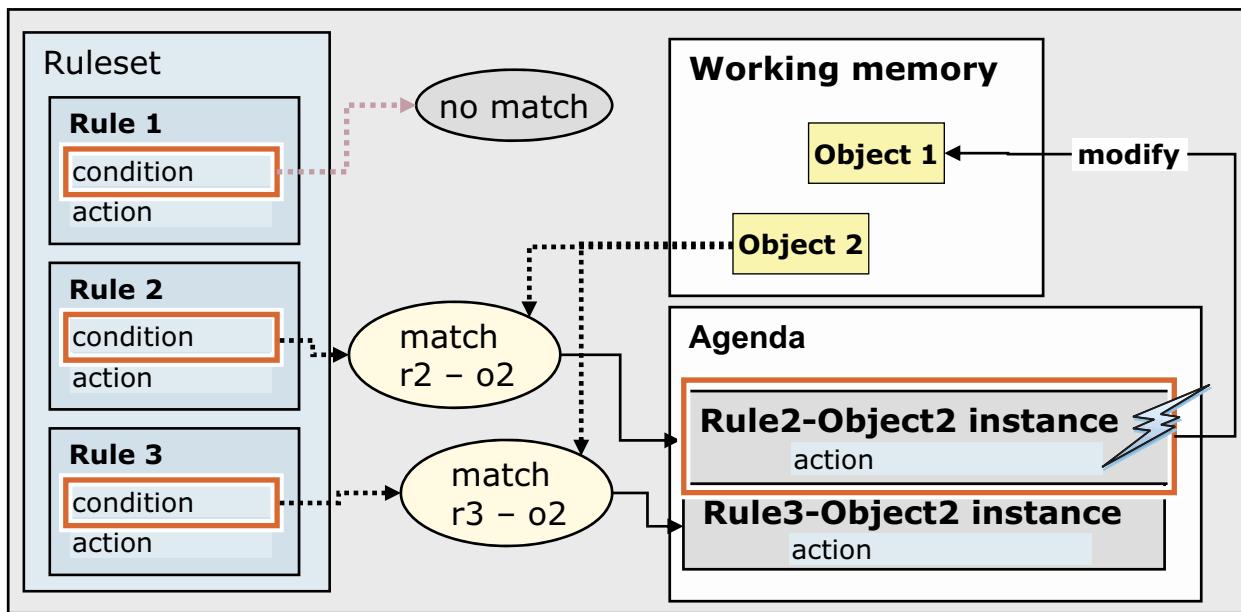
Rule instances with the highest priority value are fired first.

For example, if you have two rules, Rule1 and Rule2, that match two objects, Object1 and Object2, two rule instances, [Rule1-Object1] and [Rule2-Object2], are placed into the agenda.

Because Rule2 has a **priority** property that is set to high, and Rule1 does not have any priority that is set on it, the rule instance [Rule2-Object2] is fired first.

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



© Copyright IBM Corporation 2016

Figure 3-24. Recency (1 of 2)

WB3951.2

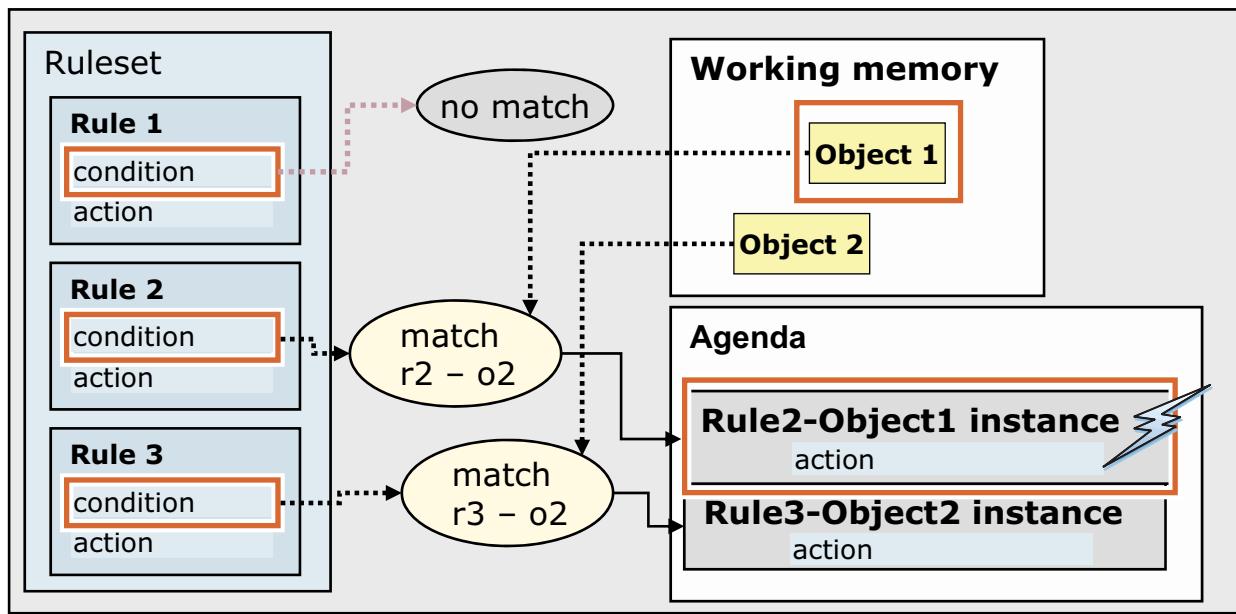
Notes:

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



© Copyright IBM Corporation 2016

Figure 3-25. Recency (2 of 2)

WB3951.2

Notes:

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.

3.4. RetePlus example: Trucks and drivers

RetePlus example: Trucks and drivers



© Copyright IBM Corporation 2016

Course materials may not be reproduced in whole or in part without the prior written permission of IBM.

10.1

Figure 3-26. RetePlus example: Trucks and drivers

WB3951.2

Notes:

RetePlus in action

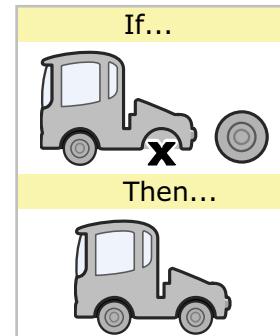
- The ruleset contains these rules:

ChangeTire:

If all of the following conditions are true:

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

Then change the tire;

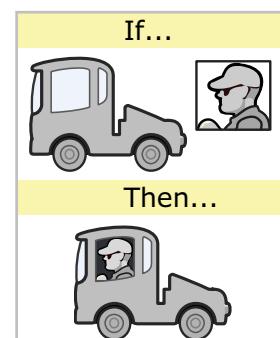


- AssignDriver:**

If all of the following conditions are true:

- a truck is available
- a driver is available

Then assign the truck to the driver;



© Copyright IBM Corporation 2016

Figure 3-27. RetePlus in action

WB3951.2

Notes:

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`.

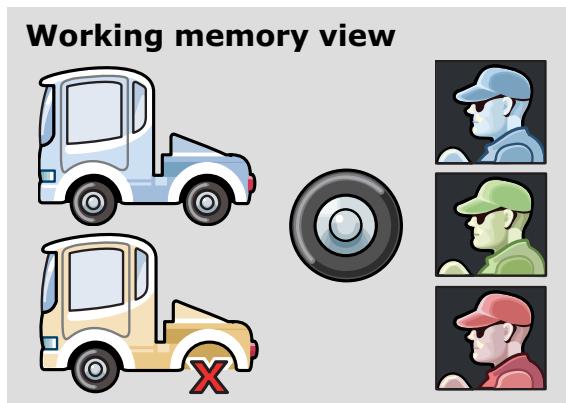
These action rules evaluate objects in working memory that are based on two BOM classes: `Truck` and `Driver`. They do not work on ruleset parameters or ruleset variables.

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

© Copyright IBM Corporation 2016

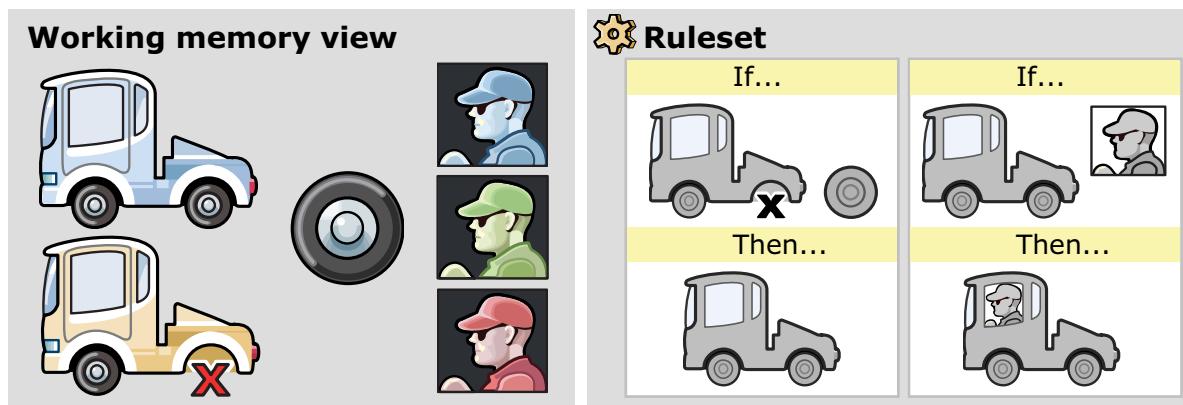
Figure 3-28. Input objects in working memory

WB3951.2

Notes:

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

© Copyright IBM Corporation 2016

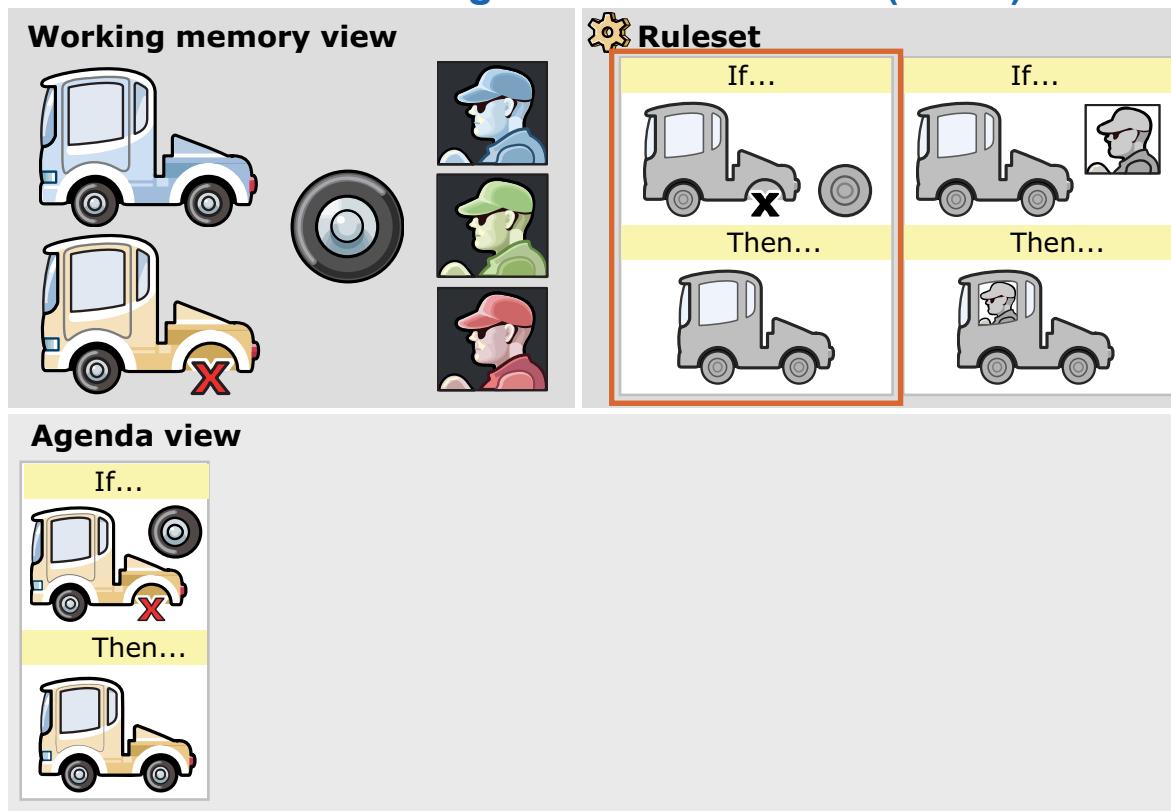
Figure 3-29. Evaluating rules with objects

WB3951.2

Notes:

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)



© Copyright IBM Corporation 2016

Figure 3-30. Rule instances in the agenda for execution (1 of 2)

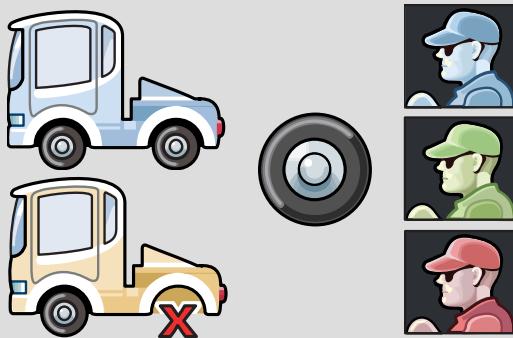
WB3951.2

Notes:

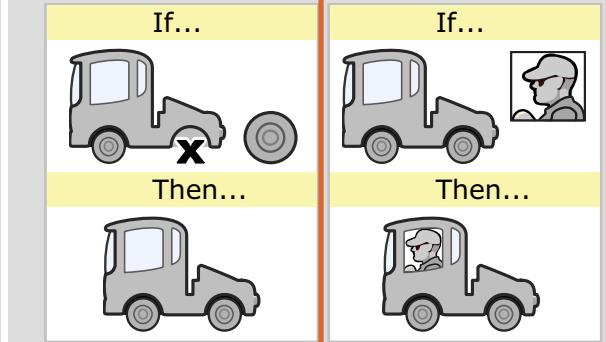
When the rule engine evaluates the `ChangeTire` rule conditions, it finds a matching truck and an available tire in the working memory. Because the rule condition is now true, an instance of the `ChangeTire` rule is put into the agenda.

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

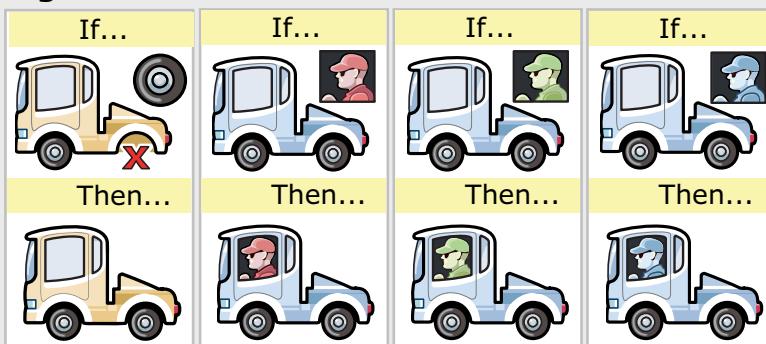
Working memory view



Ruleset



Agenda view



© Copyright IBM Corporation 2016

Figure 3-31. Rule instances in the agenda for execution (2 of 2)

WB3951.2

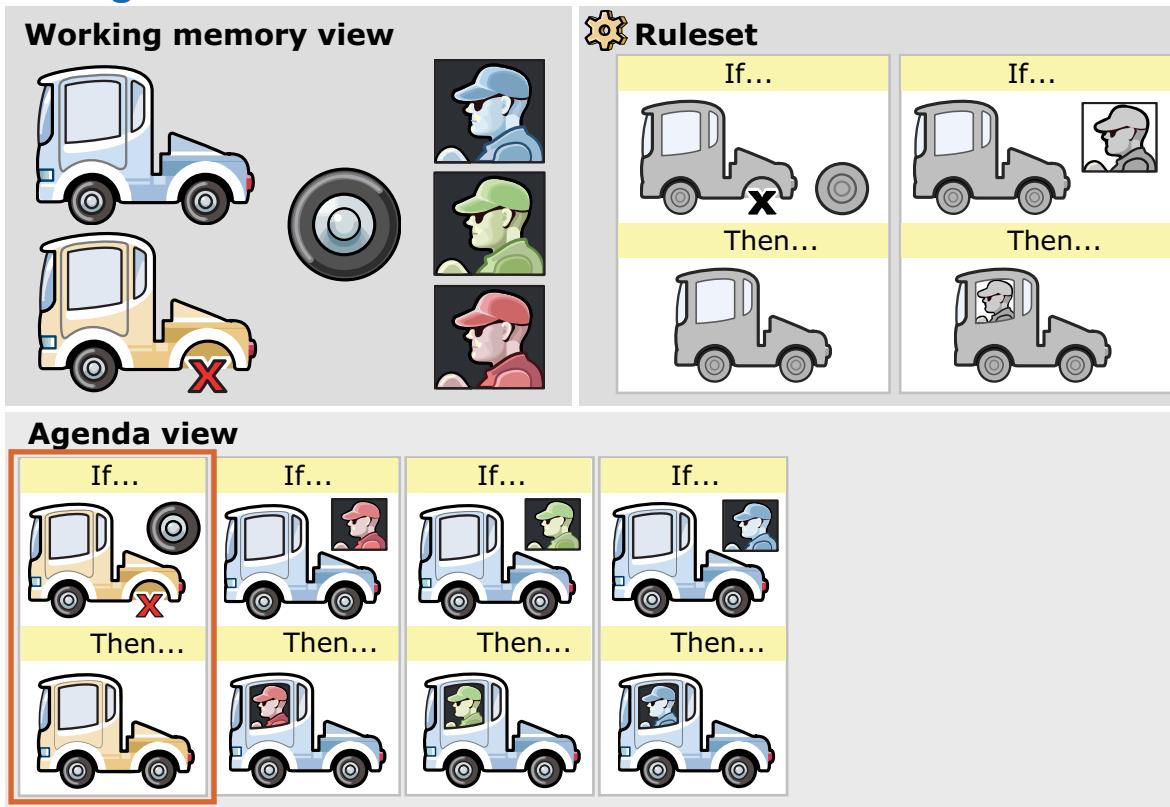
Notes:

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



© Copyright IBM Corporation 2016

Figure 3-32. ChangeTire executes

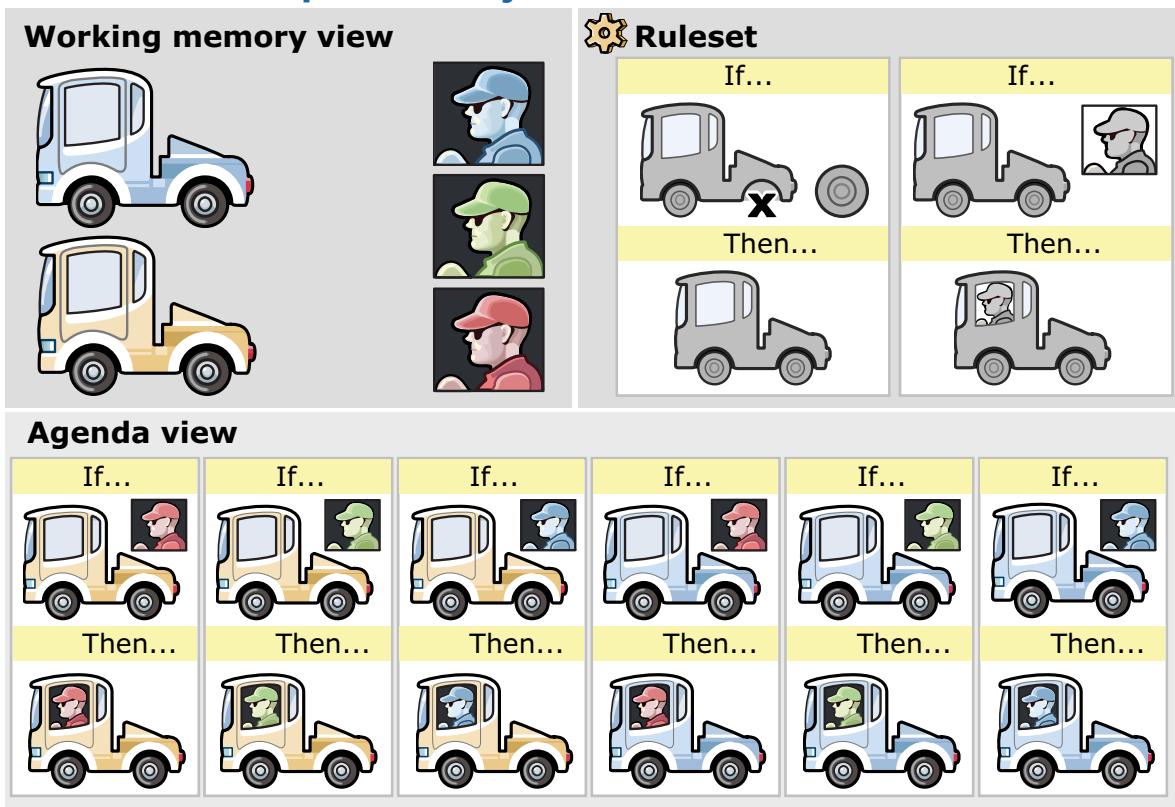
WB3951.2

Notes:

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



© Copyright IBM Corporation 2016

Figure 3-33. Side effects: Updated objects create matches

WB3951.2

Notes:

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

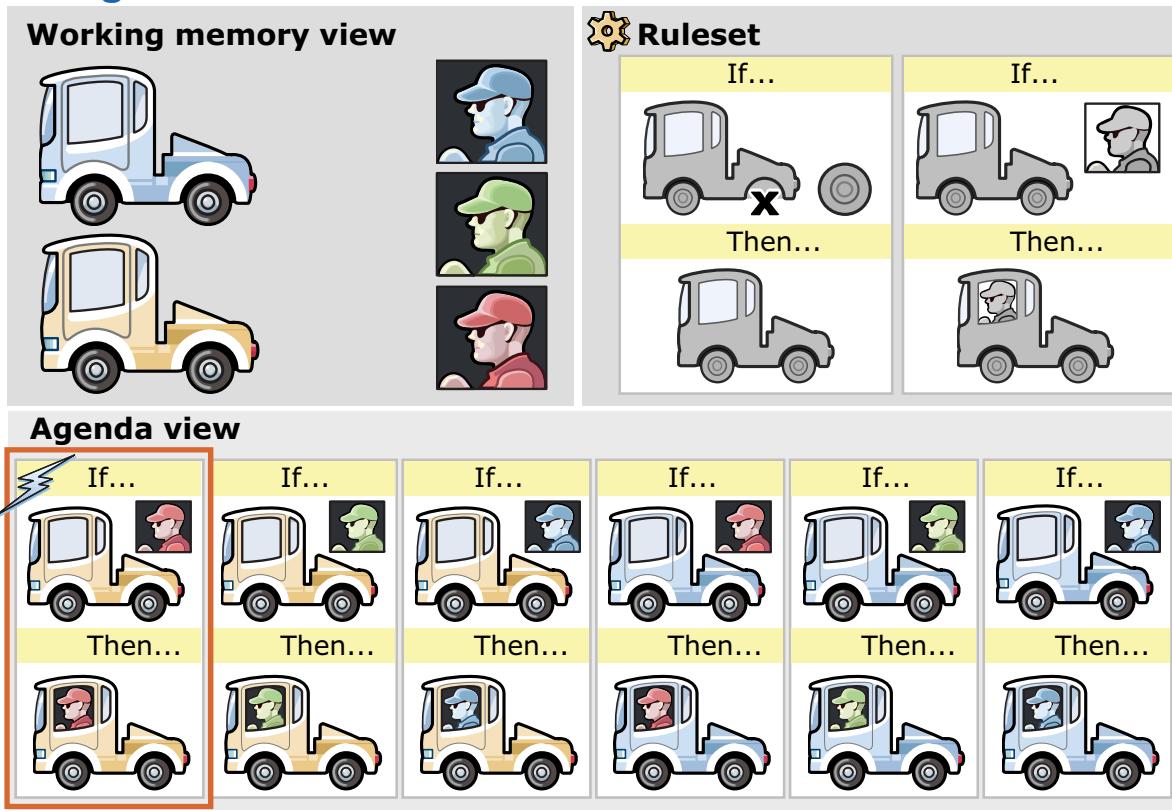


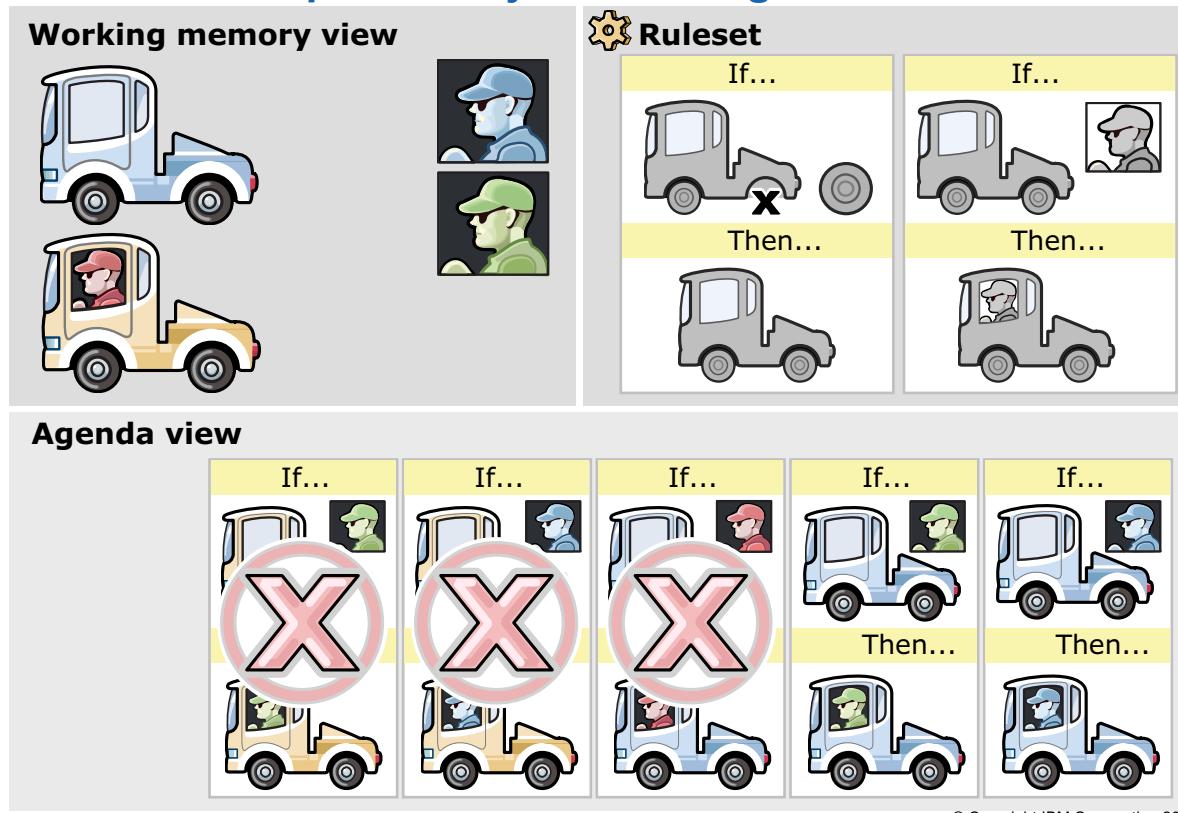
Figure 3-34. AssignDriver executes

WB3951.2

Notes:

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



© Copyright IBM Corporation 2016

Figure 3-35. Side effects: Updated objects no longer match

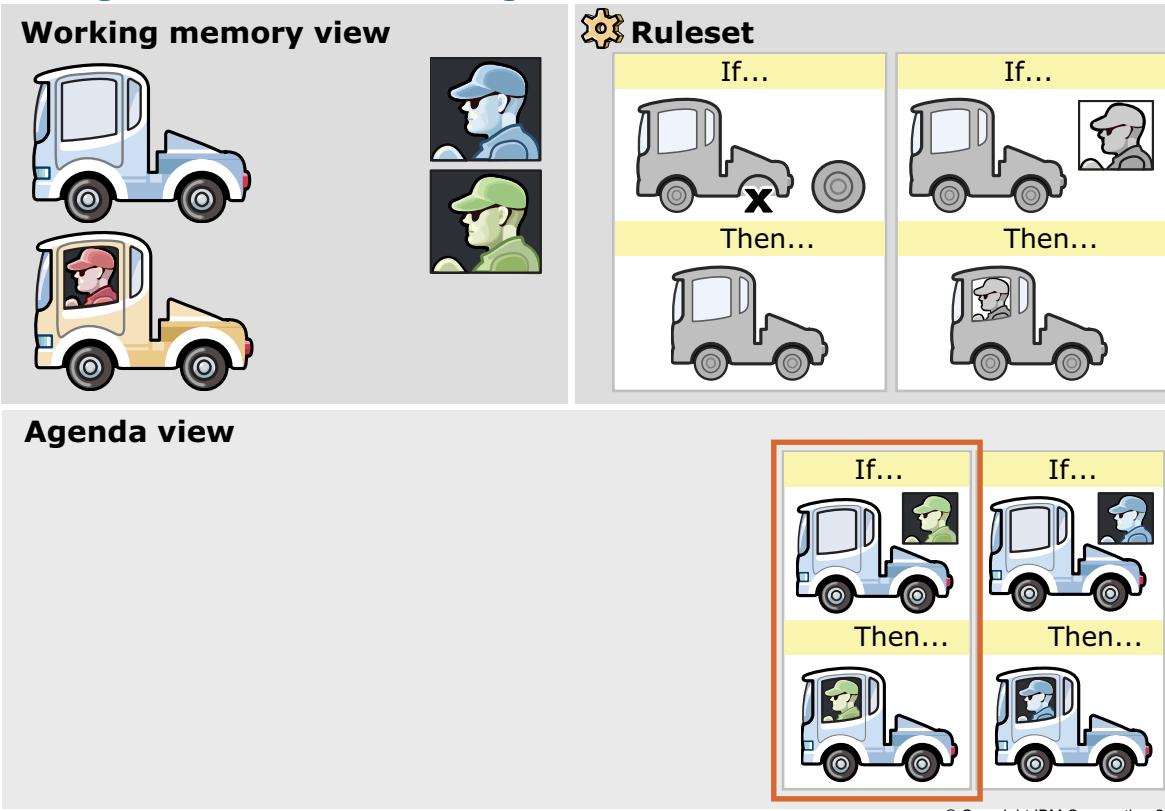
WB3951.2

Notes:

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.

WebSphere Education

AssignDriver executes again



© Copyright IBM Corporation 2016

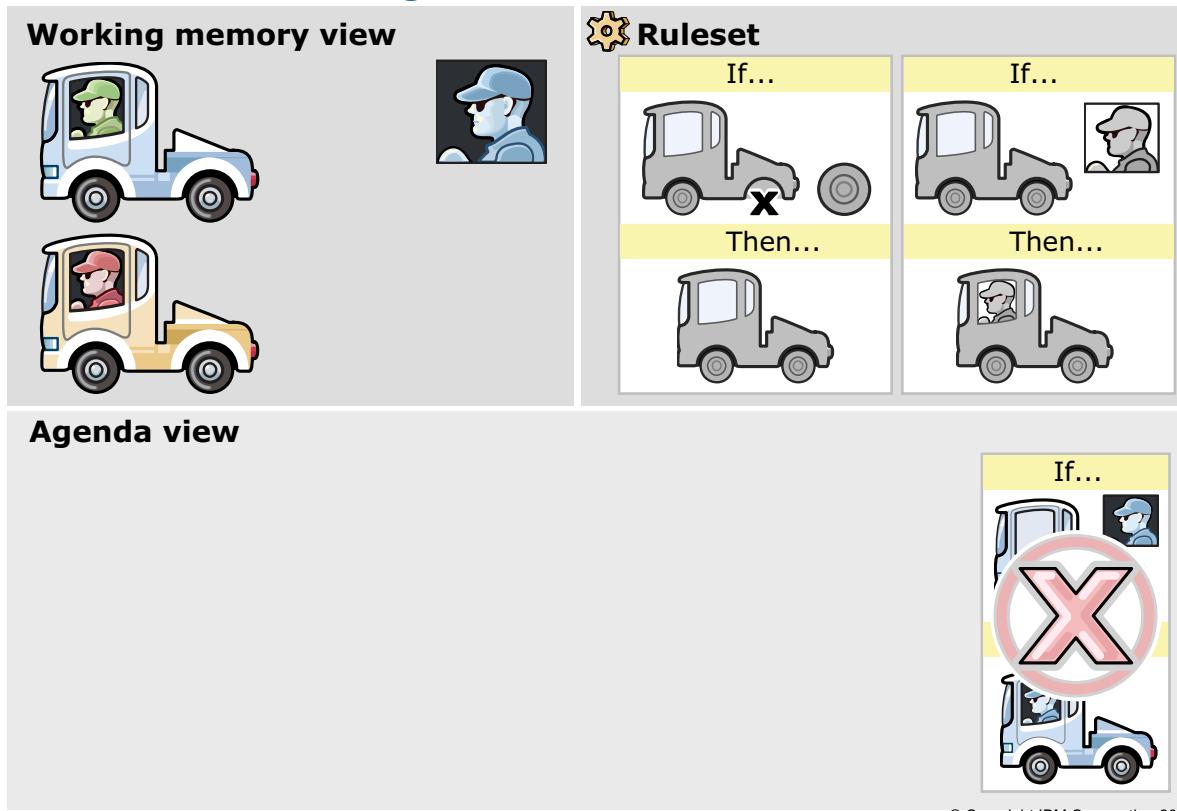
Figure 3-36. AssignDriver executes again

WB3951.2

Notes:

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



© Copyright IBM Corporation 2016

Figure 3-37. Side effects of AssignDriver

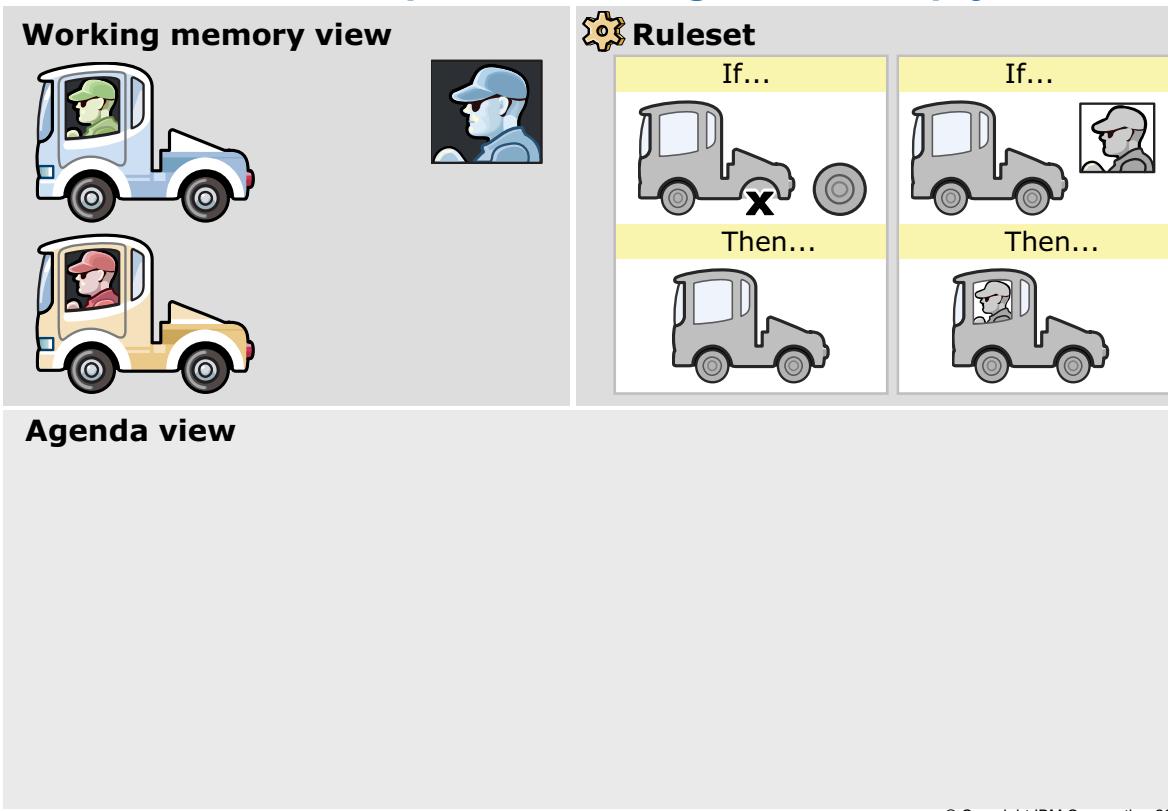
WB3951.2

Notes:

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



© Copyright IBM Corporation 2016

Figure 3-38. Rule execution completes when agenda is empty

WB3951.2

Notes:

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



Unit summary

Having completed this unit, you should be able to:

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

© Copyright IBM Corporation 2016

Figure 3-39. Unit summary

WB3951.2

Notes:



Checkpoint questions

1. **True or False:** ODM provides two rule engines: the classic rule engine or the decision engine.
2. The rule engine can use these modes of execution. Choose all that apply.
 - a. RetePlus
 - b. Fastpath
 - c. Sequential
3. **True or False:** Ruleset parameters provide the means to exchange data between the ruleset and the application.
4. **True or False:** The default rule engine mode is RetePlus.
5. **True or False:** The rule engine uses pattern matching to test rule conditions against objects that are passed from the application.

© Copyright IBM Corporation 2016

Figure 3-40. Checkpoint questions

WB3951.2

Notes:

Write your answers here:

- 1.
- 2.
- 3.
- 4.
- 5.



Checkpoint answers

- 1. True.**
- 2. The rule engine can use these modes of execution.**
 - a. RetePlus
 - b. Fastpath
 - c. Sequential
- 3. True.**
- 4. False: The default rule engine mode is Fastpath.**
- 5. True.**

© Copyright IBM Corporation 2016

Figure 3-41. Checkpoint answers

WB3951.2

Notes:

Unit 4. Developing object models

What this unit is about

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.

What you should be able to do

After completing this unit, you should be able to:

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

How you will check your progress

- Checkpoint
- Exercises



Unit objectives

After completing this unit, you should be able to:

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

© Copyright IBM Corporation 2016

Figure 4-1. Unit objectives

WB3951.2

Notes:



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 2016

Figure 4-2. Topics

WB3951.2

Notes:

4.1. Designing the models

Designing the models



© Copyright IBM Corporation 2016

Course materials may not be reproduced in whole or in part without the prior written permission of IBM.

10.1

Figure 4-3. Designing the models

WB3951.2

Notes:

Introduction to the models

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

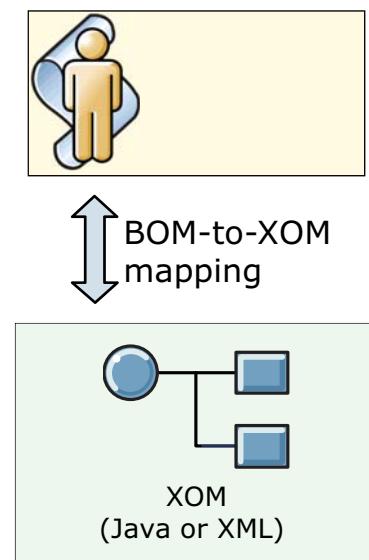


Figure 4-4. Introduction to the models

© Copyright IBM Corporation 2016

WB3951.2

Notes:

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

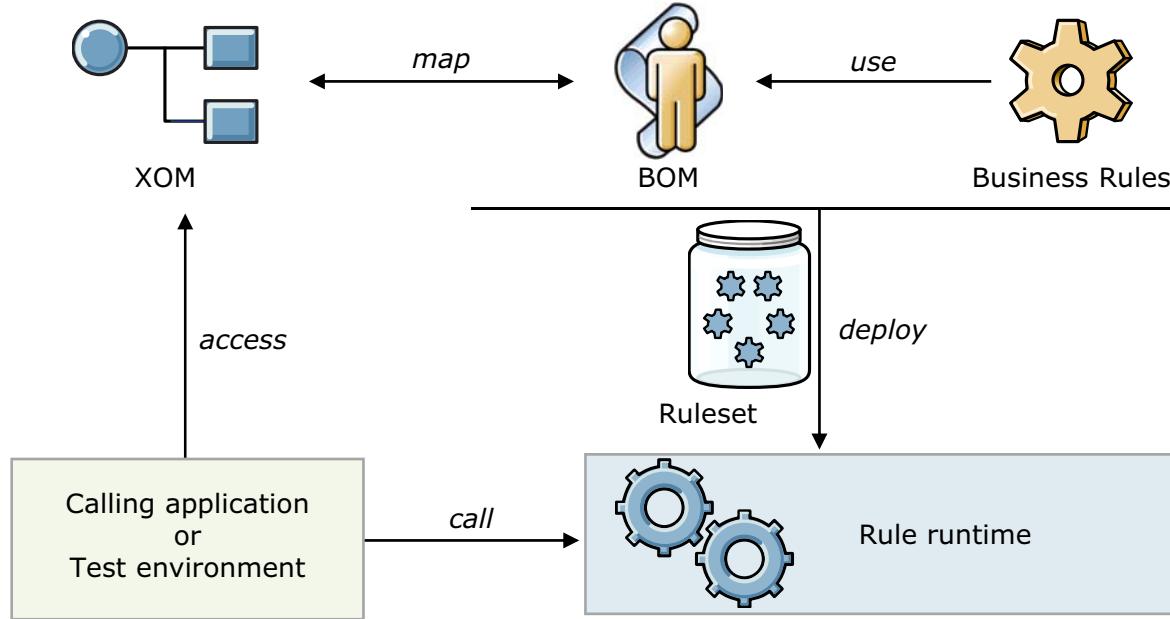
The BOM defines the entities that are used in rule artifacts, and is the model against which rules are written.

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



© Copyright IBM Corporation 2016

Figure 4-5. BOM and XOM at run time

WB3951.2

Notes:

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
- 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
- Top-down
 - Start by first implementing the BOM in Rule Designer to provide vocabulary for rule editors
 - Implement the underlying execution code later
- Development of the BOM and XOM can take several iterations

© Copyright IBM Corporation 2016

Figure 4-6. Designing the BOM and the XOM

WB3951.2

Notes:

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

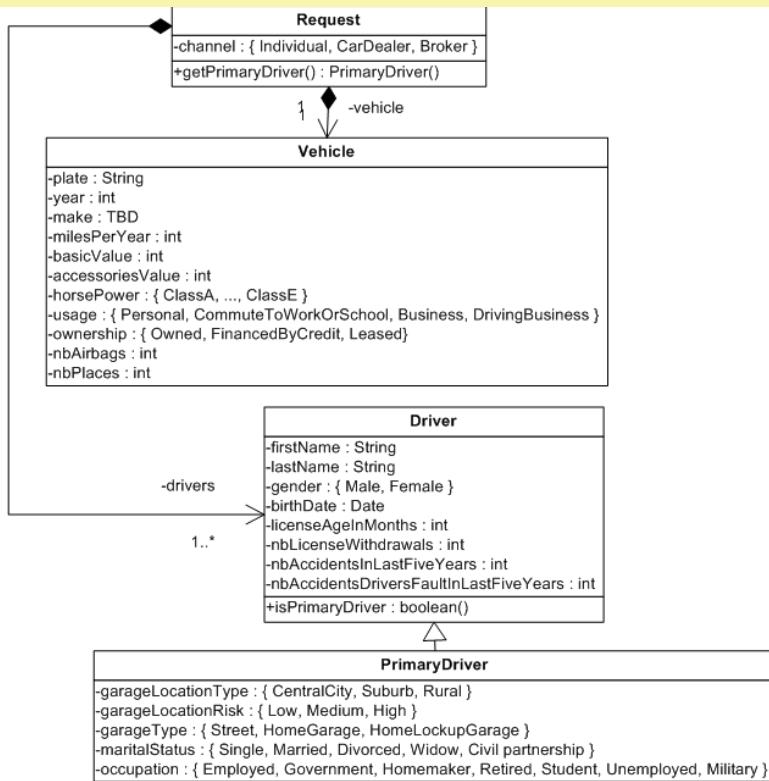
The starting point is based on the object models that the business analysts provide as vocabulary requirements. Generally, you would use those requirements to implement the code XOM first and then generate the BOM automatically. This bottom-up approach is fully supported by ODM.

It is also possible to build a BOM first to prototype rule creation, and implement the XOM later.

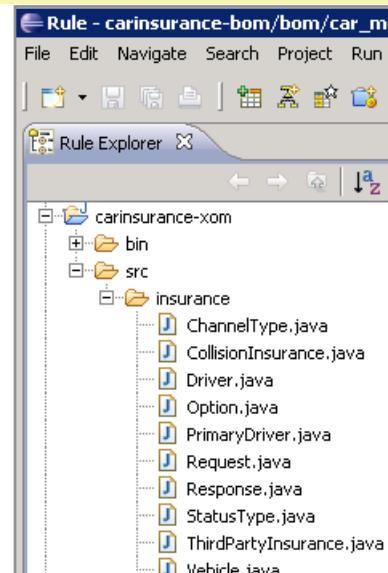
With either approach, the BOM and the XOM evolve in parallel. Changes to the vocabulary require updates to the BOM and possibly the XOM, so expect several iterations before the BOM and XOM are stable.

Example: Car insurance class diagram

Class model in Visio to represent vocabulary requirements



Execution Object Model (XOM)
in Rule Designer



© Copyright IBM Corporation 2016

Figure 4-7. Example: Car insurance class diagram

WB3951.2

Notes:

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.

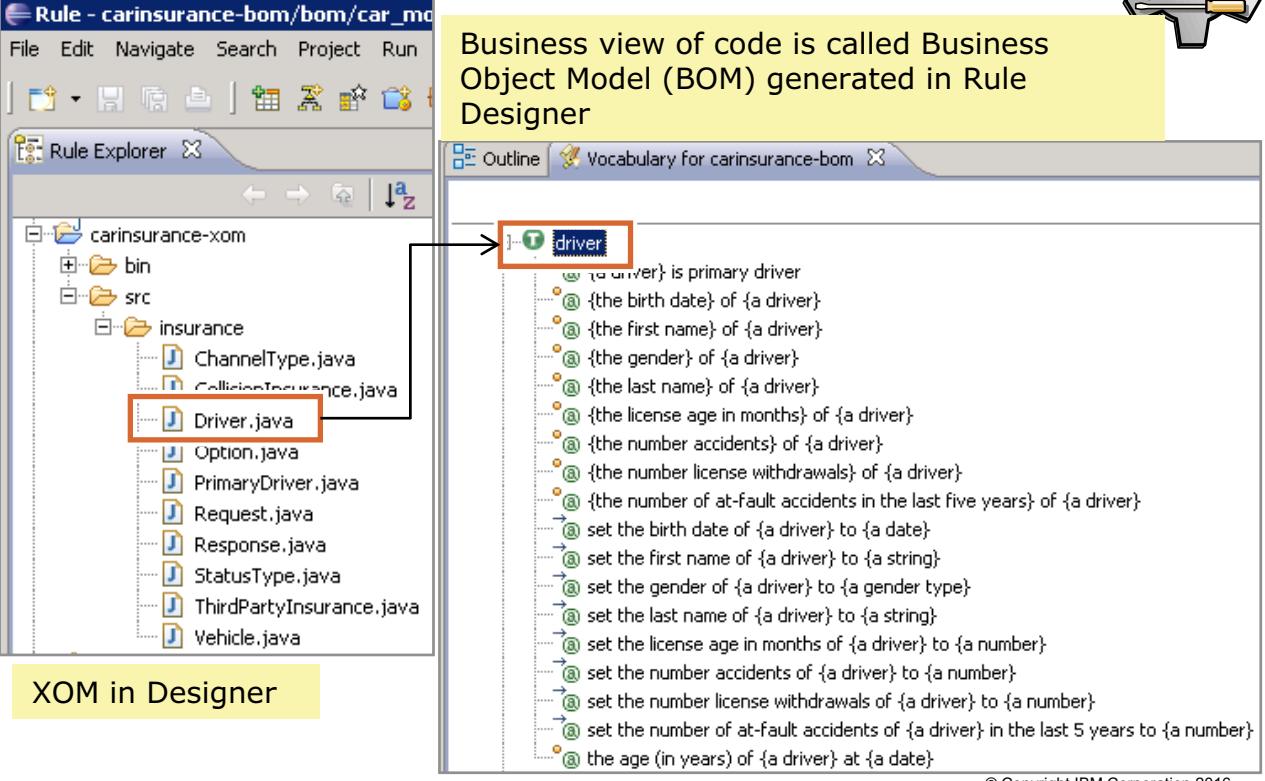
The business analysts might not always provide such a thorough model for vocabulary, but the more complete the model is, the smoother the implementation. The ODM BA class helps business analysts learn the basics of UML class diagrams so they can effectively communicate the vocabulary requirements to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




Example: Implementation of insurance model



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



XOM in Designer

© Copyright IBM Corporation 2016

Figure 4-8. Example: Implementation of insurance model

WB3951.2

Notes:

Here, from the same `Driver.java` class, you can see the list of natural language phrases that business users can read and work with. This *business view* of the code is called the Business Object Model (BOM), which includes the natural language vocabulary layer. The BOM vocabulary is what the business users see when they write rules in the rule editors.

WebSphere Education

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**" ;

© Copyright IBM Corporation 2016

Figure 4-9. Example: Implementation of insurance model

WB3951.2

Notes:

From the XOM, Rule Designer can *automatically* generate a business view of the code that is called the Business Object Model (BOM), which includes the natural language vocabulary layer. The BOM vocabulary is what the business users see when they write rules in the rule editors.

4.2. 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



© Copyright IBM Corporation 2016

Course materials may not be reproduced in whole or in part without the prior written permission of IBM.

10.1

Figure 4-10. Defining business and execution object models

WB3951.2

Notes:

Business object model (BOM)

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

© Copyright IBM Corporation 2016

Figure 4-11. Business object model (BOM)

WB3951.2

Notes:

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

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

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

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

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

BOM entries

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



© Copyright IBM Corporation 2016

Figure 4-12. BOM entries

WB3951.2

Notes:

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

- `.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

© Copyright IBM Corporation 2016

Figure 4-13. BOM path

WB3951.2

Notes:

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

© Copyright IBM Corporation 2016

Figure 4-14. Execution object model (XOM)

WB3951.2

Notes:

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

© Copyright IBM Corporation 2016

Figure 4-15. Types of XOM

WB3951.2

Notes:

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



Rule project and XOM

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

© Copyright IBM Corporation 2016

Figure 4-16. Rule project and XOM

WB3951.2

Notes:

4.3. Editing the business object model

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

Editing the business object model



© Copyright IBM Corporation 2016

Course materials may not be reproduced in whole or in part without the prior written permission of IBM.

10.1

Figure 4-17. Editing the business object model

WB3951.2

Notes:



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

© Copyright IBM Corporation 2016

Figure 4-18. Introduction

WB3951.2

Notes:

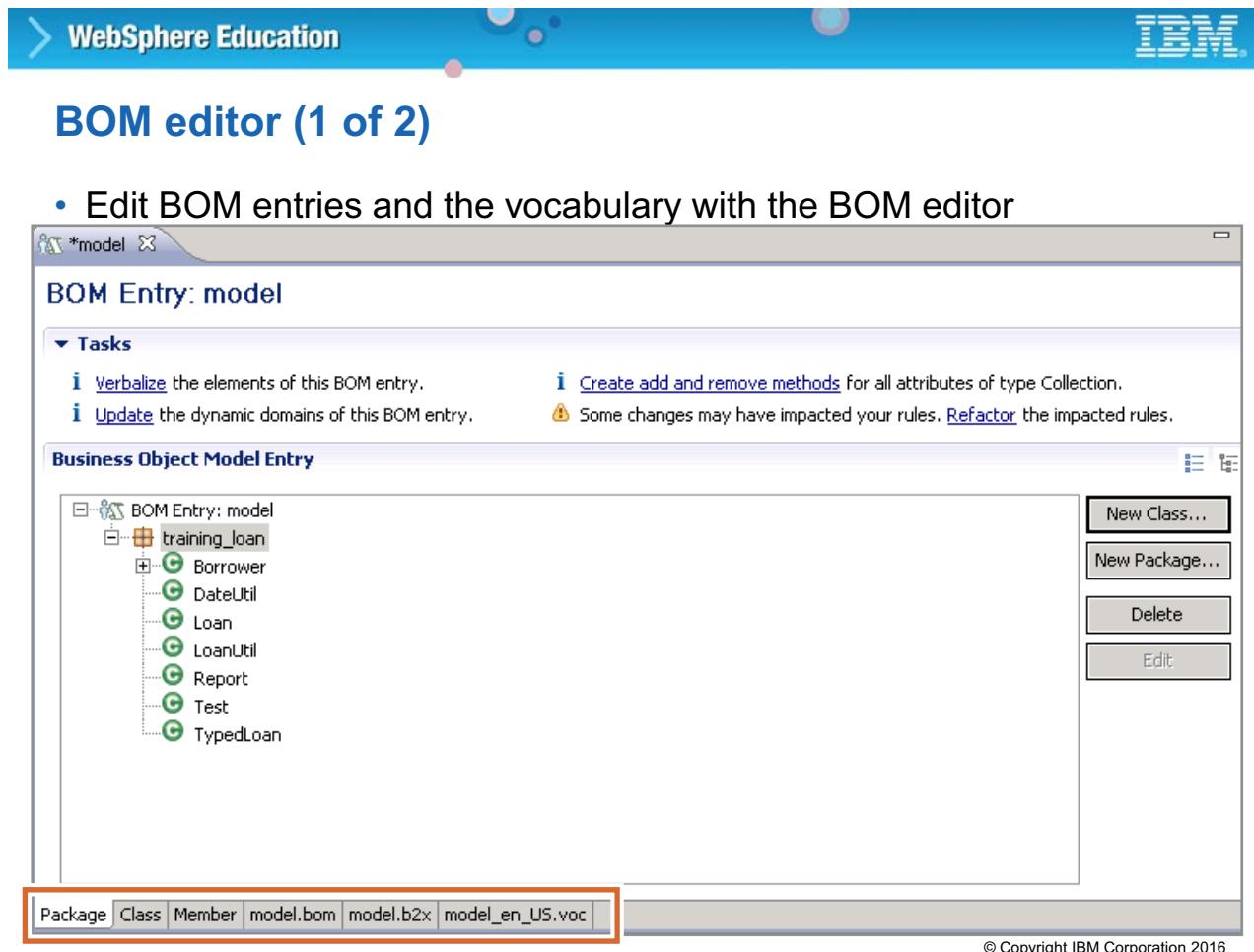


Figure 4-19. BOM editor (1 of 2)

WB3951.2

Notes:

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)

Class Borrower (package: training_loan)

General Information

- Name: Borrower
- Namespace: training_loan
- Superclasses: java.lang.Object, java.io.Serializable
- Interfaces:
- Deprecated

Class Verbalization

- Remove the verbalization.
- Edit the documentation.
- Generate automatic variable
- Term: borrower
- Edit term.
- the borrower, a borrower, the borrowers....

Members

Specify the members of this class.

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

Domain

Create and edit a domain for this class.

Create a domain.

Categories

Define the categories associated with this class.

Edit the categories.

Any

BOM to XOM Mapping

Package Class Member model.bom model.b2x model_en_US.voc

© Copyright IBM Corporation 2016

Figure 4-20. BOM editor (2 of 2)

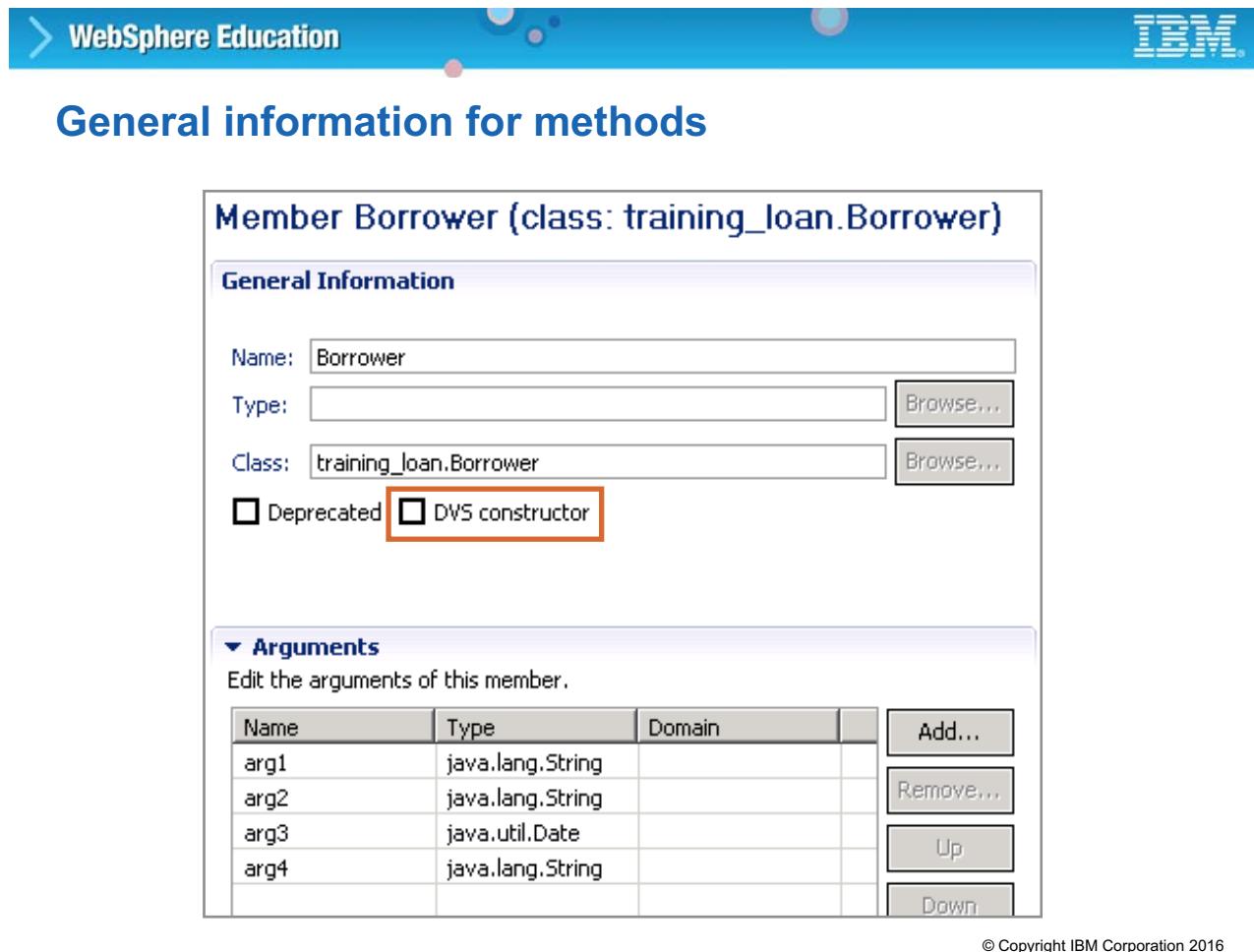
WB3951.2

Notes:

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.



The screenshot shows a user interface for managing a Java class named 'Member Borrower' (class: training_loan.Borrower). The interface is divided into sections: 'General Information' and 'Arguments'. In the 'General Information' section, fields include 'Name' (Borrower), 'Type' (with a 'Browse...' button), 'Class' (training_loan.Borrower with a 'Browse...' button), and two checkboxes: 'Deprecated' and 'DVS constructor' (the latter is highlighted with a red box). The 'Arguments' section contains a table with four rows, each representing an argument: arg1 (String), arg2 (String), arg3 (Date), and arg4 (String). To the right of the table are buttons for 'Add...', 'Remove...', 'Up', and 'Down'.

© Copyright IBM Corporation 2016

Figure 4-21. General information for methods

WB3951.2

Notes:

The General Information section for methods that are used as constructors also includes the **DVS 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.



General information for attributes

Member age (class: training_loan.Borrower)

General Information

Name:	age	Browse...
Type:	int	Browse...
Class:	training_loan.Borrower	Browse...
<input type="radio"/> Read/Write <input checked="" type="radio"/> Read Only <input type="radio"/> Write Only <input type="checkbox"/> Static <input type="checkbox"/> Final <input type="checkbox"/> Deprecated <input type="checkbox"/> Update object state <input type="checkbox"/> Ignore for DVS		

© Copyright IBM Corporation 2016

Figure 4-22. General information for attributes

WB3951.2

Notes:

The General Information section for BOM attributes includes these fields:

- **Name, Type, and Class:** These properties define the BOM element.
- **Read Only, Write Only, and Read/Write** (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 DVS:** By default, a DVS Excel 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 DVS Excel scenario file template.
- **DVS constructor** (for constructor methods only): The **DVS constructor** property indicates whether the required columns in the DVS scenario file template must be created according to the arguments of this constructor.



Note

The **Ignore for DVS** and **DVS constructor** properties relate to tests and simulations in Decision Validation Services. You learn more about Decision Validation Services in Unit 10, "Enabling tests and simulations".

4.4. 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



© Copyright IBM Corporation 2016

Course materials may not be reproduced in whole or in part without the prior written permission of IBM.

10.1

Figure 4-23. Mapping the BOM to the XOM

WB3951.2

Notes:



Introduction

- BOM-to-XOM mapping (B2X) is the correspondence between the BOM and the XOM
- This correspondence is required to translate the rule artifacts that are based on the BOM into rule artifacts that are based on the XOM
- You can edit the B2X in the **BOM to XOM Mapping** section of the BOM editor

© Copyright IBM Corporation 2016

Figure 4-24. Introduction

WB3951.2

Notes:

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.

Therefore, rule artifacts that are written in terms of elements in the BOM are interpreted in terms of elements in the XOM.

You can edit the BOM-to-XOM mapping in the **BOM to XOM Mapping** section of the BOM editor.



Note

IBM ODM on Cloud

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



BOM to XOM Mapping section in the BOM editor

- Use to define the correspondence between the BOM member (class, attribute, method) and the XOM

The screenshot shows the 'BOM to XOM Mapping' section of the BOM editor. It includes a toolbar with a pencil icon labeled 'Edit the imports.', a 'Getter' section containing code, and a 'Setter' section.

```

return DateUtil.getAge(this.getBirthDate(), DateUtil.now());
  
```

© Copyright IBM Corporation 2016

Figure 4-25. BOM to XOM Mapping section in the BOM editor

WB3951.2

Notes:

In the **BOM to XOM Mapping** section of the BOM editor, you can define the correspondence between a BOM member (class, attribute, method) and the XOM. Depending on the selected BOM element, the available properties vary.

You learn more about BOM-to-XOM mapping later in this unit.



Types of BOM-to-XOM mapping

- Default mapping
 - The BOM element maps to the XOM element with the same fully qualified name
- ILOG Rule Language (IRL) mapping
 - An IRL function that is based on the XOM is associated to the BOM element
 - IRL is an executable rule language
- Extender mapping
 - Using a Java extender class, with a static element with the same name as the BOM element
- At run time, the rule engine searches for the right mapping in this order:
 - Explicit IRL mapping
 - Explicit extender mapping
 - Implicit mapping

© Copyright IBM Corporation 2016

Figure 4-26. Types of BOM-to-XOM mapping

WB3951.2

Notes:

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 also supported.

- **IRL mapping:** With IRL (ILOG Rule Language) mapping, you associate an IRL function that is based on the XOM with a business element. (You learn more about IRL later in this course.)
- **Extender mapping:** With extender mapping in a Java extender class, you create static elements that have the same name as business class members.

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**

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 conversion to IRL is skipped, and 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, see the product documentation.



Use of explicit mappings

- Add a BOM virtual member without touching the XOM
- Quickly prototype the BOM on an existing XOM
- Rebind the BOM on a modified XOM to avoid modifying the rules

© Copyright IBM Corporation 2016

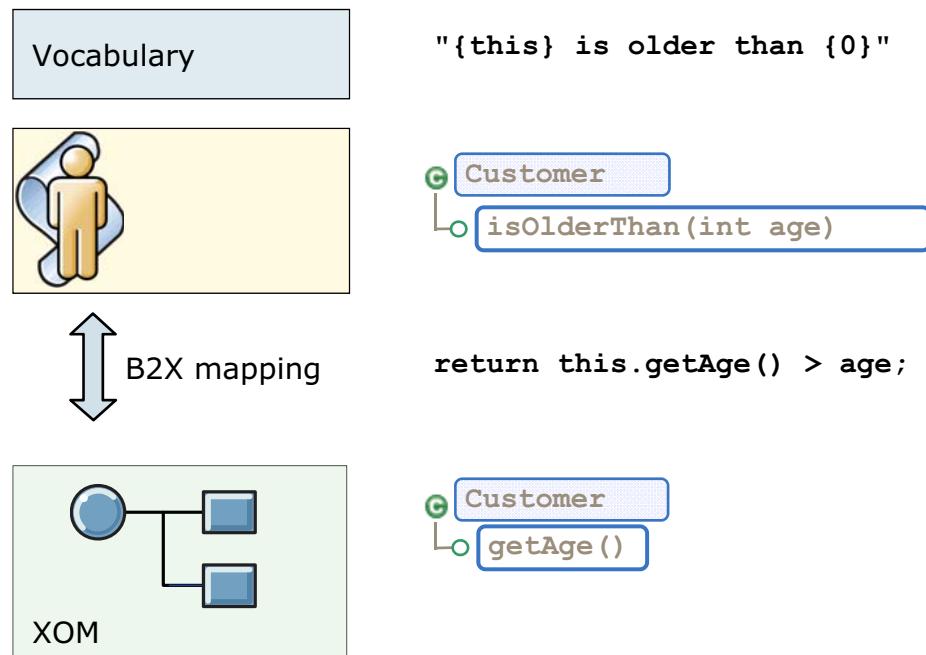
Figure 4-27. Use of explicit mappings

WB3951.2

Notes:

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

Explicit IRL mapping



© Copyright IBM Corporation 2016

Figure 4-28. Explicit IRL mapping

WB3951.2

Notes:

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

1. In the Outline view, click the business element that you want to map.
2. In the BOM editor, in the **BOM to XOM Mapping** section, define the mapping of the element in IRL by basing your function body on the XOM.

As shown in this example, consider that the class Customer in the XOM has the `getAge` method to get the age of a customer:

```
public class Customer { public int getAge() ... }
```

To have a rule that checks whether the customer is older than a certain age, you can create a predicate in the class Customer, and pass the age as a parameter to the method:

```
public class Customer { public boolean isOlderThan(int age); }
```

The body of the BOM-to-XOM mapping, which is written in IRL, can then be expressed as:

```
return this.getAge() > age;
```



Explicit extender mapping

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

© Copyright IBM Corporation 2016

Figure 4-29. Explicit extender mapping

WB3951.2

Notes:

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.

4.5. Working with the vocabulary

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

Working with the vocabulary



© Copyright IBM Corporation 2016

Course materials may not be reproduced in whole or in part without the prior written permission of IBM.

10.1

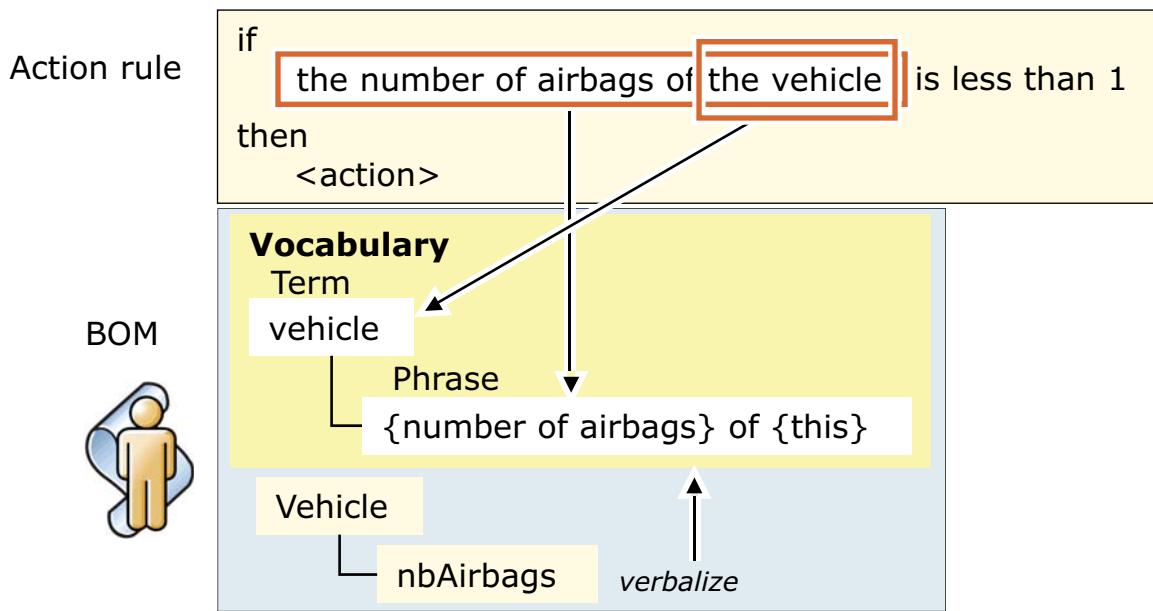
Figure 4-30. Working with the vocabulary

WB3951.2

Notes:

Rule vocabulary

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



© Copyright IBM Corporation 2016

Figure 4-31. Rule vocabulary

WB3951.2

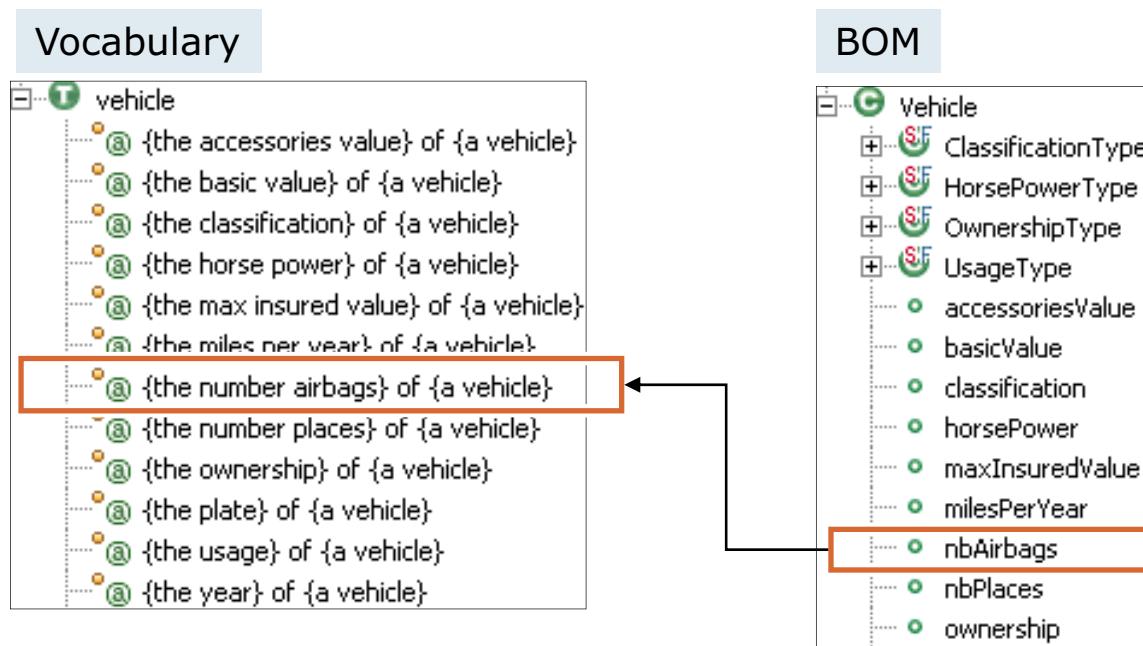
Notes:

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



© Copyright IBM Corporation 2016

Figure 4-32. Vocabulary and verbalization

WB3951.2

Notes:

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

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

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



Verbalization in the BOM editor

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

The screenshot shows a dialog box titled "Member Verbalization". It contains three items:

- Remove the verbalization.** (with a red X icon)
- Create a navigation phrase.** (with a green plus icon)
- Edit the subject used in phrases.** (with a pencil icon)

Below this is another section titled "Navigation : 'the amount of a loan'" with a red X icon to its right. It contains a template field with the placeholder "Template: {amount} of {this}" and a "Delete" button (trash icon) to its right.

© Copyright IBM Corporation 2016

Figure 4-33. Verbalization in the BOM editor

WB3951.2

Notes:

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)

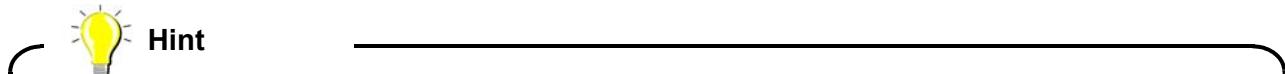
- Verbalizations are included in the vocabulary lists of rule editors
 - BOM must be defined and verbalized before you can start writing rules in the rule editors
 - Only verbalized business elements are accessible in the rule editors, and can be used in your business rules
- Rule Designer can provide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

© Copyright IBM Corporation 2016

Figure 4-34. Verbalization (1 of 2)

WB3951.2

Notes:



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}

© Copyright IBM Corporation 2016

Figure 4-35. Verbalization (2 of 2)

WB3951.2

Notes:

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: { }

© Copyright IBM Corporation 2016

Figure 4-36. Placeholders

WB3951.2

Notes:



Verbalizing BOM members

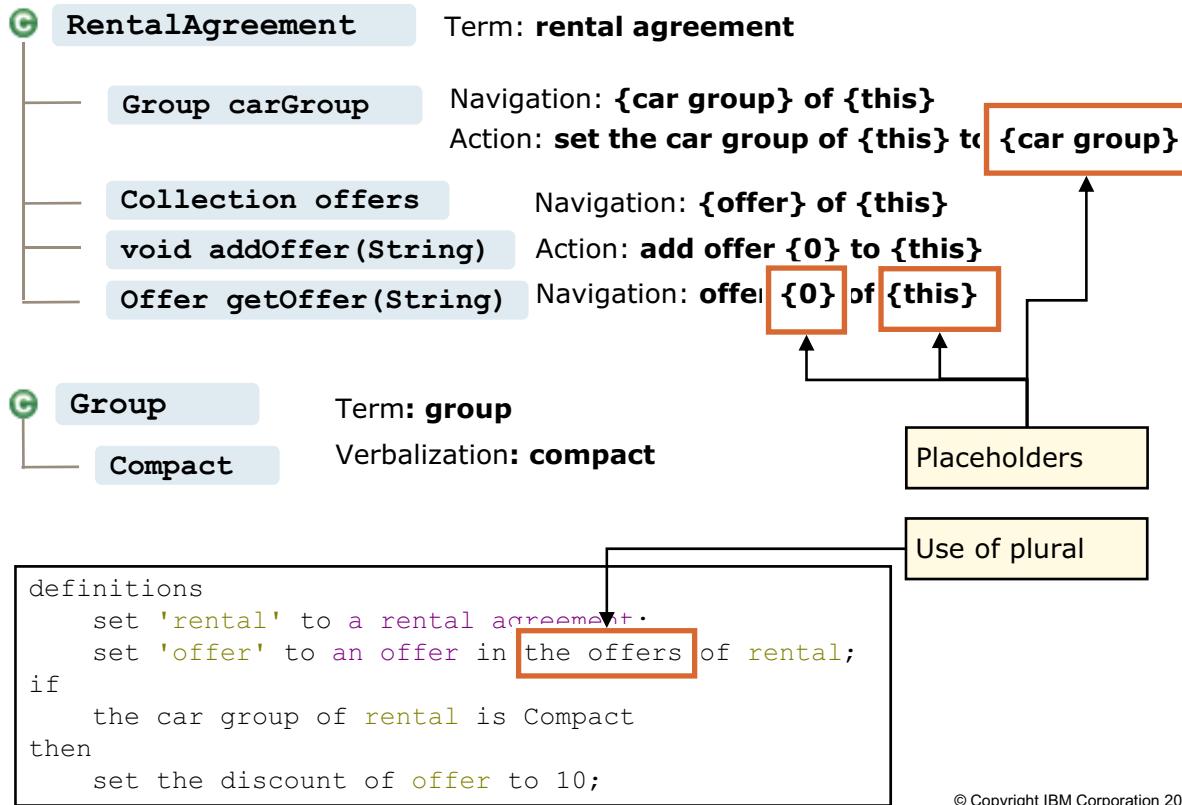


Figure 4-37. Verbalizing BOM members

WB3951.2

Notes:

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 phase "`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".

4.6. Refactoring

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



Refactoring



© Copyright IBM Corporation 2016

Course materials may not be reproduced in whole or in part without the prior written permission of IBM.

10.1

Figure 4-38. Refactoring

WB3951.2

Notes:

Refactoring

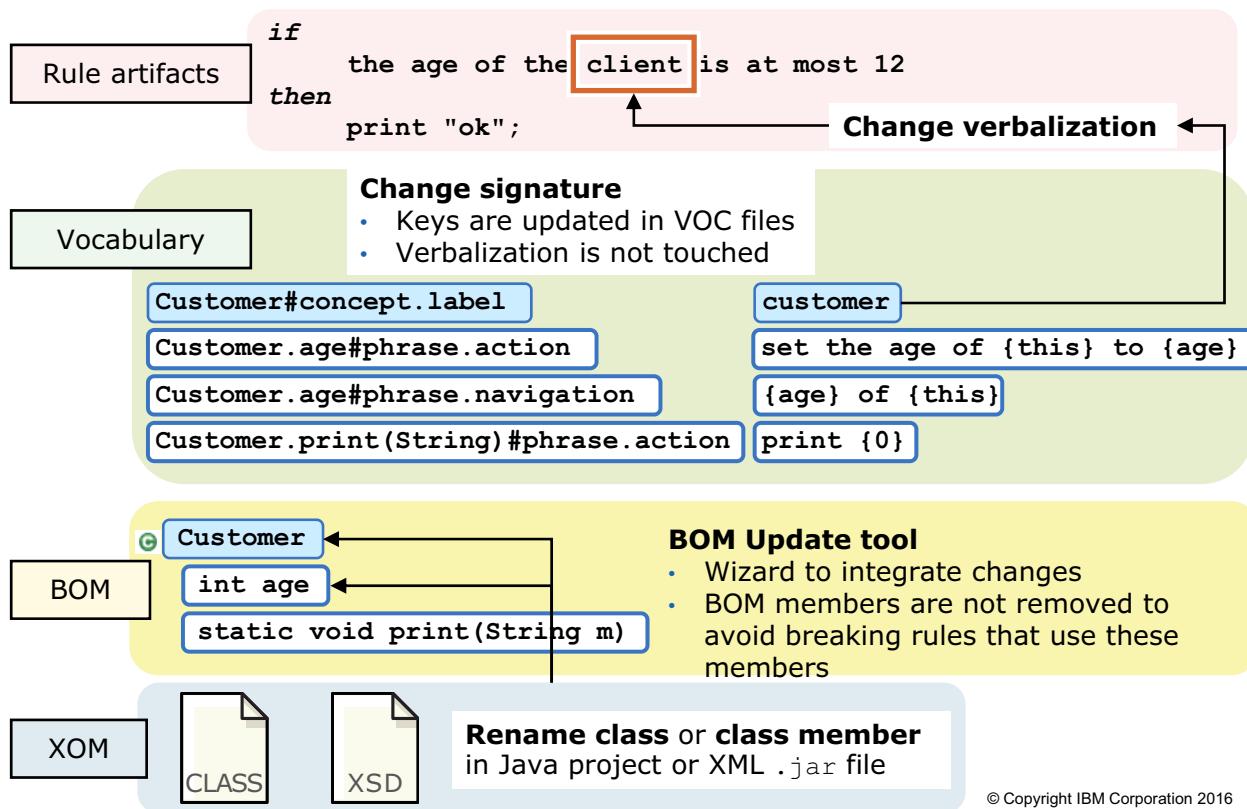


Figure 4-39. Refactoring

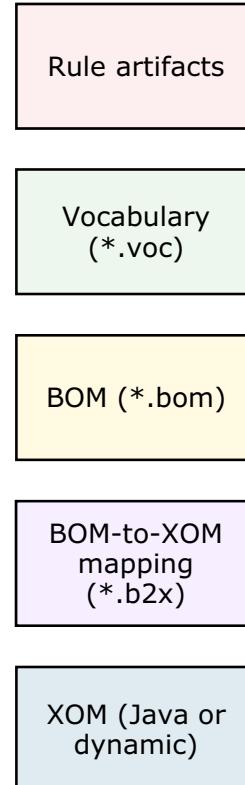
WB3951.2

Notes:



From the XOM to the rules

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



© Copyright IBM Corporation 2016

Figure 4-40. From the XOM to the rules

WB3951.2

Notes:

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

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



BOM and XOM evolution

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

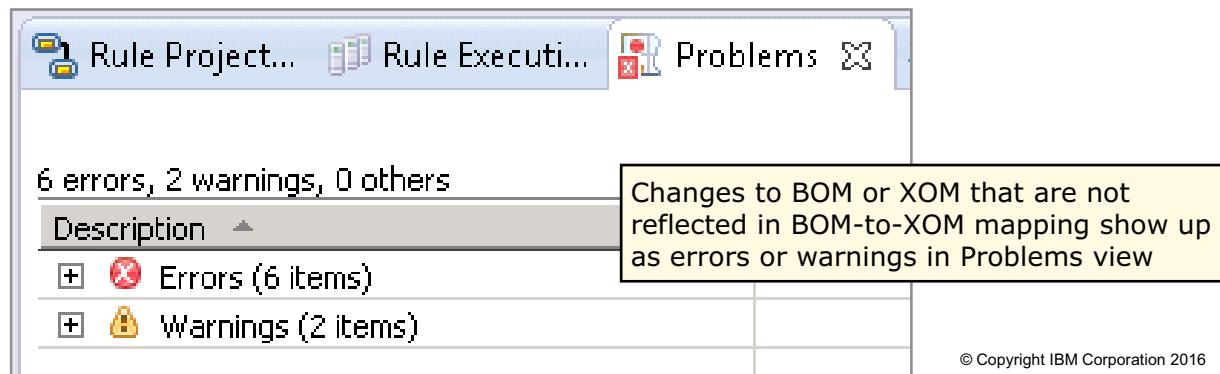


Figure 4-41. BOM and XOM evolution

WB3951.2

Notes:

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

© Copyright IBM Corporation 2016

Figure 4-42. XOM changes (1 of 3)

WB3951.2

Notes:

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.

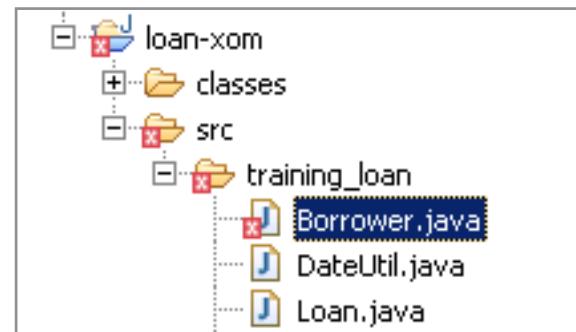
XOM changes (2 of 3)

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



© Copyright IBM Corporation 2016

Figure 4-43. XOM changes (2 of 3)

WB3951.2

Notes:

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

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



XOM changes (3 of 3)

- Renaming done manually: Consider such renaming as an addition or a removal, and map the BOM member name to the new XOM member name so as not to break existing rules
- Renaming is done through the refactor menu, with the BOM project open; the change is propagated to the BOM and the appropriate part of the vocabulary
- Other kinds of refactoring:
 - As much as possible, use **BOM Update** to propagate changes
 - Fix the rest manually

© Copyright IBM Corporation 2016

Figure 4-44. XOM changes (3 of 3)

WB3951.2

Notes:

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

© Copyright IBM Corporation 2016

Figure 4-45. BOM changes

WB3951.2

Notes:



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

© Copyright IBM Corporation 2016

Figure 4-46. Vocabulary changes

WB3951.2

Notes:

- 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

Having completed this unit, you should be able to:

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

© Copyright IBM Corporation 2016

Figure 4-47. Unit summary

WB3951.2

Notes:



Checkpoint 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

© Copyright IBM Corporation 2016

Figure 4-48. Checkpoint questions

WB3951.2

Notes:

Write your answers here:

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

Checkpoint answers

1. **True:** The BOM is the business view of the model that defines the actions and entities that are used in business rule artifacts.
2. **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:** The XOM is the model against which rules are executed.
4. How do you define the translation of BOM elements into executable elements?
 - c. With the BOM editor by using B2X

© Copyright IBM Corporation 2016

Figure 4-49. Checkpoint answers

WB3951.2

Notes:

Exercise 3



Working with the BOM

© Copyright IBM Corporation 2016

Course materials may not be reproduced in whole or in part without the prior written permission of IBM.

10.1

Figure 4-50. Exercise 3

WB3951.2

Notes:

Exercise objectives

After completing this exercise, you should be able to:

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

© Copyright IBM Corporation 2016

Figure 4-51. Exercise objectives

WB3951.2

Notes:

Exercise 4



© Copyright IBM Corporation 2016

Course materials may not be reproduced in whole or in part without the prior written permission of IBM.

10.1

Figure 4-52. Exercise 4

WB3951.2

Notes:



Exercise objectives

After completing this exercise, you should be able to:

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

© Copyright IBM Corporation 2016

Figure 4-53. Exercise objectives

WB3951.2

Notes:

Unit 5. Orchestrating ruleset execution

What this unit is about

This unit describes how to orchestrate rule execution through ruleflows. You also learn about rule engine execution modes.

What you should be able to do

After completing this unit, you should be able to:

- 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

How you will check your progress

- Checkpoint
- Exercise



Unit objectives

After completing this unit, you should be able to:

- 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 2016

Figure 5-1. Unit objectives

WB3951.2

Notes:



Topics

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

© Copyright IBM Corporation 2016

Figure 5-2. Topics

WB3951.2

Notes:

5.1. Controlling rule execution: Overview

Controlling rule execution: Overview



© Copyright IBM Corporation 2016

Course materials may not be reproduced in whole or in part without the prior written permission of IBM.

10.1

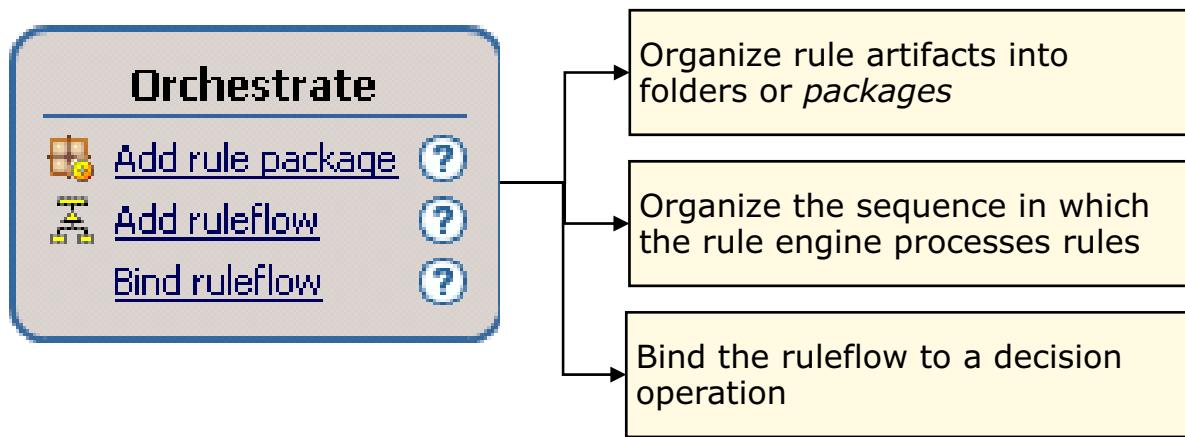
Figure 5-3. Controlling rule execution: Overview

WB3951.2

Notes:

What is orchestration?

- Rules implement business policy, but have no notion of sequence
- Ruleflows* organize rules into a series of smaller decisions and define a routing logic to produce the overall decision
- Operation Map



© Copyright IBM Corporation 2016

Figure 5-4. What is orchestration?

WB3951.2

Notes:

The ruleset groups all the rules that are involved in producing a single business decision, and within the context of the ruleset, a ruleflow enforces the sequence in which rules are selected. You can also use the ruleflow to control *how* the rule engine processes the rules.

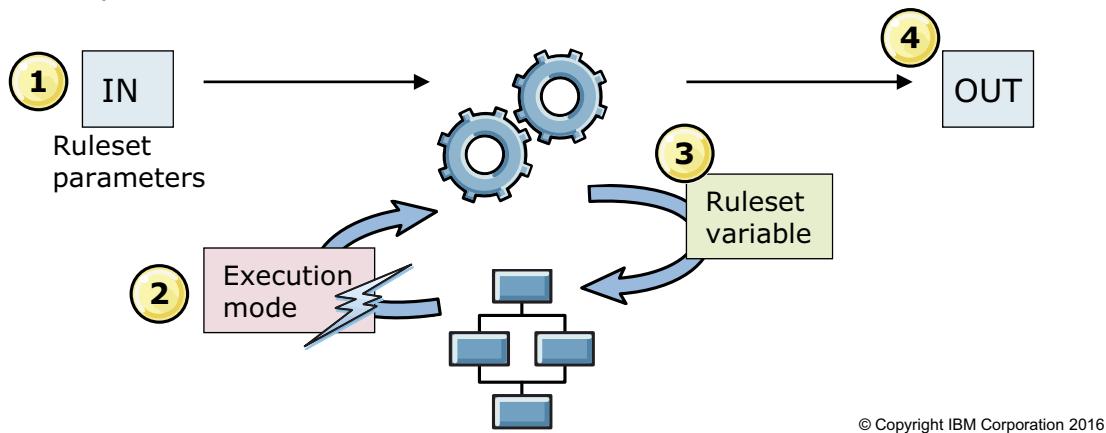
Rule packages organize the rules logically as a series of smaller decisions that lead to the overall business decision.

A routing logic can determine the appropriate sequence of evaluation from a set of possible paths through the ruleset.

Key elements for orchestration

The ruleflow coordinates rule selection and execution within the ruleset

1. Ruleset parameters are passed (with the ruleset) to the rule engine
2. Based on information in the ruleflow, the rule engine knows which rules to select and what algorithm to use to evaluate the rules
3. During the evaluation, intermediate values or computations are stored in ruleset variables and can be transferred from one task to the next
4. The rule engine's evaluation results are returned to the calling application in a ruleset parameter



© Copyright IBM Corporation 2016

Figure 5-5. Key elements for orchestration

WB3951.2

Notes:

To outline the flow of rule execution and produce the correct decision, you need to work with the business analysts and policy managers. Together you identify the type of information that is required to make the decision, and what type of information should be included as part of the decision result.

The key elements in your decision service that are required to orchestrate execution of your business rules include: ruleflows, rule packages, rulesets, ruleset parameters, and ruleset variables. You work with each of these artifacts during the exercise.

5.2. Designing ruleflows

Designing ruleflows



© Copyright IBM Corporation 2016

Course materials may not be reproduced in whole or in part without the prior written permission of IBM.

10.1

Figure 5-6. Designing ruleflows

WB3951.2

Notes:

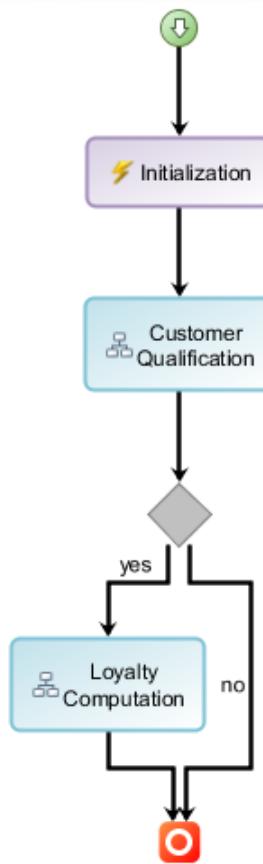
You learned about the concepts that are required to orchestrate the execution of the rules in your decision service.

You now learn how to graphically design ruleflows and set their properties by using the Ruleflow Editor in Rule Designer.



Ruleflows

- Represent the business logic
- Sequences of rules that are grouped in tasks
- Organize rules into a sequence of decisions to produce a single decision



© Copyright IBM Corporation 2016

Figure 5-7. Ruleflows

WB3951.2

Notes:

Here you see an example ruleflow. The ruleflow represents the application business logic. It defines the rule sequence and rule selection at a high level. In the ruleflow, you define smaller decisions as ruleflow *tasks*. You specify the *transitions* between the tasks and define the conditions under which these transitions are executed.

With the ruleflow, you do not have to write procedural code to control the flow of execution of your business rules. You construct ruleflows graphically in a Ruleflow editor and specify how tasks are chained together.

When objects are passed from the client application to the rule engine, the rule engine evaluates those objects against the rules in each task according to sequence defined in the ruleflow. Based on the results after evaluating one task, the routing logic determines which task to evaluate next. The final decision result, after the ruleflow completes, is then returned to the client application.

Objects can be passed to the rule engine as ruleset parameters or be accessed from the working memory. Within the ruleflow, intermediate results can be passed between tasks as ruleset variables.

A decision service can contain several ruleflows. However, your ruleset must identify one ruleflow as the *main* ruleflow for the project. Additional ruleflows are accessed as subflow tasks.

Within the ruleflow, you can also define properties to control how the rules are selected and evaluated at run time.

Ruleflow elements



- One start node

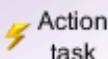


- One or more end nodes

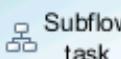
- Tasks:



- **Rule task:** Set of rules to be executed at that point in the ruleflow
You can specify rule order, rule selection, and execution modes on each task



- **Action task:** Rule action statements in BAL or IRL code that are executed each time that the task is executed



- **Subflow task:** Reference to another ruleflow in the decision service or in a referenced rule project



- Transitions connect the tasks in a ruleflow and define the sequence of the ruleflow from one task to another

© Copyright IBM Corporation 2016

Figure 5-8. Ruleflow elements

WB3951.2

Notes:



Rule tasks

- A rule task should correspond to a rule package
- You can add or remove a rule in the package without modifying the ruleflow
 - All changes to the rule package are included the next time that the ruleset is deployed

© Copyright IBM Corporation 2016

Figure 5-9. Rule tasks

WB3951.2

Notes:



Initial and final actions

- Initial and final actions are optional
- At the ruleflow level
 - Initial actions: Specified in the start node and executed before the rest of the ruleflow
 - Final actions: Specified in the end nodes and executed after the rest of the ruleflow
 - The final actions are the same for all end nodes
- At the task level
 - Initial actions: Executed before the task body
 - Final actions: Executed after the task body

© Copyright IBM Corporation 2016

Figure 5-10. Initial and final actions

WB3951.2

Notes:



Transitions

- Transitions
 - Unidirectional arrows that connect tasks and determine the “flow” of the ruleflow
- Transition conditions
 - Boolean condition for controlling the flow from task to task
- Can include an “**else**” condition
 - For all cases that do not match other conditions in the transition
 - Maximum one “**else**” per transition

© Copyright IBM Corporation 2016

Figure 5-11. Transitions

WB3951.2

Notes:

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

© Copyright IBM Corporation 2016

Figure 5-12. Forks and joins

WB3951.2

Notes:



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 ruleset parameters, of ruleset variables
 - Check initial values of input ruleset parameters
- Typical examples of transition conditions:
 - Test values of ruleset parameters, of ruleset variables

© Copyright IBM Corporation 2016

Figure 5-13. Expression of initial or final actions, and conditions

WB3951.2

Notes:



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`

© Copyright IBM Corporation 2016

Figure 5-14. Main ruleflow

WB3951.2

Notes:

5.3. Controlling rule selection for execution

Controlling rule selection for execution



© Copyright IBM Corporation 2016

Course materials may not be reproduced in whole or in part without the prior written permission of IBM.

10.1

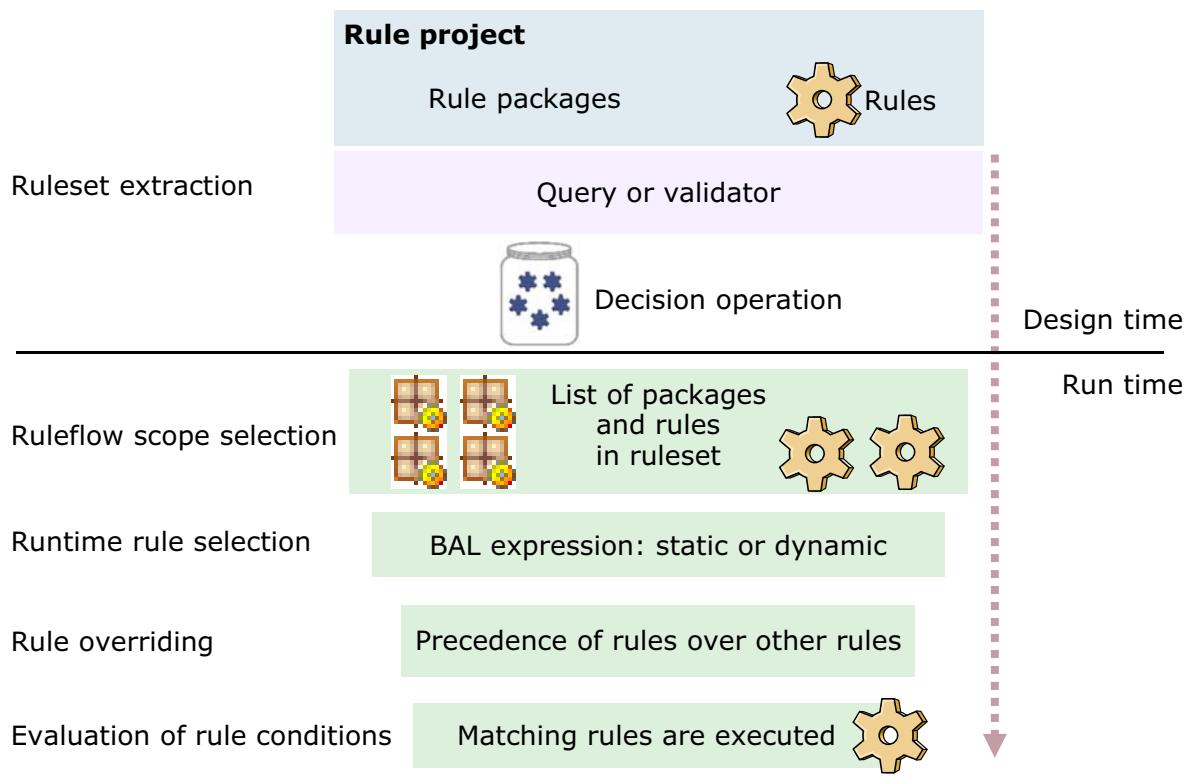
Figure 5-15. Controlling rule selection for execution

WB3951.2

Notes:



Rule selection pipe



© Copyright IBM Corporation 2016

Figure 5-16. Rule selection pipe

WB3951.2

Notes:



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

© Copyright IBM Corporation 2016

Figure 5-17. Ruleflow scope selection

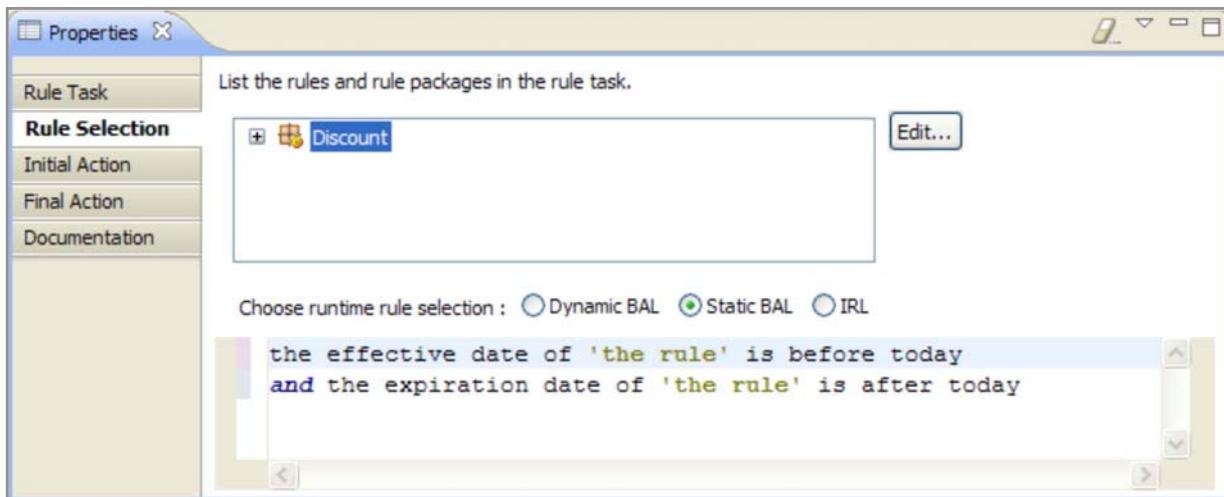
WB3951.2

Notes:



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 = . . .”



© Copyright IBM Corporation 2016

Figure 5-18. Runtime rule selection

WB3951.2

Notes:



Rule hierarchies

- Can be used to further refine rule selection
- Define a tree of values
- Are customizable
- Are typically combined with overriding

© Copyright IBM Corporation 2016

Figure 5-19. Rule hierarchies

WB3951.2

Notes:

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

© Copyright IBM Corporation 2016

Figure 5-20. Rule overriding

WB3951.2

Notes:



Rule condition evaluation

- Conditions of remaining rules are evaluated
- Only rules with matching conditions are fired

© Copyright IBM Corporation 2016

Figure 5-21. Rule condition evaluation

WB3951.2

Notes:

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.

5.4. Controlling rule order during rule execution

Controlling rule order during rule execution



© Copyright IBM Corporation 2016

Course materials may not be reproduced in whole or in part without the prior written permission of IBM.

10.1

Figure 5-22. Controlling rule order during rule execution

WB3951.2

Notes:

You can set further properties to the business rules or to the rule tasks, like the execution mode. In this way, you can give finer control to the execution order among the business rules that are candidates for execution after rules selection.



Introduction

- After the rules are selected, you can more finely control the order in which the rules are fired by setting more properties:
 - Rule priority
 - Rule task execution order
 - Execution modes

© Copyright IBM Corporation 2016

Figure 5-23. Introduction

WB3951.2

Notes:



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

© Copyright IBM Corporation 2016

Figure 5-24. Rule priority

WB3951.2

Notes:



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

© Copyright IBM Corporation 2016

Figure 5-25. Rule task execution order (1 of 2)

WB3951.2

Notes:



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

© Copyright IBM Corporation 2016

Figure 5-26. Rule task execution order (2 of 2)

WB3951.2

Notes:

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

© Copyright IBM Corporation 2016

Figure 5-27. Execution modes

WB3951.2

Notes:



Unit summary

Having completed this unit, you should be able to:

- 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 2016

Figure 5-28. Unit summary

WB3951.2

Notes:

Checkpoint 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

© Copyright IBM Corporation 2016

Figure 5-29. Checkpoint questions

WB3951.2

Notes:

Write your answers here:

- 1.
- 2.
- 3.



Checkpoint answers

1. True.
2. True.
3. Rule tasks use which execution mode as the default?
 - c. Fastpath

© Copyright IBM Corporation 2016

Figure 5-30. Checkpoint answers

WB3951.2

Notes:

Exercise 5



© Copyright IBM Corporation 2016

Course materials may not be reproduced in whole or in part without the prior written permission of IBM.

10.1

Figure 5-31. Exercise 5

WB3951.2

Notes:



Exercise objectives

After completing this exercise, you should be able to:

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

© Copyright IBM Corporation 2016

Figure 5-32. Exercise objectives

WB3951.2

Notes:

Unit 6. Authoring rules

What this unit is about

This unit teaches you how to author rule artifacts that implement the business logic and policies of a business rule application.

What you should be able to do

After completing this unit, you should be able to:

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

How you will check your progress

- Checkpoint
- Exercises



Unit objectives

After completing this unit, you should be able to:

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

© Copyright IBM Corporation 2016

Figure 6-1. Unit objectives

WB3951.2

Notes:



Topics

- From business policy to business rules
- Authoring action rules with BAL
- Authoring decision tables and trees
- Working with templates
- Introducing ILOG Rule Language
- Defining objects that are used in rules

© Copyright IBM Corporation 2016

Figure 6-2. Topics

WB3951.2

Notes:

6.1. From business policy to business rules

From business policy to business rules



© Copyright IBM Corporation 2016

Course materials may not be reproduced in whole or in part without the prior written permission of IBM.

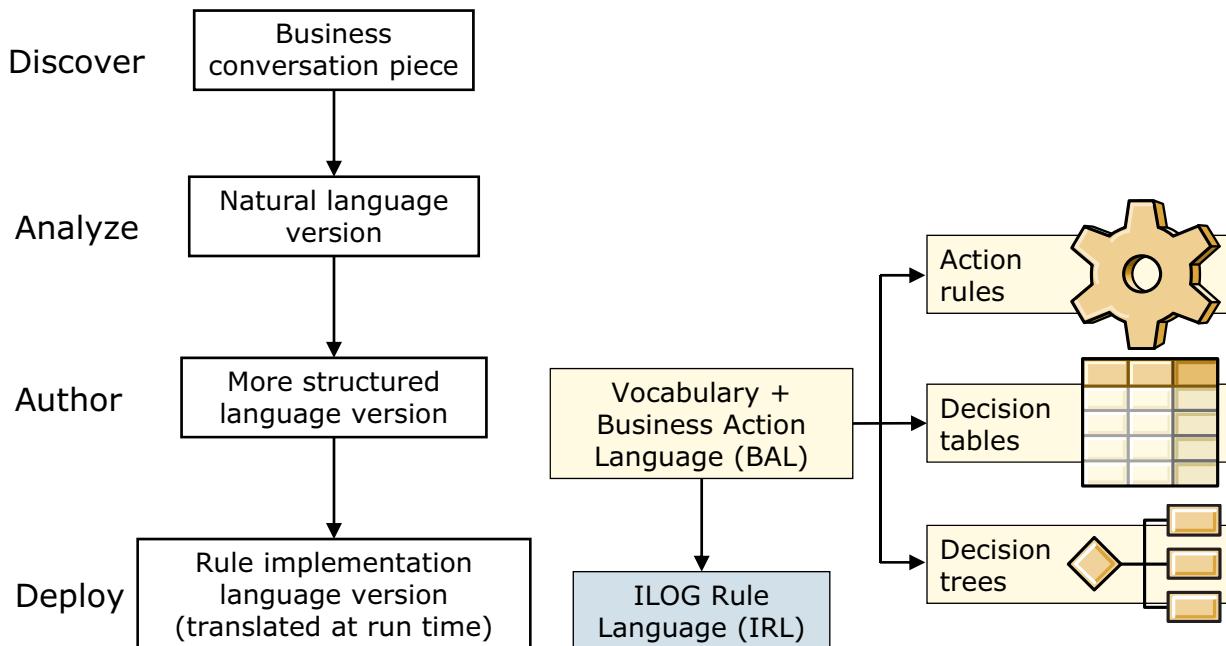
10.1

Figure 6-3. From business policy to business rules

WB3951.2

Notes:

Rule language evolves during a project (1 of 2)



© Copyright IBM Corporation 2016

Figure 6-4. Rule language evolves during a project (1 of 2)

WB3951.2

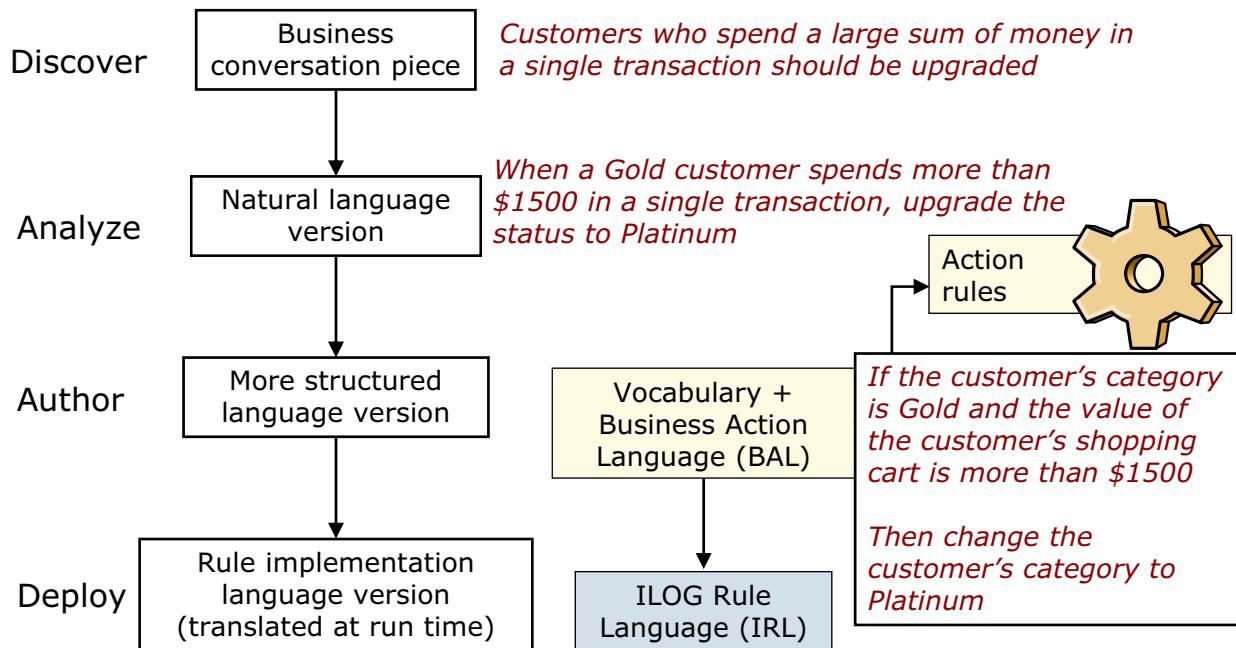
Notes:

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, decision trees, and technical rules.
- Rules are written in Business Action Language (BAL), which is a rule language that is used in Operational Decision Manager. BAL is a quasi-natural language that combines rule constructs with terms and phrases that are expressed in the vocabulary. As you learned earlier, the vocabulary is created through verbalization of the business object model in Rule Designer.

- *Deploy:* When rules are deployed, the rule statement, which is written in BAL from the BOM vocabulary, is translated into the rule implementation language, which is what the rule engine sees at run time. This language is called the ILOG rule language, or IRL.

Rule language evolves during a project (2 of 2)



© Copyright IBM Corporation 2016

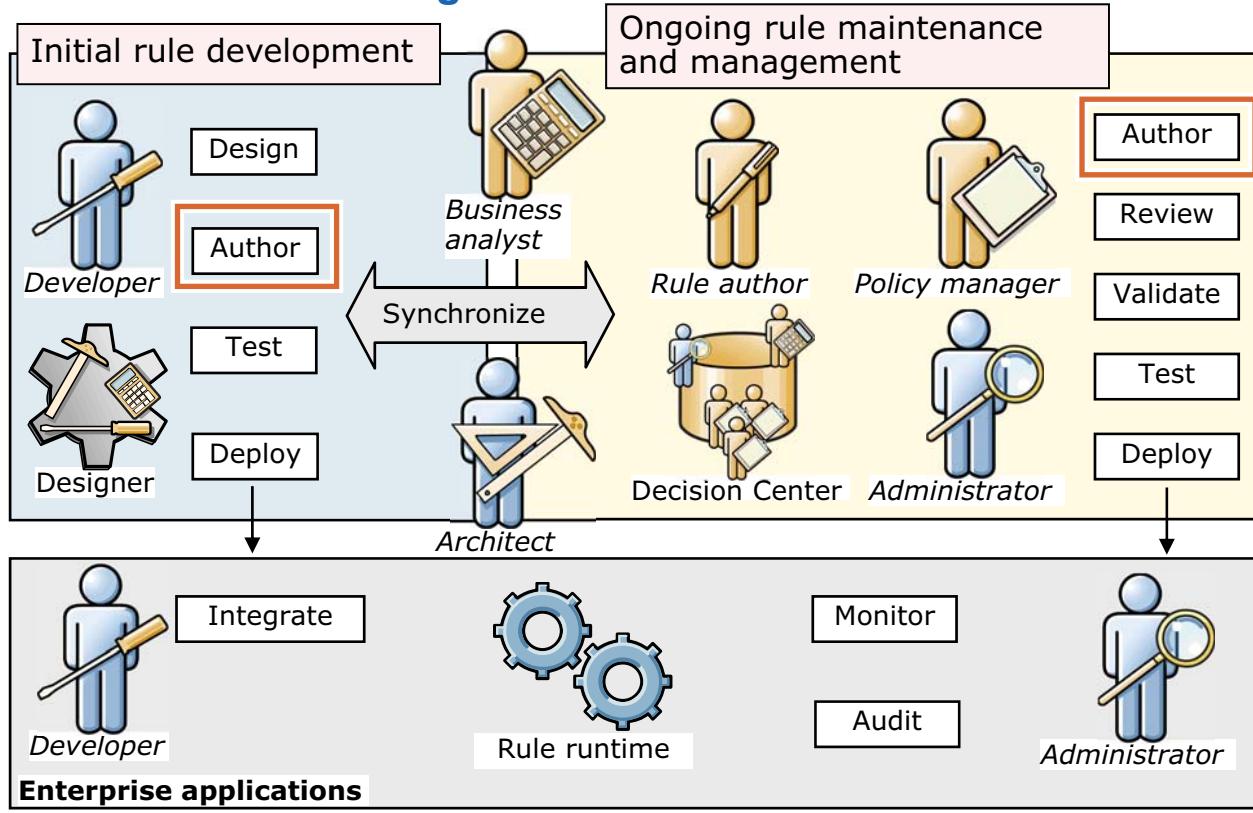
Figure 6-5. Rule language evolves during a project (2 of 2)

WB3951.2

Notes:



When rule authoring occurs



© Copyright IBM Corporation 2016

Figure 6-6. When rule authoring occurs

WB3951.2

Notes:

- Initial rule development:

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. The rule project is designed with a folder structure to organize the rules in rule packages. As soon as the vocabulary is implemented as a Business Object Model (BOM), rule authoring can begin.

- Ongoing rule maintenance:

Management and maintenance continue after the business rule application is rolled out to production. Rule maintenance becomes a regular task for some business users when business policies shift and more rules are identified.



Who writes rules?

- Business analysts
 - Typically involved heavily during initial rule development
 - Might author many rules
- Policy managers and rule authors
 - Act as subject matter experts (SMEs) during early stages
 - Involved in reviewing and validating rules
 - Responsible for maintaining rules (write and edit) after the system is rolled out
- Developers
 - Involved heavily during initial rule development
 - Create technical rules and complex rules

© Copyright IBM Corporation 2016

Figure 6-7. Who writes rules?

WB3951.2

Notes:



Where are rules written?

- Decision Center Business console and Enterprise console
 - Business console is main rule authoring and management environment for business users
 - Includes features for rule administrators
- Rule Solutions for Office
 - Provides for offline rule authoring and review in Microsoft Office Word and Excel for business users
- Rule Designer
 - Eclipse-based development environment for developers and technical business analysts
 - Includes tools to manage synchronization with Decision Center

© Copyright IBM Corporation 2016

Figure 6-8. Where are rules written?

WB3951.2

Notes:

6.2. Authoring action rules with BAL

Authoring action rules with BAL



© Copyright IBM Corporation 2016

Course materials may not be reproduced in whole or in part without the prior written permission of IBM.

10.1

Figure 6-9. Authoring action rules with BAL

WB3951.2

Notes:

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
 - In Rule Solutions for Office

© Copyright IBM Corporation 2016

Figure 6-10. Business Action Language (BAL)

WB3951.2

Notes:

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, Decision Center consoles, and Rule Solutions for Office.

Rule structure

Definitions	<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 <code>else</code> define the actions when conditions are not metKeyword <code>else</code> is optional, and valid only when the keyword <code>if</code> is present

© Copyright IBM Corporation 2016

Figure 6-11. Rule structure

WB3951.2

Notes:

Example of action rule

definitions

```
set 'loyal customer' to a customer
    where the category of this customer is Gold;
```

1

if

```
the age of 'loyal customer' is more than 60
```

2

then ← a

```
give a 10% discount to 'loyal customer';
```

3

else ← b

```
give a 5% discount to 'loyal customer';
```

© Copyright IBM Corporation 2016

Figure 6-12. Example of action rule

WB3951.2

Notes:

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>`

© Copyright IBM Corporation 2016

Figure 6-13. Rule definitions: Overview

WB3951.2

Notes:

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 };
```

© Copyright IBM Corporation 2016

Figure 6-14. Rule definitions: Values

WB3951.2

Notes:

You can assign the following kinds of values to your rule variable:

- A constant
- An expression
- An object
- A collection of objects

Rule definitions: Constraints

- set 'customer' to a customer from the customer of 'agreement'
where the category of this customer is one of 'categories';
- set 'offer' to an offer in the offers of agreement
where the discount of this offer is at least 'c1';
- set 'offerlist' to all offers in the offers of agreement
where the discount of each offer is less than 'c1';

© Copyright IBM Corporation 2016

Figure 6-15. Rule definitions: Constraints

WB3951.2

Notes:

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

© Copyright IBM Corporation 2016

Figure 6-16. Multiple variables in BAL

WB3951.2

Notes:



Warning

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;

```

© Copyright IBM Corporation 2016

Figure 6-17. Rule conditions: Introduction

WB3951.2

Notes:

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'

© Copyright IBM Corporation 2016

Figure 6-18. Rule conditions: Comparison test

WB3951.2

Notes:



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'

© Copyright IBM Corporation 2016

Figure 6-19. Rule conditions: Membership test

WB3951.2

Notes:

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'

© Copyright IBM Corporation 2016

Figure 6-20. Rule conditions: Existence test

WB3951.2

Notes:

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

© Copyright IBM Corporation 2016

Figure 6-21. Rule conditions: Count test

WB3951.2

Notes:



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

© Copyright IBM Corporation 2016

Figure 6-22. Multiple conditions

WB3951.2

Notes:

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)

© Copyright IBM Corporation 2016

Figure 6-23. Avoiding ambiguity with parentheses (1 of 2)

WB3951.2

Notes:



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

© Copyright IBM Corporation 2016

Figure 6-24. Avoiding ambiguity with parentheses (2 of 2)

WB3951.2

Notes:

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

© Copyright IBM Corporation 2016

Figure 6-25. Use of commas (,) in rule conditions

WB3951.2

Notes:

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

© Copyright IBM Corporation 2016

Figure 6-26. Rule actions

WB3951.2

Notes:

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 15% 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;

© Copyright IBM Corporation 2016

Figure 6-27. Examples of rule actions

WB3951.2

Notes:

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>] ,]

© Copyright IBM Corporation 2016

Figure 6-28. BAL number operators

WB3951.2

Notes:

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'

© Copyright IBM Corporation 2016

Figure 6-29. BAL arithmetic operators

WB3951.2

Notes:

6.3. Authoring decision tables and trees

Authoring decision tables and trees



© Copyright IBM Corporation 2016

Course materials may not be reproduced in whole or in part without the prior written permission of IBM.

10.1

Figure 6-30. Authoring decision tables and trees

WB3951.2

Notes:

Decision tables and trees

- Concise way of viewing a set of action rules in the form of a spreadsheet or tree structure
- **Decision tables:** Suitable for representing sets of action rules that have a uniform structure
- **Decision trees:** Suitable for large sets of action rules that are not symmetric
 - A different set of actions for each branch
 - A different set of conditions for each branch
- Advantages of decision tables and decision trees
 - Preconditions that apply to an entire decision table or tree
 - Built-in error checking that verifies the structure of your data for gaps and overlaps

© Copyright IBM Corporation 2016

Figure 6-31. Decision tables and trees

WB3951.2

Notes:

Decision tables and decision trees provide a concise way of viewing a set of business rules in the form of a spreadsheet or tree structure.

Decision tables are suitable for representing sets of business rules that have a uniform structure.

Decision trees are suitable for representing sets of business rules that have a different set of actions for each branch or a different set of conditions for each branch.

This unit focuses primarily on decision tables because they are more commonly used, but it also briefly introduces decision trees and describes when they can be useful.



Preconditions

- A precondition can be added to a decision table or tree
 - To limit the scope of the rules in decision trees or tables
 - To define variables that can be used in all the rules of the decision table or tree
- The precondition of a decision table or tree applies to all the rules that this decision table or tree defines

© Copyright IBM Corporation 2016

Figure 6-32. Preconditions

WB3951.2

Notes:



Error checking in the editors

- Decision table editor and decision tree 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
- Decision trees verify:
 - Overlapping data with hierarchical discrimination
 - Contiguity of test links

© Copyright IBM Corporation 2016

Figure 6-33. Error checking in the editors

WB3951.2

Notes:



About decision tables

- Convenient way to view and manage large sets of similar business rules
- Composed of rows and columns:
 - Each row corresponds to a single rule
 - Columns define the conditions and actions
- Other benefits:
 - Can apply preconditions to all rules
 - Easy to see overlaps or gaps
 - Consistency checking features (static rule analysis)
- When to use decision tables:
 - Common condition tests
 - Similar actions

© Copyright IBM Corporation 2016

Figure 6-34. About decision tables

WB3951.2

Notes:

Decision tables are generally used when rules share common condition terms, and when they have similar actions.

Example: Symmetrical rules

- Patterns are a good indicator that a decision table might be useful
 - If ***the miles per year of the vehicle*** is less than 4,000, then **set the third-party price before tax of the policy** to the third-party price before tax of the policy * 0.6
 - If ***the miles per year of the vehicle*** is at least 4,000 and less than 8,000, then **set the third-party price before tax of the policy** to the third-party price before tax of the policy * 0.8
 - If ***the miles per year of the vehicle*** is at least 8,000 and less than 15,000, then **set the third-party price before tax of the policy** to the third-party price before tax of the policy * 1
 - If ***the miles per year of the vehicle*** is at least 15,000 and less than 25,000, then **set the third-party price before tax of the policy** to the third-party price before tax of the policy * 1.2
 - If ***the miles per year of the vehicle*** is at most 25,000, then **set the third-party price before tax of the policy** to the third-party price before tax of the policy * 1.3
- Each of these rules can become a row in a decision table

© Copyright IBM Corporation 2016

Figure 6-35. Example: Symmetrical rules

WB3951.2

Notes:

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

- Each row represents a rule
 - When conditions for a row are met, actions in that row are executed

Rule

if
all of the following conditions are true :
- (the miles per year of 'the vehicle' is less than 4000),

then
set the third party price before tax of 'the policy' to the third party price before tax of 'the policy' * 0.6 ;

Vehicle miles per year	MPY FActor
1 < 4,000	0.6

© Copyright IBM Corporation 2016

Figure 6-36. Example: Symmetrical rules in a decision table

WB3951.2

Notes:

Structure of a decision table

The diagram illustrates the structure of a decision table. It features a grid with five rows, each representing a rule. The first column contains rule numbers (1 through 5). The second column contains conditions based on vehicle miles per year, and the third column contains corresponding MPY FFactors. The columns are labeled at the top: 'Vehicle miles per year' and 'MPY FFactor'. Brackets on the left group the rows as 'Rows = Rules', and brackets above the columns group them as 'Column header', 'Row header', 'Condition column', and 'Action column'.

	Vehicle miles per year	MPY FFactor
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

© Copyright IBM Corporation 2016

Figure 6-37. Structure of a decision table

WB3951.2

Notes:

Columns for conditions and actions

- Condition columns are on the left, unshaded
- Action columns are on the right, shaded

	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

© Copyright IBM Corporation 2016

Figure 6-38. Columns for conditions and actions

WB3951.2

Notes:

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



Locking facilities

- Decision tables have locking facilities to protect your decision tables against editing by others
- Locking facilities apply to both the condition columns and action columns
- You can lock the structure of the decision table
- You can also lock any predefined data

© Copyright IBM Corporation 2016

Figure 6-39. Locking facilities

WB3951.2

Notes:

Decision table size

- Try not to create decision tables too large in terms of the number of rows and number of columns
 - A few hundred rows and 10 – 20 columns are acceptable values to keep the tools efficient
- If a decision table becomes too large, the time it takes to save and compile the data becomes too long
 - Compilation is when the IRL of the table is created

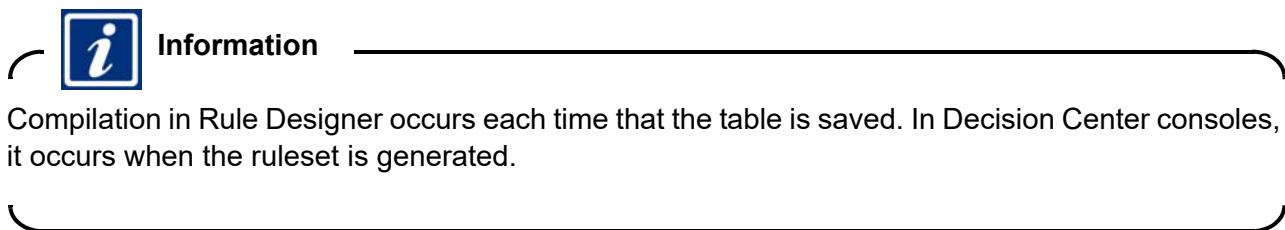
© Copyright IBM Corporation 2016

Figure 6-40. Decision table size

WB3951.2

Notes:

Try not to create decision tables with too many rows and columns. A few hundred rows and 10–20 columns are acceptable values to keep the decision tables efficient. If a decision table becomes too large, then saving and compiling the data can take longer. Compilation is when the IRL of the table is created.



About decision trees

- Provide same function as decision tables
- More adapted to a set of action rules that are not sufficiently symmetric to present in a decision table
- Make it easy to understand large sets of nonsymmetrical rules
 - The path from the first condition to the end of the actions along any branch corresponds to one rule
- Other benefits:
 - Can apply preconditions to all rules
 - Easy to see overlaps or gaps

© Copyright IBM Corporation 2016

Figure 6-41. About decision trees

WB3951.2

Notes:

Decision tree (pseudocode)

```
if the age of the client is less than 18 then
    the loan is refused because the client is underaged
else {
    if condition A then
        the loan is accepted
    else if condition B then
        an insurance is required
    else if condition C then {
        if the client spouse had a recent bankruptcy then
            refuse the loan
        else
            accept the loan
    }
}
```

© Copyright IBM Corporation 2016

Figure 6-42. Decision tree (pseudocode)

WB3951.2

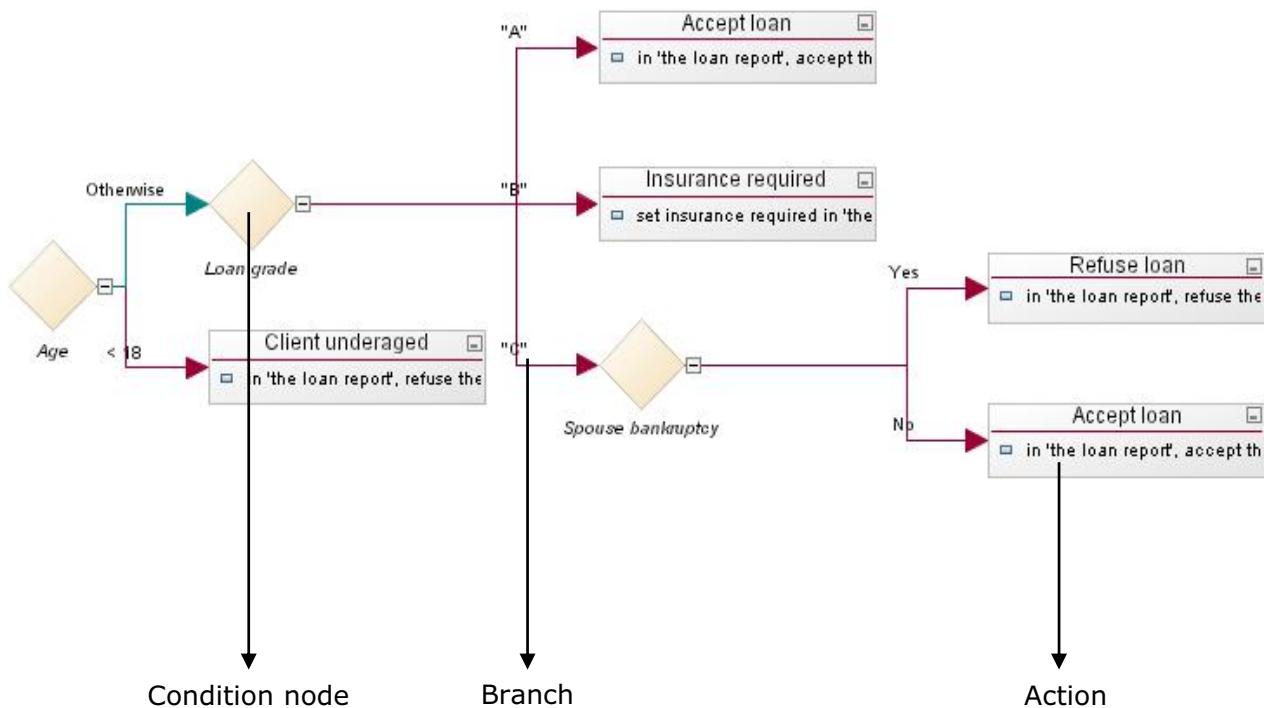
Notes:

Consider the example of the textual business policy that is written as pseudocode on this slide.

You see that the first branch is shorter than the other branches. It has no condition A, B or C, or test about the client spouse bankruptcy.

These rules can be implemented in a decision tree like the one that is presented in the next graphic.

Corresponding decision tree



© Copyright IBM Corporation 2016

Figure 6-43. Corresponding decision tree

WB3951.2

Notes:

These rules can be implemented in a decision tree like the one shown on this slide, where:

- A condition is declared in its diamond-shaped node
- Branches represent the possible values for the conditions
- The actions are declared at the ends of the branches

6.4. Working with templates

Working with templates



© Copyright IBM Corporation 2016

Course materials may not be reproduced in whole or in part without the prior written permission of IBM.

10.1

Figure 6-44. Working with templates

WB3951.2

Notes:



Creating templates for business users

- You can simplify rule authoring tasks for business users by creating templates:
 - Action rule templates
 - Decision table templates
- Templates are partially completed rules that can be used as a starting point for building multiple rules or decision tables with a similar structure
- Use the Guided editor in Rule Designer to create or edit templates
- Use “freeze” and “unfreeze” tools to define which parts of the template can be edited
 - If you change a freeze or unfreeze setting in an action rule template, you can propagate your changes in the action rules that are created from this template
 - You cannot automatically propagate changes that are done in the action rule template other than the freeze or unfreeze settings

© Copyright IBM Corporation 2016

Figure 6-45. Creating templates for business users

WB3951.2

Notes:

You might have cases when you have a number of action rules with similar content and structure. To help you author them efficiently, create an action rule template for this series of rules. An action rule template is a partially written action rule that is used to create multiple action rules with a similar structure.

You can freeze parts of the action rule template to protect against editing in the action rules that are created from it. Frozen parts in the action rule template limit which parts of the rule can be edited.

Based on this action rule template, you and business users find it easier to author rules. The structure is ready, and you modify only the non-frozen parts of the rule. It is typically your role as a developer to create these action rule templates in Rule Designer to facilitate rule authoring for business users in Decision Center Console.

In Rule Designer, you build action rule templates in the same way that you create a regular action rule. However, unlike a regular action rule, you leave parts of the rule empty to allow for variations in the action rules that are created from it.

You can use only the Guided Editor to build rule templates and the action rules that are created from the template because frozen parts cannot be enforced in the Intellirule editor.

You can then create a series of action rules that share patterns from that action rule template by using the Guided Editor.

6.5. Introducing ILOG Rule Language

Introducing ILOG Rule Language



© Copyright IBM Corporation 2016

Course materials may not be reproduced in whole or in part without the prior written permission of IBM.

10.1

Figure 6-46. Introducing ILOG Rule Language

WB3951.2

Notes:

The business rules are written in BAL. Before the rule engine can process the rules, they must first be translated into IRL, which is the technical language that the engine understands.

Business users do not have to understand this translation when they write business rules in BAL. The business rules, which use the verbalized terms and phrases from the BOM, are automatically translated to technical rule constructs at design time, as you can see on the **IRL** tab of the rule editor. For example, each row of a decision table is generated as a rule in IRL. Each action column corresponds to an IRL action in the corresponding rule.

Technical rules and functions are written directly against the BOM elements, and do not use verbalized terms or phrases.

When the ruleset is built, all rule artifacts, either business or technical, are translated as IRL-based artifacts. At run time, the BOM elements are translated into their XOM counterparts for execution through the BOM-to-XOM mapping, as you learned in Unit 4, "Developing object models".



ILOG Rule Language (IRL)

- Executable form of the rule artifacts
- Rule language similar to Java and executable directly by the rule engine

© Copyright IBM Corporation 2016

Figure 6-47. ILOG Rule Language (IRL)

WB3951.2

Notes:

The ILOG Rule Language (IRL) is the executable form of the rule artifacts. The rule engine can process and execute IRL directly.



Translation of business rule artifacts

- In the ruleset, business rule artifacts that are written in BAL are translated to the IRL
- The BAL code is automatically translated to the corresponding IRL for you
- You can view the IRL that corresponds to a BAL rule, in the IRL tab in the rule editor

© Copyright IBM Corporation 2016

Figure 6-48. Translation of business rule artifacts

WB3951.2

Notes:



Language for technical rule artifacts

- Use IRL to author:
 - Technical rules
 - Function
 - Action tasks in the ruleflow
 - BOM-to-XOM mapping
- You can use technical rules to express more powerful rules

© Copyright IBM Corporation 2016

Figure 6-49. Language for technical rule artifacts

WB3951.2

Notes:

IRL syntax

- A technical rule is made of *conditions* and *actions*
- Conditions begin with the keyword `when` (`if` in BAL)
- Actions begin with the keyword `then`
- Variables: `?x`
 - The `x` variable at the beginning of the condition identifies any object that matches this condition that you can further use in other conditions or in the actions of the rule
- To import required elements, use the keywords `use` and `import`

© Copyright IBM Corporation 2016

Figure 6-50. IRL syntax

WB3951.2

Notes:

- 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.



Multiple variables in IRL

- Unlike BAL, if you define multiple rule variables in IRL to denote instances of the same class, they might denote the same instance when the rule executes, unless you specify further constraints
- To make sure that these instances are different, you must explicitly add an IRL test

© Copyright IBM Corporation 2016

Figure 6-51. Multiple variables in IRL

WB3951.2

Notes:

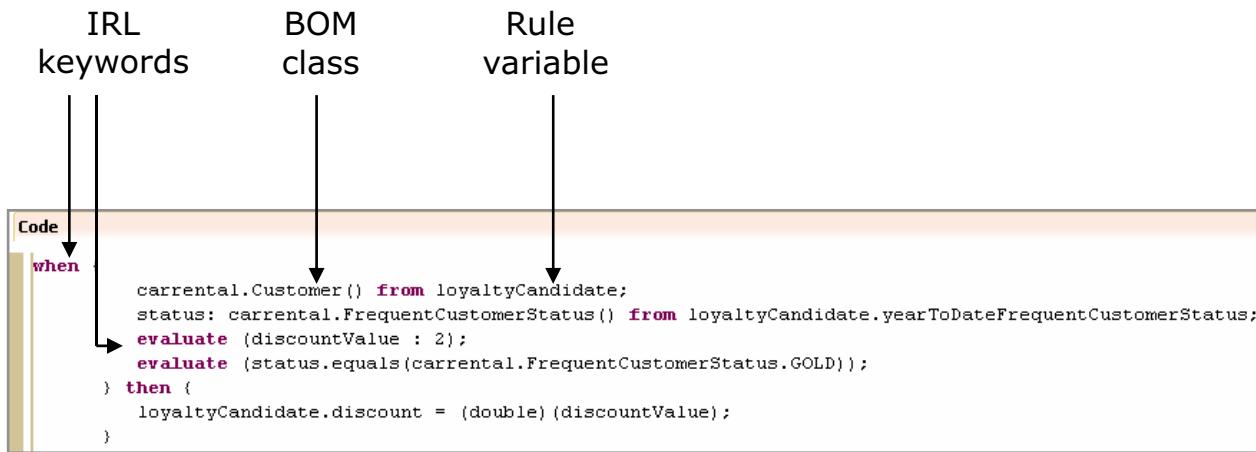
In IRL, you can also define multiple rule variables to denote instances of the same class. However, in IRL, they can denote the same instance if you do not specify further constraints. For example, the following IRL rule is similar in aspect to the BAL rule indicated previously. It defines two variables `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.

```
when {
  customer1: carrental.Customer();
  customer2: carrental.Customer();
} then {
  ...
}
```

To make sure that these instances are different, add an IRL test. For example, you can add it in the definition of `customer2`:

```
when {  
    customer1: carrental.Customer();  
    customer2: carrental.Customer(?this != customer1);  
} then {  
    ...  
}
```

Technical rules



© Copyright IBM Corporation 2016

Figure 6-52. Technical rules

WB3951.2

Notes:

6.6. Defining objects that are used in rules

Defining objects that are used in rules



© Copyright IBM Corporation 2016

Course materials may not be reproduced in whole or in part without the prior written permission of IBM.

10.1

Figure 6-53. Defining objects that are used in rules

WB3951.2

Notes:

Introduction

- Business objects are made available to the rule artifacts and the rule engine by using:
 - Ruleset 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

© Copyright IBM Corporation 2016

Figure 6-54. Introduction

WB3951.2

Notes:

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 *ruleset parameters*, as *ruleset variables* in your decision service, or as *objects in the working memory*.



Important

The objects that you model as ruleset parameters or ruleset variables might also play a role in orchestrating the execution of your rule artifacts.

Ruleset 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

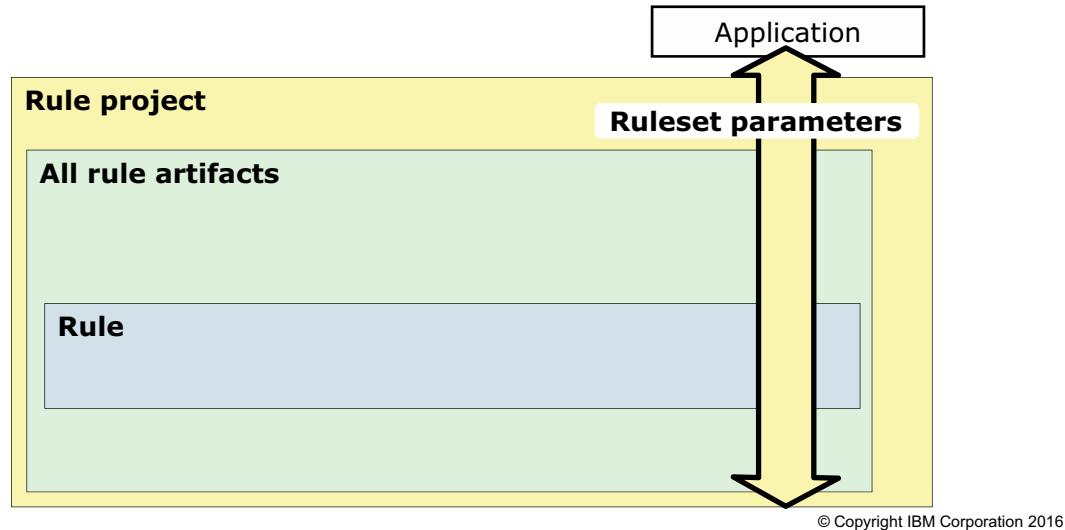


Figure 6-55. Ruleset parameters

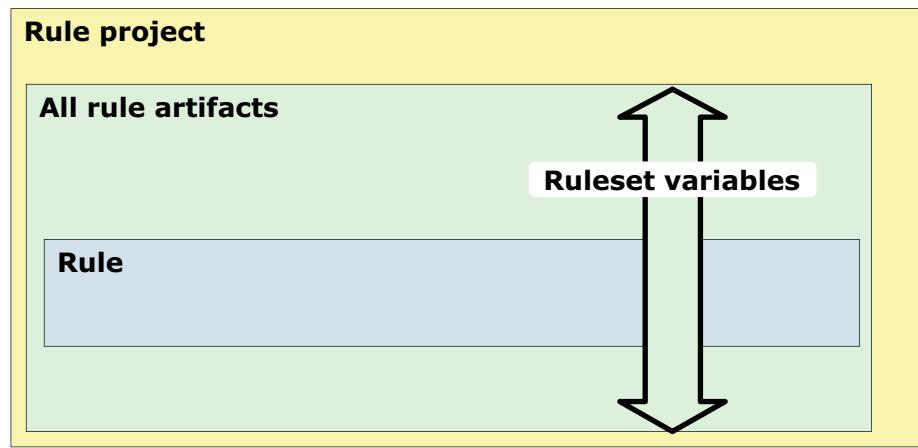
WB3951.2

Notes:

The ruleset parameters form the signature of a ruleset. Their values that are passed to or from the client application, or in both directions, are the actual objects on which your rule artifacts work.

Ruleset variables

- Pass data between rules and ruleflow tasks
- Grouped into variable sets that are stored in rule packages or at the project root
- Scope: Available to all rule artifacts in the ruleset



© Copyright IBM Corporation 2016

Figure 6-56. Ruleset variables

WB3951.2

Notes:

Ruleset variables define objects internal to the ruleset. Like the ruleset 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 ruleset 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)

© Copyright IBM Corporation 2016

Figure 6-57. Objects in working memory

WB3951.2

Notes:

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
Ruleset 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

© Copyright IBM Corporation 2016

Figure 6-58. Working with objects in rule artifacts

WB3951.2

Notes:



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 ruleset parameter that is called `report` can be manipulated in IRL as followed: `evaluate (report.approved);`
- Because objects in the working memory have no name, they cannot be manipulated that way

© Copyright IBM Corporation 2016

Figure 6-59. Using IRL

WB3951.2

Notes:

Ruleset 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 ruleset 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

© Copyright IBM Corporation 2016

Figure 6-60. Using verbalized elements in BAL

WB3951.2

Notes:

When your ruleset 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:

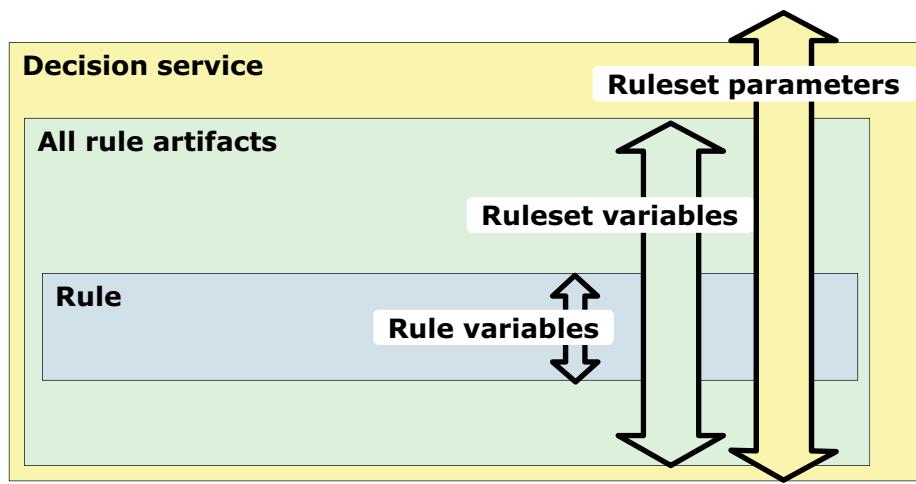
`if 'the loan report' is approved;`

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 ruleset 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



© Copyright IBM Corporation 2016

Figure 6-61. Rule variables (1 of 2)

WB3951.2

Notes:

Rule variables (2 of 2)

- Because objects in the working memory cannot be named or verbalized, you must bind them to rule variables or automatic variables by defining a pattern in the definitions of the rule artifact to manipulate them
- You can also use rule variables to enhance the way that your rules are written

© Copyright IBM Corporation 2016

Figure 6-62. Rule variables (2 of 2)

WB3951.2

Notes:



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

© Copyright IBM Corporation 2016

Figure 6-63. Using automatic variables

WB3951.2

Notes:



Unit summary

Having completed this unit, you should be able to:

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

© Copyright IBM Corporation 2016

Figure 6-64. Unit summary

WB3951.2

Notes:



Checkpoint questions

1. Apart from Rule Designer, which of the following Operational Decision Manager modules can you use to create and maintain rule artifacts?
 - a. Decision Center Business console
 - b. Decision Center Enterprise console
 - c. Rule Execution Server
 - d. Both a and b
 - e. Both a and c
 - f. All of the above

2. Apart from Business Action Language (BAL), which other language can you use to write rule artifacts?
 - a. Technical Rule Language (TRL)
 - b. BOM-to-XOM mapping (B2X)
 - c. ILOG Rule Language (IRL)
 - d. Guided Rule Language (GRL)
 - e. None of the above

© Copyright IBM Corporation 2016

Figure 6-65. Checkpoint questions

WB3951.2

Notes:

Write your answers here:

- 1.

- 2.



Checkpoint answers

1. Apart from Rule Designer, which of the following Operational Decision Manager modules can you use to create and maintain rule artifacts?
 - d. Both a and b. Decision Center Business console and Enterprise console

2. Apart from Business Action Language (BAL), which other language can you use to write rule artifacts?
 - c. ILOG Rule Language (IRL)

© Copyright IBM Corporation 2016

Figure 6-66. Checkpoint answers

WB3951.2

Notes:

Exercise 6

Exploring action rules

© Copyright IBM Corporation 2016

Course materials may not be reproduced in whole or in part without the prior written permission of IBM.

10.1

Figure 6-67. Exercise 6

WB3951.2

Notes:



Exercise objectives

After completing this exercise, you should be able to:

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

© Copyright IBM Corporation 2016

Figure 6-68. Exercise objectives

WB3951.2

Notes:



Exercise 7



© Copyright IBM Corporation 2016

Course materials may not be reproduced in whole or in part without the prior written permission of IBM.

10.1

Figure 6-69. Exercise 7

WB3951.2

Notes:

Exercise objectives

After completing this exercise, you should be able to:

- Use the Intellirule editor and Guided editor to author action rules
- Use rule variables, automatic variables, and ruleset parameters in rule statements

© Copyright IBM Corporation 2016

Figure 6-70. Exercise objectives

WB3951.2

Notes:

Exercise 8



Authoring decision tables and decision trees

© Copyright IBM Corporation 2016

Course materials may not be reproduced in whole or in part without the prior written permission of IBM.

10.1

Figure 6-71. Exercise 8

WB3951.2

Notes:



Exercise objectives

After completing this exercise, you should be able to:

- Use the decision table editor to create a decision table
- Use the decision tree editor to create a decision tree

© Copyright IBM Corporation 2016

Figure 6-72. Exercise objectives

WB3951.2

Notes:

Unit 7. Customizing rule vocabulary with categories and domains

What this unit is about

This unit teaches you how to work with categories and domains to customize rule vocabulary.

What you should be able to do

After completing this unit, you should be able to:

- Simplify rule authoring by using categories
- Define domains

How you will check your progress

- Checkpoint
- Exercises



Unit objectives

After completing this unit, you should be able to:

- Simplify rule authoring by using categories
- Define domains

© Copyright IBM Corporation 2016

Figure 7-1. Unit objectives

WB3951.2

Notes:



Topics

- Simplifying vocabulary with categories
- Defining domains
- Static domains
- Dynamic domains

© Copyright IBM Corporation 2016

Figure 7-2. Topics

WB3951.2

Notes:

7.1. Simplifying vocabulary with categories

Simplifying vocabulary with categories



© Copyright IBM Corporation 2016

Course materials may not be reproduced in whole or in part without the prior written permission of IBM.

10.1

Figure 7-3. Simplifying vocabulary with categories

WB3951.2

Notes:



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

A screenshot of a user interface titled 'Categories'. It contains the following text:

Define the categories associated with this member.
[Edit the categories.](#)

Any

© Copyright IBM Corporation 2016

Figure 7-4. Categories in the BOM editor

WB3951.2

Notes:

A *category* is an identifier that you apply to BOM classes and members to filter the BOM elements available in your business rules.

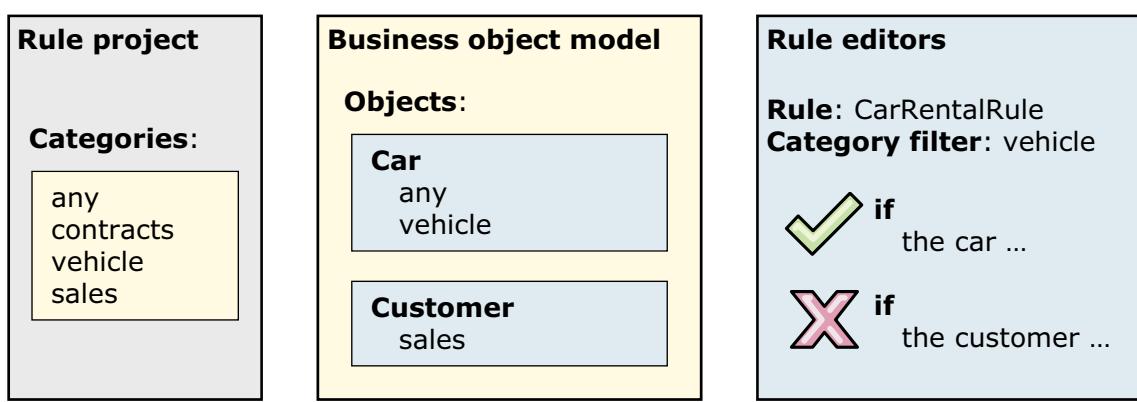
You tag BOM elements with categories to specify whether these BOM elements can be used in specific rules.

- If a category is “hidden” for a specific rule, business elements that are tagged with that category are not visible 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 of the rule editor.

Use categories to simplify the list of vocabulary that is available in the rule editors. Categories make rule authoring easier for business users and they 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:



© Copyright IBM Corporation 2016

Figure 7-5. Categories

WB3951.2

Notes:

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

© Copyright IBM Corporation 2016

Figure 7-6. Category semantics

WB3951.2

Notes:

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

7.2. Defining domains

Defining domains



© Copyright IBM Corporation 2016

Course materials may not be reproduced in whole or in part without the prior written permission of IBM.

10.1

Figure 7-7. Defining domains

WB3951.2

Notes:



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

▼ Domain

Create and edit a domain for this member.

[Create a domain.](#)

© Copyright IBM Corporation 2016

Figure 7-8. Domains section

WB3951.2

Notes:

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 you specify.
- When you analyze rules, domain types in the BOM are used to check the consistency of action rule semantics.

7.3. Static domains

Static domains



© Copyright IBM Corporation 2016

Course materials may not be reproduced in whole or in part without the prior written permission of IBM.

10.1

Figure 7-9. Static domains

WB3951.2

Notes:



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

© Copyright IBM Corporation 2016

Figure 7-10. Domains: Literals

WB3951.2

Notes:



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`

© Copyright IBM Corporation 2016

Figure 7-11. Domains: Static references

WB3951.2

Notes:

Domains: Bounded

- A bounded domain is defined as an interval between two bounding values (included, or not)
- Example:
 - [0, 120]: The possible value is an integer value greater than or equal to 0, and less than or equal to 120
- If the domain has a missing bound, specify that missing bound with an asterisk (*)
 - [0, *]: The possible value is an integer value greater than or equal to 0
 - [* , 0]: The possible value is an integer value less than or equal to 0
- Examples:
 - [0, *]: The possible value is an integer value greater than or equal to 0
 - [* , 0]: The possible value is an integer value less than or equal to 0

© Copyright IBM Corporation 2016

Figure 7-12. Domains: Bounded

WB3951.2

Notes:

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

© Copyright IBM Corporation 2016

Figure 7-13. Domains: Collection

WB3951.2

Notes:



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"

© Copyright IBM Corporation 2016

Figure 7-14. Domains: Other

WB3951.2

Notes:

7.4. Dynamic domains

Dynamic domains



© Copyright IBM Corporation 2016

Course materials may not be reproduced in whole or in part without the prior written permission of IBM.

10.1

Figure 7-15. Dynamic domains

WB3951.2

Notes:



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

© Copyright IBM Corporation 2016

Figure 7-16. Domains: Dynamic (1 of 2)

WB3951.2

Notes:

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.

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

© Copyright IBM Corporation 2016

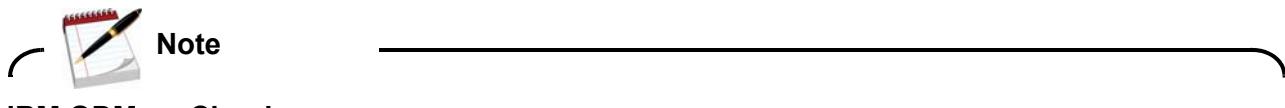
Figure 7-17. Domains: Dynamic (2 of 2)

WB3951.2

Notes:

As their name indicates, domains of type `Dynamic` are “dynamic” rather than “static.”

Similar to other types of domains, you create a domain of type `Dynamic` in the BOM editor.



IBM ODM on Cloud

ODM on Cloud supports dynamic domains with Microsoft Excel. It does not support custom data providers for dynamic domains.



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

© Copyright IBM Corporation 2016

Figure 7-18. Dynamic domains: Microsoft Excel source (1 of 4)

WB3951.2

Notes:

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 (2 of 4)

- The supported Excel formats are:
 - Microsoft Excel 2003
 - Microsoft Excel 2007
- 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

© Copyright IBM Corporation 2016

Figure 7-19. Dynamic domains: Microsoft Excel source (2 of 4)

WB3951.2

Notes:

As you learned in Unit 3, "Programming with business rules", 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 (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

© Copyright IBM Corporation 2016

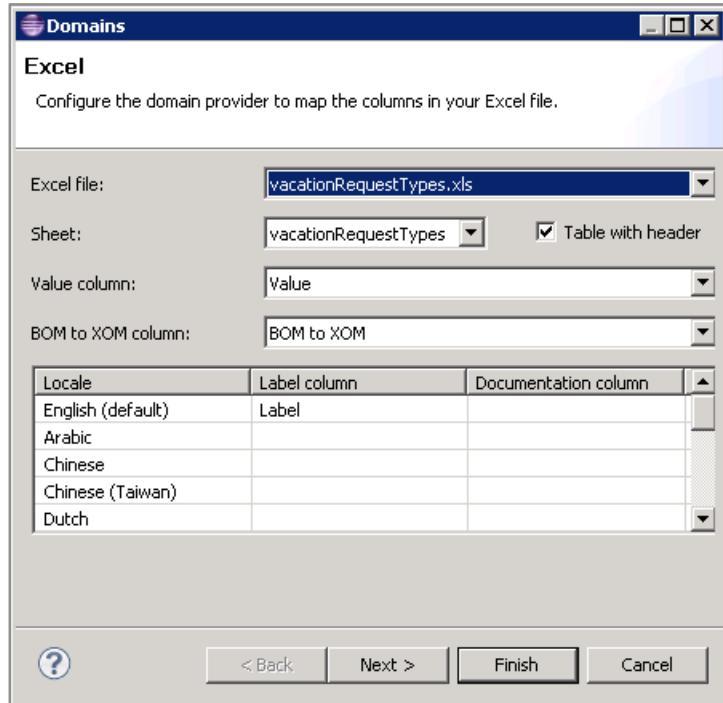
Figure 7-20. Dynamic domains: Microsoft Excel source (3 of 4)

WB3951.2

Notes:

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



© Copyright IBM Corporation 2016

Figure 7-21. Dynamic domains: Microsoft Excel source (4 of 4)

WB3951.2

Notes:

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.



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

© Copyright IBM Corporation 2016

Figure 7-22. Enumerated versus complex domains

WB3951.2

Notes:

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.



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

© Copyright IBM Corporation 2016

Figure 7-23. Automatic creation of domains

WB3951.2

Notes:

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`.



Unit summary

Having completed this unit, you should be able to:

- Simplify rule authoring by using categories
- Define domains

© Copyright IBM Corporation 2016

Figure 7-24. Unit summary

WB3951.2

Notes:



Checkpoint 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 rule project properties.

© Copyright IBM Corporation 2016

Figure 7-25. Checkpoint questions

WB3951.2

Notes:

Write your answers here:

1.

2.



Checkpoint answers

1. **True.**
2. **False:** Domains are defined in the BOM editor.

© Copyright IBM Corporation 2016

Figure 7-26. Checkpoint answers

WB3951.2

Notes:

Exercise 9



© Copyright IBM Corporation 2016

Course materials may not be reproduced in whole or in part without the prior written permission of IBM.

10.1

Figure 7-27. Exercise 9

WB3951.2

Notes:



Exercise objectives

After completing this exercise, you should be able to:

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

© Copyright IBM Corporation 2016

Figure 7-28. Exercise objectives

WB3951.2

Notes:

Exercise 10



© Copyright IBM Corporation 2016

Course materials may not be reproduced in whole or in part without the prior written permission of IBM.

10.1

Figure 7-29. Exercise 10

WB3951.2

Notes:



Exercise objectives

After completing this exercise, you should be able to:

- 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 2016

Figure 7-30. Exercise objectives

WB3951.2

Notes:

Unit 8. Working with queries

What this unit is about

This unit explains how to use search and query tools with rule artifacts.

What you should be able to do

After completing this unit, you should be able to:

- 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

How you will check your progress

- Checkpoint
- Exercise



Unit objectives

After completing this unit, you should be able to:

- 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 2016

Figure 8-1. Unit objectives

WB3951.2

Notes:



Topics

- Searching for rule artifacts
- Querying rules
- Extracting rulesets

© Copyright IBM Corporation 2016

Figure 8-2. Topics

WB3951.2

Notes:

8.1. Searching for rule artifacts

Searching for rule artifacts



© Copyright IBM Corporation 2016

Course materials may not be reproduced in whole or in part without the prior written permission of IBM.

10.1

Figure 8-3. Searching for rule artifacts

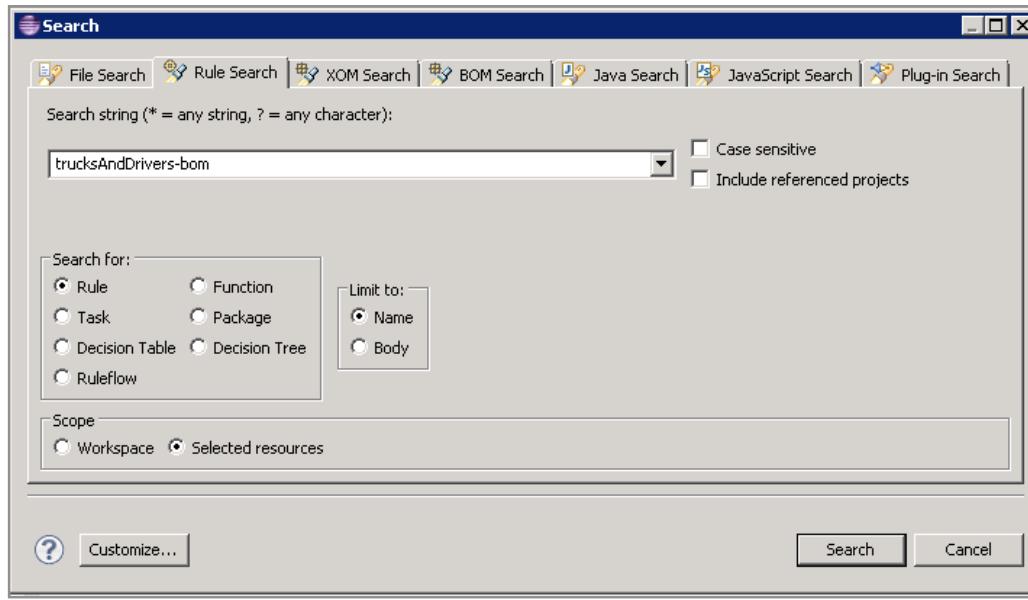
WB3951.2

Notes:



Search

- Search XOM and BOM classes by their name
- Search rule artifacts by their name or body content
- Integrated into Eclipse search



© Copyright IBM Corporation 2016

Figure 8-4. Search

WB3951.2

Notes:

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 rules that the execution of other rules affects
- Search for rules that affect the execution of other rules

© Copyright IBM Corporation 2016

Figure 8-5. Search for rule dependencies

WB3951.2

Notes:

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.

8.2. Querying rules

Querying rules



© Copyright IBM Corporation 2016

Course materials may not be reproduced in whole or in part without the prior written permission of IBM.

10.1

Figure 8-6. Querying rules

WB3951.2

Notes:



Introduction

- 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 Console

© Copyright IBM Corporation 2016

Figure 8-7. Introduction

WB3951.2

Notes:



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
 - For 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

© Copyright IBM Corporation 2016

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

WB3951.2

Notes:

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
- Use specific criteria to determine the rules that you require to create the ruleset
 - See Queries and ruleset extractors

© Copyright IBM Corporation 2016

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

WB3951.2

Notes:



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

© Copyright IBM Corporation 2016

Figure 8-10. Business Query Language (BQL)

WB3951.2

Notes:



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

© Copyright IBM Corporation 2016

Figure 8-11. Query conditions

WB3951.2

Notes:

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

© Copyright IBM Corporation 2016

Figure 8-12. Example of rule query conditions

WB3951.2

Notes:

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

© Copyright IBM Corporation 2016

Figure 8-13. Filter on properties

WB3951.2

Notes:

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).



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

© Copyright IBM Corporation 2016

Figure 8-14. Filter on definitions

WB3951.2

Notes:

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.

**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

© Copyright IBM Corporation 2016

Figure 8-15. Filter on behavior

WB3951.2

Notes:

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`
- By default, actions are not run when you run the query
- To run the actions on the query results, you must select **Run query actions** after you run the query

© Copyright IBM Corporation 2016

Figure 8-16. Query actions

WB3951.2

Notes:

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.



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`



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

© Copyright IBM Corporation 2016

Figure 8-17. Queries and ruleset extractor

WB3951.2

Notes:

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.

8.3. Extracting rulesets

Extracting rulesets



© Copyright IBM Corporation 2016

Course materials may not be reproduced in whole or in part without the prior written permission of IBM.

10.1

Figure 8-18. Extracting rulesets

WB3951.2

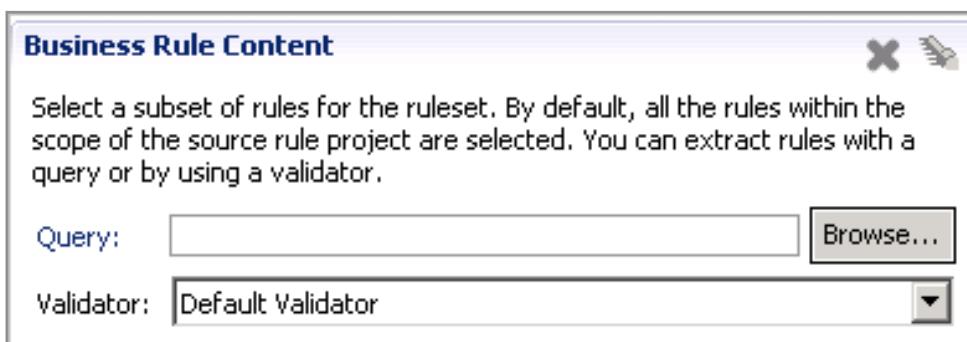
Notes:

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.



Ruleset archive

- To pass only a subset of rules from a decision operation to the rule engine:
 - Use a query to extract the rules
 - Write a validator class to select the required rules
- Specify the extractor in the decision operation on the **Overview** tab in the **Business Rule Content** section.



© Copyright IBM Corporation 2016

Figure 8-19. Ruleset archive

WB3951.2

Notes:

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

Having completed this unit, you should be able to:

- 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 2016

Figure 8-20. Unit summary

WB3951.2

Notes:



Checkpoint questions

1. Which of the following features can you use to create queries in the Query Editor to search in packages, rules, the BOM, and the XOM?
 - a. Business Query Language (BQL)
 - b. Rule Query Language (RQL)
 - c. Intellirule Query Language (IQL)
 - d. Both a and b
2. **True or False:** A query can be used to extract a subset of rules for a decision operation.

© Copyright IBM Corporation 2016

Figure 8-21. Checkpoint questions

WB3951.2

Notes:

Write your answer here:

1.

2.



Checkpoint answers

1. Which of the following features can you use to create queries in the Query Editor to search in packages, rules, the BOM, and the XOM?
 - a. Business Query Language (BQL)
2. **True.**

© Copyright IBM Corporation 2016

Figure 8-22. Checkpoint answers

WB3951.2

Notes:

Exercise 11

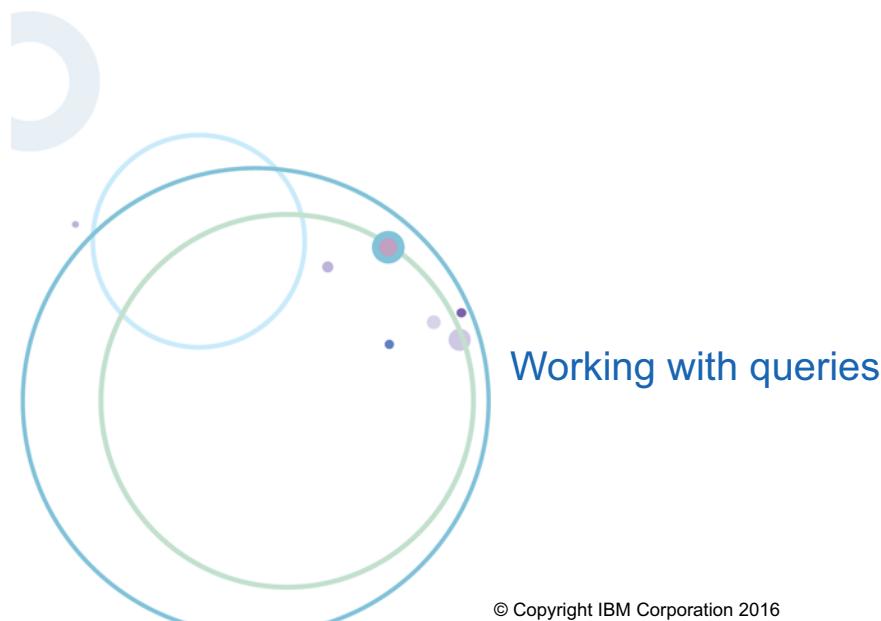


Figure 8-23. Exercise 11

WB3951.2

Notes:



Exercise objectives

After completing this exercise, you should be able to:

- Search for rule artifacts and find rules according to their dependencies
- Define and run queries and apply actions on query results

© Copyright IBM Corporation 2016

Figure 8-24. Exercise objectives

WB3951.2

Notes:

Unit 9. Debugging rules

What this unit is about

In this unit, you learn how to verify that the implemented business logic is free of errors.

What you should be able to do

After completing this unit, you should be able to:

- Use a launch configuration to execute rulesets
- Use the Rule Designer debug tools

How you will check your progress

- Checkpoint
- Exercises



Unit objectives

After completing this unit, you should be able to:

- Use a launch configuration to execute rulesets
- Use the Rule Designer debug tools

© Copyright IBM Corporation 2016

Figure 9-1. Unit objectives

WB3951.2

Notes:



Topics

- Working with launch configurations
- Using Rule Designer debug tools

© Copyright IBM Corporation 2016

Figure 9-2. Topics

WB3951.2

Notes:

9.1. Working with launch configurations



Working with launch configurations



© Copyright IBM Corporation 2016

Course materials may not be reproduced in whole or in part without the prior written permission of IBM.

10.1

Figure 9-3. Working with launch configurations

WB3951.2

Notes:



Running and debugging decision services

- Before executing rules on your application, you can run and debug rules in Rule Designer to make sure that the rules behave as expected
- You can launch the rulesets in 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
- The execution of a ruleset goes through the following steps:
 1. Instantiating a new engine or connecting to an existing engine
 2. Sending the ruleset to the engine
 3. Executing the `ilrmain` function, if it exists, or executing the ruleset in the engine

© Copyright IBM Corporation 2016

Figure 9-4. Running and debugging decision services

WB3951.2

Notes:

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:

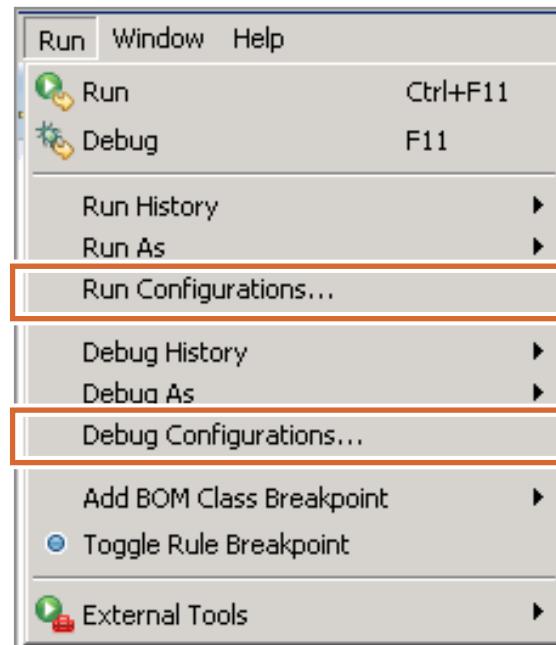
- *Run*: When you want to run the rules without debugging and check the results
- *Debug*: When you want to suspend execution and use the debug tools

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



© Copyright IBM Corporation 2016

Figure 9-5. Defining launch configuration (1 of 2)

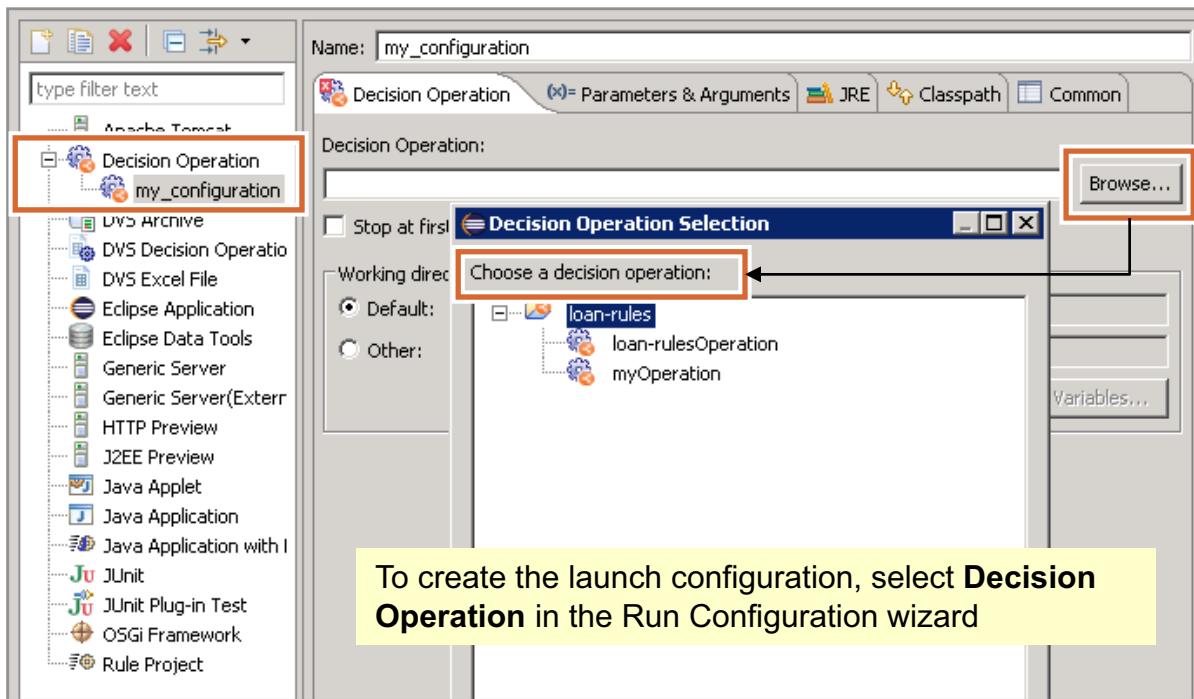
WB3951.2

Notes:

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.



Defining launch configuration (2 of 2)



© Copyright IBM Corporation 2016

Figure 9-6. Defining launch configuration (2 of 2)

WB3951.2

Notes:

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 type **Decision Operation** and choose which decision operation to run.

9.2. Using Rule Designer debug tools

Using Rule Designer debug tools



© Copyright IBM Corporation 2016

Course materials may not be reproduced in whole or in part without the prior written permission of IBM.

10.1

Figure 9-7. Using Rule Designer debug tools

WB3951.2

Notes:



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

© Copyright IBM Corporation 2016

Figure 9-8. Rule Designer debugging tools

WB3951.2

Notes:

If you must troubleshoot rulesets that fail to run properly, you can run them with the Rule Designer debug tools, and set breakpoints, step through the execution, or use expression evaluation to debug the ruleset.

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 trees, 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



Rule Designer debug views

- Debug
 - Stack trace that mixes rules and Java
- Working Memory
 - List of all objects in the working memory
- Agenda
 - List of rule instances that are scheduled for execution
- Variables
 - List of the variables that are bound in the current rule
- Breakpoints
 - List of the breakpoints on Java code, rule action, working memory
- Console
 - Messages that are sent to the engine output stream

© Copyright IBM Corporation 2016

Figure 9-9. Rule Designer debug views

WB3951.2

Notes:

The Debugger offers the following views:

- **Debug**

With the Debug view, you can debug a rule project or Java program in the workbench.

- **Working Memory**

, 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**

- **Variables**

The values in the Working Memory view are updated when the engine is updated, but the values in the Variables view are updated after the evaluation of each expression.

- **Breakpoints**

- **Console**

The Console view shows any messages that were sent as output from the rule engine.



Unit summary

Having completed this unit, you should be able to:

- Use a launch configuration to execute rulesets
- Use the Rule Designer debug tools

© Copyright IBM Corporation 2016

Figure 9-10. Unit summary

WB3951.2

Notes:



Checkpoint questions

1. Which term denotes the type of session under which you execute a rule project in Rule Designer?
 - a. Decision Warehouse
 - b. Launch configuration
 - c. BOM-to-XOM mapping
 - d. All of the above
2. Which of the following types are valid launch configuration types?
 - a. Decision Service
 - b. Rule Project
 - c. Java application with rules
 - d. Remote Java application with rules
 - e. All of the above
3. **True or False:** Debugging is a development task that is done iteratively in Rule Designer, while working with your rule artifacts.

© Copyright IBM Corporation 2016

Figure 9-11. Checkpoint questions

WB3951.2

Notes:

Write your answers here:

- 1.
- 2.
- 3.



Checkpoint answers

1. Which term denotes the type of session under which you execute a rule project in Rule Designer?
 - b. Launch configuration
2. Which of the following types are valid launch configuration types?
 - d. All of the above
3. **True:** Debugging is a step that you undertake in Rule Designer, iteratively, while you develop your business rule artifacts.

© Copyright IBM Corporation 2016

Figure 9-12. Checkpoint answers

WB3951.2

Notes:

Exercise 12



Figure 9-13. Exercise 12

WB3951.2

Notes:



Exercise objectives

After completing this exercise, you should be able to:

- Create launch configurations to run rulesets locally

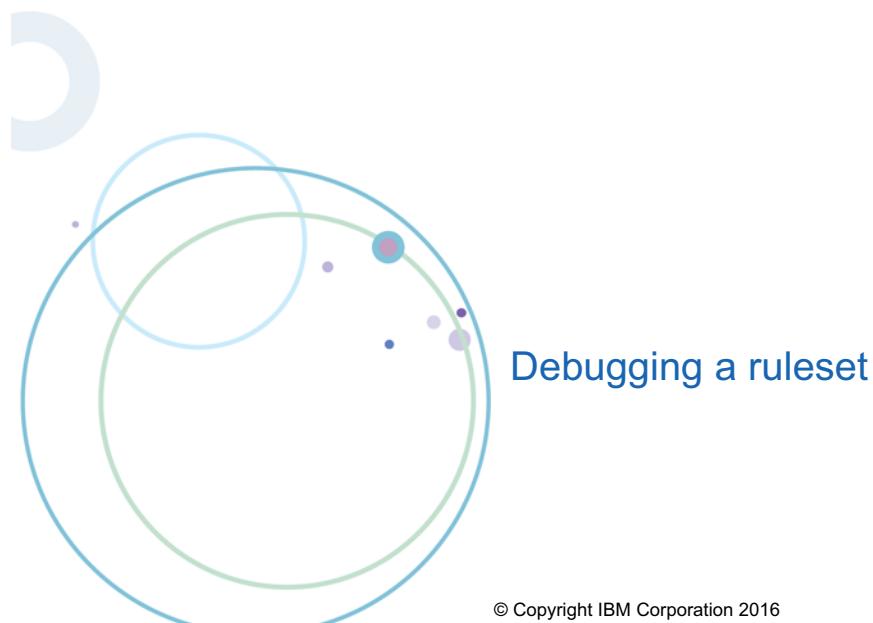
© Copyright IBM Corporation 2016

Figure 9-14. Exercise objectives

WB3951.2

Notes:

Exercise 13



© Copyright IBM Corporation 2016

Course materials may not be reproduced in whole or in part without the prior written permission of IBM.

10.1

Figure 9-15. Exercise 13

WB3951.2

Notes:



Exercise objectives

After completing this exercise, you should be able to:

- Debug a ruleset
- Set breakpoints on an action rule and on a task in a ruleflow
- Inspect objects in the working memory or rule instances in the agenda
- Use the various views of the Debug perspective

© Copyright IBM Corporation 2016

Figure 9-16. Exercise objectives

WB3951.2

Notes:

Unit 10. Enabling tests and simulations

What this unit is about

This unit teaches you how to enable business users to run tests and simulations.

What you should be able to do

After completing this unit, you should be able to:

- Describe the basic features of testing and simulation
- Collaborate with business users to set up testing and simulation

How you will check your progress

- Checkpoint
- Exercise



Unit objectives

After completing this unit, you should be able to:

- Describe the basic features of testing and simulation
- Collaborate with business users to set up testing and simulation

© Copyright IBM Corporation 2016

Figure 10-1. Unit objectives

WB3951.2

Notes:



Topics

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

© Copyright IBM Corporation 2016

Figure 10-2. Topics

WB3951.2

Notes:

10.1.Overview of testing and simulation

Overview of testing and simulation



© Copyright IBM Corporation 2016

Course materials may not be reproduced in whole or in part without the prior written permission of IBM.

10.1

Figure 10-3. Overview of testing and simulation

WB3951.2

Notes:



What is testing and simulation?

- Testing
 - Validate the correctness of a ruleset
 - Compare expected results with the actual results of ruleset execution
- Simulations
 - Evaluate the potential impact of changes to rules (“what-if” analysis)
 - Results are shown in key performance indicators (KPIs)
- Scenarios
 - Provide data for tests and simulations
 - Use real or fictitious use cases

© Copyright IBM Corporation 2016

Figure 10-4. What is testing and simulation?

WB3951.2

Notes:

Use the testing and simulation features to validate rules in development and business user environments.

Testing enables you to validate rules by comparing expected results with the actual results obtained from running the rules.

Simulations enable you to see the effects of changes to rules and data through key performance indicators (KPIs).

You can support business users by preparing the tests and simulations by creating scenario files, KPIs, and report formats.

Decision Runner and Scenario Service Provider (SSP) (1 of 2)

- Runtime service applications used as test and simulation engines
 - Decision Runner
 - SSP
- Decision Runner
 - Used by Business console
 - Supports the classic rule engine and decision engine
- SSP
 - Used by Enterprise console (on-premises only)
 - Supports the classic rule engine only
- Input for tests and simulations:
 - Scenarios and rules are submitted to the runtime service
 - For tests, you also submit expected results
 - The SSP sends the rules and scenarios to Rule Execution Server, which executes the rules on the scenarios

© Copyright IBM Corporation 2016

Figure 10-5. Decision Runner and Scenario Service Provider (SSP) (1 of 2)

WB3951.2

Notes:

To run tests and simulations, you use a test and simulation engine. In Decision Center supports two runtime service applications for running tests and simulations:

- Decision Runner
- Scenario Service Provider (SSP)

These runtime services are web applications that are 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.

**Important****IBM ODM on Cloud**

Testing and simulation is not supported by the Enterprise console in the cloud. The Enterprise console does not support the decision engine, and it does not support the Decision Runner.



Decision Runner and Scenario Service Provider (SSP) (2 of 2)

- 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
- To integrate test execution in your build systems, use Operational Decision Manager APIs to run the SSP or Decision Runner

© Copyright IBM Corporation 2016

Figure 10-6. Decision Runner and Scenario Service Provider (SSP) (2 of 2)

WB3951.2

Notes:

The runtime service returns a report that details whether each scenario passed or failed.

For simulations, The runtime service returns a report with 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 or the SSP so that you can integrate test execution into your own build systems.

What are scenarios?

- Fictional or actual business data that you use as input for both testing and simulation
- Works with Microsoft Excel 2003 or later
- Microsoft Excel scenario files are designed for:
 - Projects with relatively simple and small object models
 - Testing that uses thousands of scenarios
- Scenario file templates can be generated in Rule Designer or Decision Center

© Copyright IBM Corporation 2016

Figure 10-7. What are scenarios?

WB3951.2

Notes:

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)

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

© Copyright IBM Corporation 2016

Figure 10-8. Example scenario: Loan application

WB3951.2

Notes:

For example, the following Microsoft Excel sheet contains these scenarios:

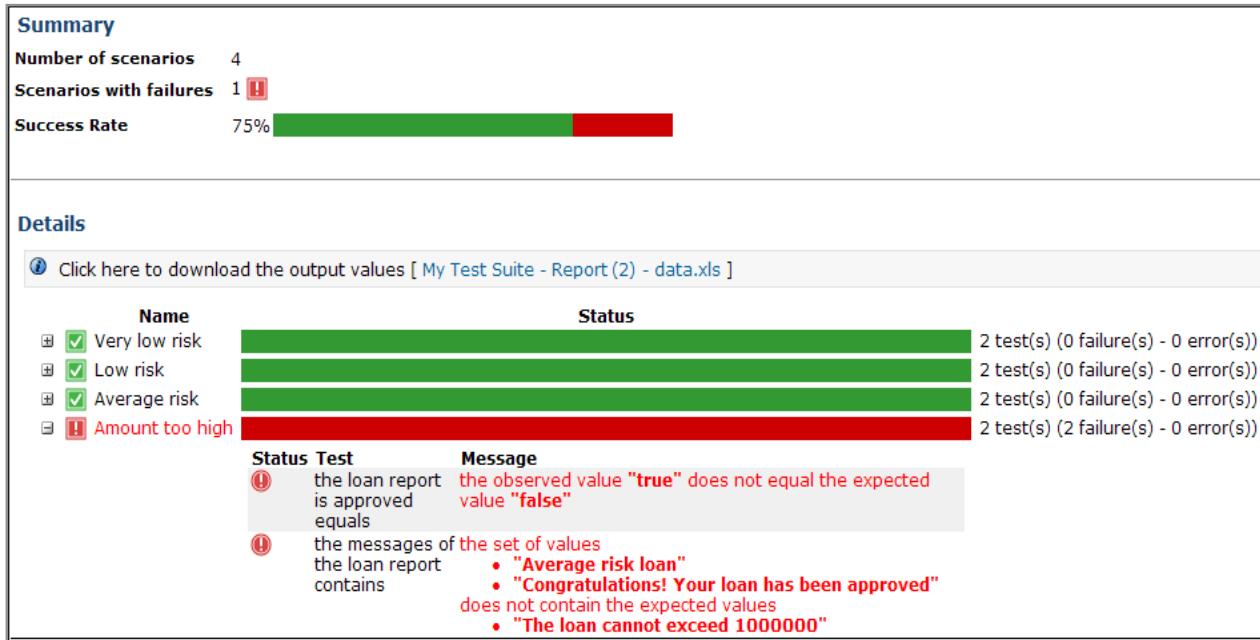
- A scenario that is named Big Loan to see how your rules work with an important loan request
- A scenario that is named Small Loan to see how your rules work with a smaller loan request

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 Microsoft Excel scenario file templates in Decision Center Enterprise and Business consoles 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.



Reports

- The runtime service returns the information that is generated during execution in a detailed report that can be viewed in the Business console



© Copyright IBM Corporation 2016

Figure 10-9. Reports

WB3951.2

Notes:

Running a test suite or simulation generates a large amount of information. A report summarizes the generated information, and can be viewed in Rule Designer or Decision Center consoles. The graphic shows an example report in Decision Center.

- The **General information** section includes the version of the test suite or simulation and the scenarios used.
- The **Summary** section shows the number of scenarios that are used and the success rate, that is, the percentage of scenarios that were executed successfully.

The **Details** section shows the decision trace and the results of each test on each scenario. Test results can be:

- Success:** Expected results match the actual results
- Failure:** Expected results are different from the actual results
- Error:** Scenarios cannot be executed because of improper formatting or other errors

The **Details** section also contains a link to the output values file.



Hint

If business users run into problems or do not get the results that they expect, they can send you a file that contains everything that is required to reproduce their test environment.



KPI results

- KPIs define:
 - How the performance of a simulation is calculated
 - How results are presented in the simulation report
- Multiple KPIs can be used for each simulation
- KPIs can be defined in Business console
- For simulations, the **Key Performance Indicators** section of the report provides some business-relevant interpretation of the results

© Copyright IBM Corporation 2016

Figure 10-10. KPI results

WB3951.2

Notes:



Validation services in Operational Decision Manager

- Rule Designer: Decision Validation Services (DVS)
 - Set up and customize testing and simulation for business users
 - Define, run, and debug artifacts
 - Test the remote conditions to ensure that tests and simulations can be run from Decision Center
- Decision Center Enterprise console: SSP
 - Create and run test suites and simulations to validate rule changes
- Decision Center Business console: Decision Runner
 - Create and run test suites and test rule changes
- Rule Execution Server:
 - Execute tests and simulations

© Copyright IBM Corporation 2016

Figure 10-11. Validation services in Operational Decision Manager

WB3951.2

Notes:

As part of business rule management, business users can use Decision Validation Services to test newly authored rules and to ensure that the original rules from previous releases continue to work properly. Business users also run simulations during rule analysis and authoring to test the effects of rule changes.

In Operational Decision Manager, Decision Validation Services integrates in these modules:

- **Rule Designer:**
 - You set up and customize testing and simulation for business users.
 - You test the remote conditions to ensure that tests and simulations can be run from Decision Center.
- **Decision Center consoles:**
 - Business users create and run tests and simulations directly in the Business and Enterprise consoles to validate rule changes.
 - You can store test suites and simulations in Decision Center database.

- You can query test and simulation artifacts, and manage their versions, like other rule artifacts
- **Rule Execution Server:** During both testing and simulation, Rule Execution Server is used to execute the rules on scenarios. Rule Execution Server must be Installed on an application server and configured with SSP for Rule Designer and the Enterprise Console, and Decision Runner for the Decision Center.



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

© Copyright IBM Corporation 2016

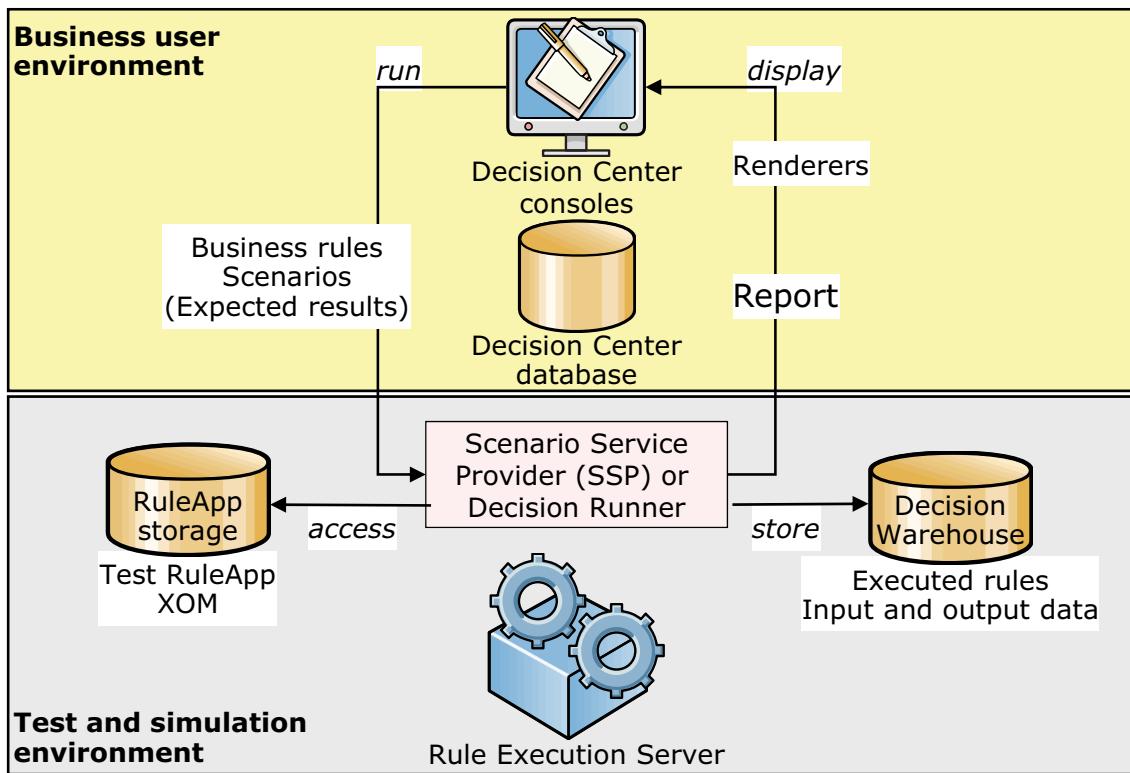
Figure 10-12. Testing and simulation in the decision lifecycle

WB3951.2

Notes:

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.

How does it work?



© Copyright IBM Corporation 2016

Figure 10-13. How does it work?

WB3951.2

Notes:

Setting up the test environment for business users involves these tasks in Rule Designer:

- Make sure that the decision service is set up correctly for testing
- Define, run, and debug scenarios and other artifacts
- Make sure that a runtime service application (either Decision Runner or the SSP) is deployed to the application server with Rule Execution Server
- Publish the decision service to Decision Center
- Test the remote conditions to ensure that tests and simulations can be run from Decision Center

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 and Enterprise consoles. Scenarios in the Excel format are stored in the Decision Center database, and they are queried and version-controlled like other rule artifacts.

During both testing and simulation, the runtime service provides the rules and scenarios to Rule Execution Server. The runtime service requests Rule Execution Server to execute the rules on the scenarios.

After execution, the runtime service returns an HTML report with detailed results of the test or simulation. Renderers determine how the results are presented in Decision Center.

Test execution results can be stored in Decision Warehouse. However, the best way to audit results is by using the detailed report in Decision Center.

10.2. Setting up testing and simulation

Setting up testing and simulation



© Copyright IBM Corporation 2016

Course materials may not be reproduced in whole or in part without the prior written permission of IBM.

10.1

Figure 10-14. Setting up testing and simulation

WB3951.2

Notes:



Setting up testing and simulation functionality

- Varying levels of collaboration with business users might be required for some tasks
 - Define the rule project with the correct ruleset parameters, which determine the scenario values to test
 - Validate the BOM and generate complex Microsoft Excel scenario file templates
 - Customize the test and simulation features, such as formats or reports
- Publish the decision service to Decision Center
- Make sure that the XOM is accessible to Rule Execution Server
 - When you publish the decision service to Decision Center, the XOM is included
 - When business users deploy for testing, the XOM is also deployed
- In Decision Center, developers or administrators ensure that the test server is accessible

© Copyright IBM Corporation 2016

Figure 10-15. Setting up testing and simulation functionality

WB3951.2

Notes:

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.

To set up testing and simulation, you publish the decision service projects to Decision Center and make sure that the XOM is accessible to the Rule Execution Server.

Varying levels of collaboration with business users might be required for some tasks:

- Defining the rule project with the correct ruleset parameters, which determine the scenario values to test
- Validating the BOM and generate complex Microsoft Excel scenario file templates
- Customizing the test and simulation features, such as formats and reports

Business users can generate and populate scenario file templates in Decision Center, but they might need your help for more complex scenario files. In Business console, business users can also define KPIs and report formats.

For more information about particular tasks, see the samples that are provided with your product installation.

Generating scenario file templates

- To enable business users to test a ruleset
 - Validate the BOM and ensure that scenario file templates can be generated properly
 - Publish the validated decision service to Decision Center
- Scenario file templates are generated in Business console
- Populate the scenario file with test data taken from a database or another source

© Copyright IBM Corporation 2016

Figure 10-16. Generating scenario file templates

WB3951.2

Notes:

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. Microsoft Excel spreadsheets are the default format for scenario files. Business users can generate templates from Business console.

You learn how to work with templates during the exercise that is associated with this unit.

After you generate a scenario file template, it can be populated with test data taken from a database or another source. In Rule Designer, you can use the DVS APIs to populate the Microsoft Excel scenario file templates with data stored in XML files or in an external database. A prepopulated scenario file can save much time and effort, particularly with complex object models, and allows business users to focus on defining the tests and modifying data values that are relevant to the tests. Prepopulated scenarios can use test values that business users can modify, or real data values that are obtained from a database or other source.

10.3. Working with scenarios

Working with scenarios



© Copyright IBM Corporation 2016

Course materials may not be reproduced in whole or in part without the prior written permission of IBM.

10.1

Figure 10-17. Working with scenarios

WB3951.2

Notes:

Excel scenario files and the BOM

- Excel 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 Excel scenario files and classes in the BOM
- You must understand your object model and the main objects you are sending as scenario data

© Copyright IBM Corporation 2016

Figure 10-18. Excel scenario files and the BOM

WB3951.2

Notes:

Excel 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.

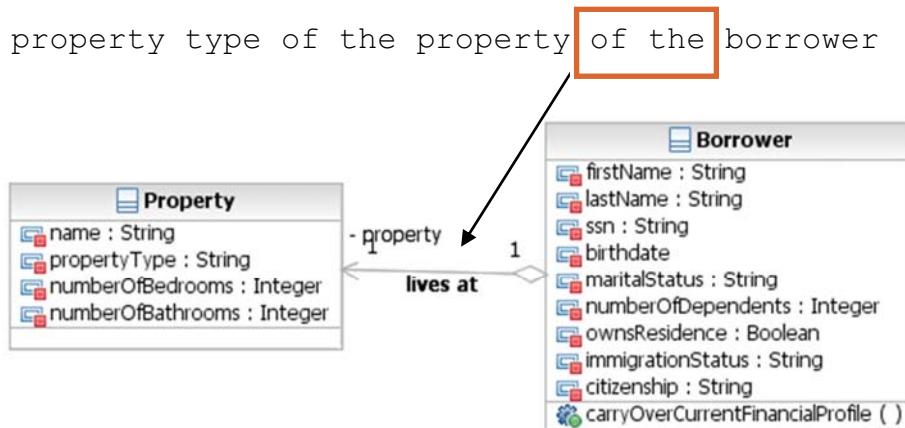
Before generating the scenario file templates, you must validate that the BOM is properly configured to use with Decision Validation Services so that columns in the **Scenarios** and **Expected Results** sheets generate correctly.

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 they are with Decision Validation Services
- In Decision Center, the “of the” constructions in the rule statements indicate relationships between objects

the property type of the property **of the borrower**



© Copyright IBM Corporation 2016

Figure 10-19. Relationships in the BOM (1 of 2)

WB3951.2

Notes:

Relationships within the BOM are translated differently in Decision Center than they are in Decision Validation Services.

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 Microsoft Excel 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 [Compatibility Mode] - Microsoft Excel". The "Scenarios" tab is active, displaying a table with 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, and amount. A row for "Scenario 1" is present. Below the table, there is a note: "Create your scenarios. Click here to access the help sheet". The "PropertyInfo" tab is also visible at the bottom of the ribbon.

Figure 10-20. Relationships in the BOM (2 of 2)

WB3951.2

Notes:

With Decision Validation Services, 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 in Microsoft Excel (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

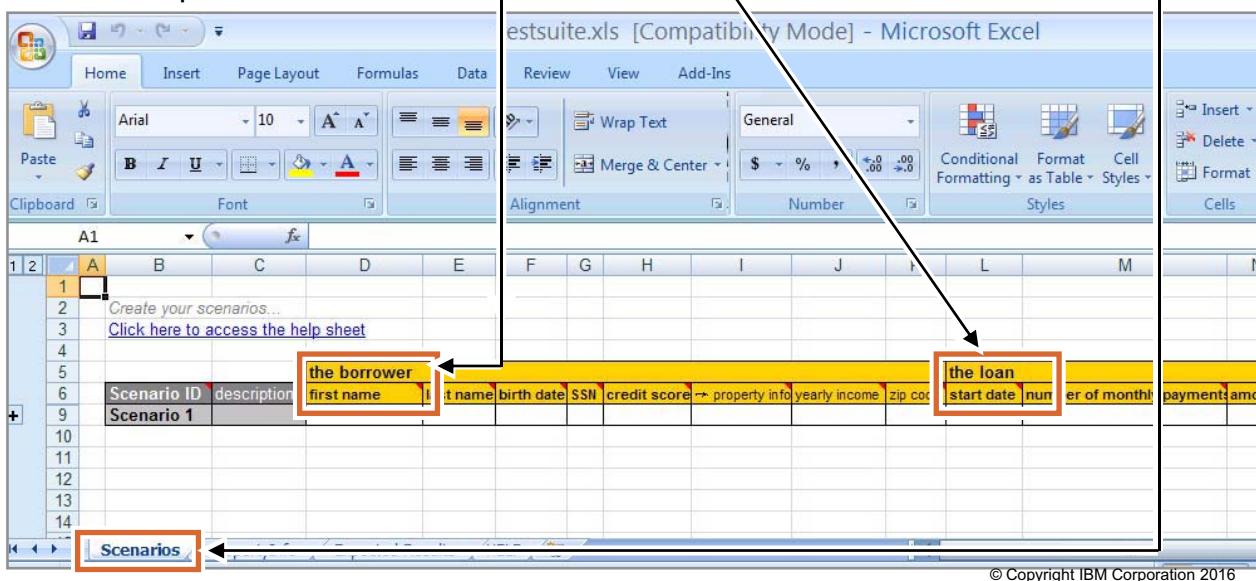


Figure 10-21. Structure of the scenario files in Microsoft Excel (1 of 3)

WB3951.2

Notes:

Microsoft Excel 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** value is **PropertyInfo1**

The screenshot shows a Microsoft Excel spreadsheet titled "testsuite_2.xls [Compatibility Mode]". The spreadsheet has three tabs at the bottom: "Scenarios", "PropertyInfo", and "Expected Results". The "Scenarios" tab is selected. The data starts with a header row (row 5) containing "the borrower" and "the loan". Below this, there are two rows of data. The first data row (row 6) has columns for "Scenario ID" (bolded), "description" (bolded), "first name", "last name", "birth date", "SSN", and "credit sco". The second data row (row 9) has columns for "Scenario ID" (bolded), "description" (bolded), "first name" (bolded), "last name", "birth date", "SSN", and "credit sco". The "first name" cell in the second row contains "Joe". The "last name" cell contains "Smith". The "birth date" cell contains "4/12/1983". The "SSN" cell contains "123456789". The "credit sco" cell contains "60". The "property info" cell in the second row is highlighted with a red box and contains "PropertyInfo1". The "loan" section of the table continues with columns for "early income", "zip code", "start date", and "number of mon". The "early income" cell contains "52,500". The "zip code" cell contains "12345". The "start date" cell contains "4/1/2010". The "number of mon" cell is empty.

Figure 10-22. Structure of the scenario files in Microsoft Excel (2 of 3)

WB3951.2

Notes:

The first column in the **Scenarios** sheet is the **Scenario ID**, which uniquely identifies each scenario.

The **Scenarios** sheet includes:

- Required columns: These columns provide the data that is required to execute your rulesets. The name of a required column is shown in bold in the Microsoft Excel sheet.
Required columns are defined through a constructor of the BOM class. You must define a constructor. When you have more than one constructor, choose the constructor with arguments that you want included in the template. You might need to modify the names of the constructor arguments to make them meaningful to business users.
- Optional columns: Optional columns correspond to attributes of the BOM class. If the attribute is verbalized, the column heading uses the verbalization of the attribute.

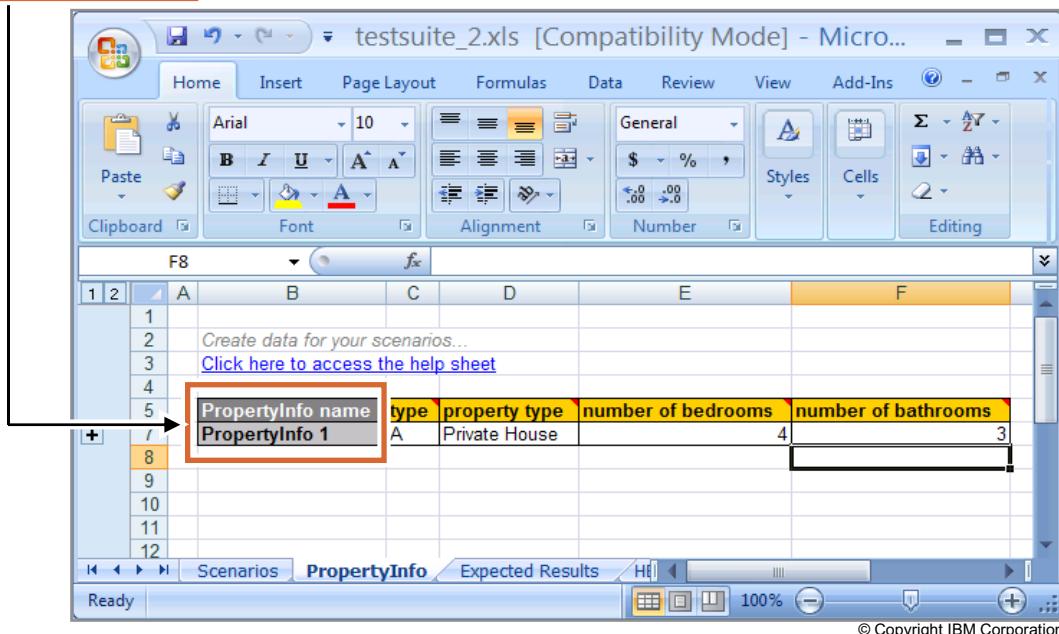


Information

Only attributes that are defined as “**Read/Write**” or “**Write Only**” can be used to create optional columns in the **Scenarios** sheet.

Structure of the scenario files in Microsoft Excel (3 of 3)

- The spreadsheet tab for the child class uses the class ID to show which scenario the values are related to
 - PropertyInfo1** identifies the attribute values for **property info** in **Scenario 1**



The screenshot shows a Microsoft Excel window with the title 'testsuite_2.xls [Compatibility Mode] - Micro...'. The 'PropertyInfo' tab is selected. The first few rows of the sheet contain the following data:

PropertyInfo name	type	property type	number of bedrooms	number of bathrooms
PropertyInfo1	A	Private House	4	3

© Copyright IBM Corporation 2016

Figure 10-23. Structure of the scenario files in Microsoft Excel (3 of 3)

WB3951.2

Notes:

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

	Scenario ID	the yearly repayment equals	the loan report is approved equals	the message equals
5	Big Loan	39597	FALSE	Too big Debt-To-Income ratio
9	Small Loan	1979	TRUE	Loan approved
10				
11				

Scenarios Expected Results HELP

© Copyright IBM Corporation 2016

Figure 10-24. Expected results (1 of 2)

WB3951.2

Notes:

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

© Copyright IBM Corporation 2016

Figure 10-25. Expected results (2 of 2)

WB3951.2

Notes:

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

© Copyright IBM Corporation 2016

Figure 10-26. Adding and removing columns in the Microsoft Excel file

WB3951.2

Notes:

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

© Copyright IBM Corporation 2016

Figure 10-27. Adding columns with virtual attributes

WB3951.2

Notes:

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 Microsoft Excel file

- Columns in the Microsoft Excel 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

© Copyright IBM Corporation 2016

Figure 10-28. Removing columns in the Microsoft Excel file

WB3951.2

Notes:

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

Having completed this unit, you should be able to:

- Describe the basic features of testing and simulation
- Collaborate with business users to set up testing and simulation

© Copyright IBM Corporation 2016

Figure 10-29. Unit summary

WB3951.2

Notes:



Checkpoint questions

- 1. True or False:** Testing allows business users to validate the behavior of a ruleset by comparing the expected results with KPIs.
- 2. True or False:** Scenarios represent fictitious or actual business data that you use as input for both testing and simulation and can store in Microsoft Excel format.
- 3. True or False:** Business users can run tests and simulations with minimal dependence on the development team.
- 4. True or False:** You must always generate and define test suites and simulations in Rule Designer, and then publish them to Decision Center for business users.

© Copyright IBM Corporation 2016

Figure 10-30. Checkpoint questions

WB3951.2

Notes:

Write your answers here:

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



Checkpoint answers

1. **False:** *With testing, you validate the behavior of a ruleset by comparing the results that you expect with the results obtained during execution.*
2. **True.**
3. **True.**
4. **False:** *Business users can generate and define scenarios directly in Decision Center without help from developers.*

© Copyright IBM Corporation 2016

Figure 10-31. Checkpoint answers

WB3951.2

Notes:

Exercise 14



© Copyright IBM Corporation 2016

Course materials may not be reproduced in whole or in part without the prior written permission of IBM.

10.1

Figure 10-32. Exercise 14

WB3951.2

Notes:



Exercise objectives

After completing this exercise, you should be able to:

- Validate the BOM and generate scenario file templates in Excel format
- Customize scenario file templates
- Create virtual attributes to test collections of complex objects
- Validate remote testing conditions for business users in the Business console

© Copyright IBM Corporation 2016

Figure 10-33. Exercise objectives

WB3951.2

Notes:

Unit 11. Managing deployment

What this unit is about

This unit teaches you how to deploy the rule artifacts to Rule Execution Server for their managed execution. It also covers XOM management.

What you should be able to do

After completing this unit, you should be able to:

- Explain the basic features of Rule Execution Server
- Prepare deployment configurations
- Deploy RuleApps to Rule Execution Server
- Describe the basics of XOM management

How you will check your progress

- Checkpoint
- Exercise



Unit objectives

After completing this unit, you should be able to:

- Explain the basic features of Rule Execution Server
- Prepare deployment configurations
- Deploy RuleApps to Rule Execution Server
- Describe the basics of XOM management

© Copyright IBM Corporation 2016

Figure 11-1. Unit objectives

WB3951.2

Notes:



Topics

- Introducing managed execution
- Deploying decision services
- Managing XOMs

© Copyright IBM Corporation 2016

Figure 11-2. Topics

WB3951.2

Notes:

11.1. Introducing managed execution

Introducing managed execution



© Copyright IBM Corporation 2016

Course materials may not be reproduced in whole or in part without the prior written permission of IBM.

10.1

Figure 11-3. Introducing managed execution

WB3951.2

Notes:



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

© Copyright IBM Corporation 2016

Figure 11-4. Introduction to Rule Execution Server

WB3951.2

Notes:

Rule Execution Server is the managed and monitored environment for executing business rules. You use it to simultaneously execute several rulesets, update rulesets, or manage transactions. Rule Execution Server targets, and provides services for, the Java SE and Java EE environments, and client development and deployment cycles. It also handles the creation, pooling, and management of ruleset instances, and makes execution of business rules as simple as possible in your application.



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

© Copyright IBM Corporation 2016

Figure 11-5. Key functions of Rule Execution Server

WB3951.2

Notes:

More specifically, Rule Execution Server incorporates the following key features that provide enterprise-level ruleset execution and management facilities:

- Execution scalability by using resource pooling
- Management through a web-based interface and JMX tools
- Full integration in Java EE and Java SE
- Automation with Ant tasks
- Logging, monitoring, and debugging integration
- Integration into the development process

Rule Execution Server supports clustering and addresses many of the demands of 24x7 production applications. It can handle business rule execution for multiple business rule applications, and is designed for scalability.

You learn more about Rule Execution Server in Unit 12, "Executing rules with Rule Execution Server".



Ruleset execution with Rule Execution Server

- Design and create a RuleApp from a rule project
- Set up Rule Execution Server on an application server
- Deploy the RuleApp to one or more Rule Execution Servers
- Design and create a client application that uses the appropriate rule session
- Run your client application to test it, and correct any errors that you find
- Tune and administer Rule Execution Server and decision services to provide stability and performance

© Copyright IBM Corporation 2016

Figure 11-6. Ruleset execution with Rule Execution Server

WB3951.2

Notes:

Executing rulesets with Rule Execution Server involves different steps:

1. Design and create a RuleApp from a rule project.

A RuleApp is a deployment and management unit for Rule Execution Server.

A RuleApp contains one or more rulesets.

2. Set up and configure Rule Execution Server on an application server.

In this course, Rule Execution Server is installed on the Sample Server.

3. Deploy the RuleApp to one or more Rule Execution Servers.

You deploy your RuleApps to the application server to make the ruleset available to a client application.

In your enterprise environment, you might have several Rule Execution Servers running on different application servers, such as:

- Test servers:

By default, Rule Execution Server is installed on the Sample Server, which is used to run the provided samples.

You can use this server to test your installation of Rule Execution Server.

- Production servers:

Set up and configure Rule Execution Server and Decision Warehouse on a production application server cluster.

You then redeploy the validated RuleApps and client applications.

Your production server can involve two or more application servers in a cluster that are installed on two or more computers.

4. Design and create a client application that uses the appropriate rule session.

5. Run your client application to test it, and correct any errors that you find.

You can test by using the client application.

You can also use Decision Validation Services to verify whether your RuleApps return valid results.

You can set up Decision Warehouse to store execution traces.

6. Tune and administer (that is, maintain and monitor) your Rule Execution Server and decision services to provide stability and performance.

You do administration tasks by using the Rule Execution Server console.

Although not listed in the slide, tuning and improving the performance of the production server is also key to giving the most efficient performance possible.

- The goal of performance tuning is to decrease the amount of time and resources that your application server requires to process requests.
- You can configure the following server settings in your cluster to achieve the best performance:
 - Log
 - Trace level
 - Thread pool
 - Garbage collection
- You can also look at the rulesets that your decision services, and improve performance by reducing the ruleset size and by using mutually exclusive rule tasks.

Setting up a Rule Execution Server

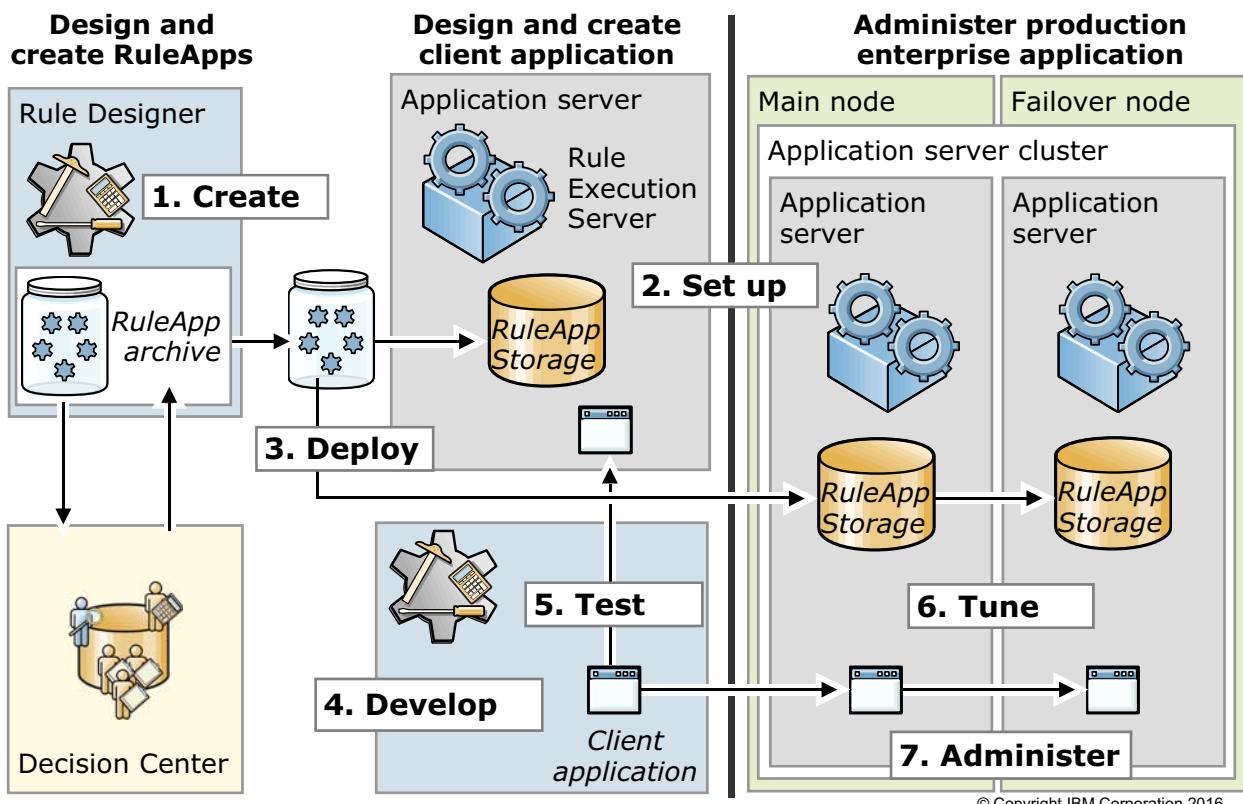


Figure 11-7. Setting up a Rule Execution Server

WB3951.2

Notes:

Architects, administrators, developers, and testers work together to create, deploy, tune, and administer RuleApps and client applications that run on Rule Execution Server instances on a production enterprise cell.

To use Rule Execution Server, follow the task flow to complete the steps that are applicable to you. The order in which tasks are done depends on your environment.

The diagram shows the task flow to set up a Rule Execution Server, and the infrastructure that is used at each step. The highlighted steps are administrative tasks that are explained in more detail on the next slide.

1. Create

Developers design and create a RuleApp from a rule project. A RuleApp is a deployment and management unit for Rule Execution Server. A RuleApp contains one or more rulesets. RuleApps are deployed to the application server to make the ruleset available to a client application.

2. Set up

Administrators set up test servers and production servers. For more information, see the next slide.

3. Deploy

Developers or administrators can deploy the RuleApp to one or more Rule Execution Servers.

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 use Decision Validation Services 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

Administrators can tune Rule Execution Server and server settings in a production cluster to achieve the best performance. 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.

This unit, and the following ones, cover the steps of the ruleset execution with Rule Execution Server.

11.2. Deploying decision services

Deploying decision services



© Copyright IBM Corporation 2016

Course materials may not be reproduced in whole or in part without the prior written permission of IBM.

10.1

Figure 11-8. Deploying decision services

WB3951.2

Notes:



Deployment principles

- Prerequisites:
 - Rulesets are defined as decision operations
 - Target architecture is known
- Deployment phases:
 - Create a deployment configuration
 - Define the decision operation to deploy
 - Define the target server in the deployment configuration
 - Deploy to the identified Rule Execution Server

© Copyright IBM Corporation 2016

Figure 11-9. Deployment principles

WB3951.2

Notes:

The next steps after business rule development are to package and deploy the rules to your execution environment. For managed execution of rules with Rule Execution Server, rulesets are packaged in *RuleApps* before you deploy them to Rule Execution Server.



RuleApp

- Deployable management unit that contains one or more rulesets
- A deployed ruleset contains an archive, which supports its execution
- For classic rule projects, RuleApps are handled inside RuleApp projects
 - Contain the RuleApp descriptor and information such as the Rule project dependencies and the ruleset archive paths
- For decision services, the deployment configuration generates the RuleApp archive that contains the rulesets during the deployment process

© Copyright IBM Corporation 2016

Figure 11-10. RuleApp

WB3951.2

Notes:

A RuleApp is a deployable management unit that contains one or more rulesets. A deployed ruleset contains an archive, which supports its execution. A RuleApp is described in an XML file that is named a RuleApp descriptor. All executable resources are defined as elements of the RuleApp descriptor.

When you deploy a decision service with a deployment configuration, Rule Designer generates a RuleApp archive that contains the rulesets. These rulesets are defined in the decision operations in the decision service.



Ruleset path

- Each ruleset in a RuleApp can be identified by using a ruleset path:
`/ {RuleApp name} [/ {version}]/{ruleset name} [/ {version}]`
- The ruleset path acts as the entry point for clients to access the business logic that is encapsulated in the RuleApp
- In a client application, the ruleset path identifies the ruleset to execute
- You can omit the version fields in the ruleset path to refer to the latest deployed ruleset

© Copyright IBM Corporation 2016

Figure 11-11. Ruleset path

WB3951.2

Notes:

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.



RuleApp archive

- **Reminder:** You create a ruleset archive to pass a ruleset to the rule engine for its execution
- Similarly, you can create a RuleApp archive from a RuleApp, and then deploy this RuleApp archive to create the RuleApp on Rule Execution Server
- Rule Designer generates the RuleApp archive when you deploy the RuleApp to Rule Execution Server

© Copyright IBM Corporation 2016

Figure 11-12. RuleApp archive

WB3951.2

Notes:

You must create a ruleset archive to pass your rules to the rule engine for their execution.

Similarly, you create a RuleApp archive from your RuleApp, and deploy this RuleApp archive to create the RuleApp on Rule Execution Server.

You can export a RuleApp archive by opening the `archive.xml` file in your RuleApp project and by selecting **Export a RuleApp archive** on the **Overview** tab.

In Rule Designer, when you ask to deploy the RuleApp to Rule Execution Server, Rule Designer creates the RuleApp archive for you.



RuleApp properties and ruleset properties

- Several RuleApp properties and ruleset properties are predefined to handle the most common requirements in terms of configuration of the rule engine behavior at run time
- Examples:
 - ruleset.sequential.trace.enabled
 - ruleapp.interceptor.classname
- You can set properties to a RuleApp, applicable to all rulesets in this RuleApp
- You can set properties to a ruleset, applicable to this ruleset only

© Copyright IBM Corporation 2016

Figure 11-13. RuleApp properties and ruleset properties

WB3951.2

Notes:

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.

You can set properties on your RuleApp. These properties apply to all the rulesets in this RuleApp. To set RuleApp properties, open the `archive.xml` file in your RuleApp project, and use the **RuleApp Properties** grid on the **Overview** tab to add, edit, or remove RuleApp properties.

You can also set properties on your rulesets in your RuleApp. These properties apply to your ruleset only. To set ruleset properties, you open the `archive.xml` file in your RuleApp project. On the **Ruleset Archives** tab, select the ruleset where to set properties. You can then add, edit, or remove ruleset properties in the **Ruleset Properties** grid.



Note

You can also set RuleApp properties and ruleset properties by using the Rule Execution Server console.



Deployment configurations (1 of 4)

- From Rule Designer and Decision Center Business console, you can deploy RuleApps by using a deployment configuration
- Decision operations
 - Configurations can reference one or more decision operations
- Target servers
 - Configurations can reference one or more target servers
 - Target servers can be defined for nonproduction or production
- Create more than one deployment configuration for a decision service
 - Each deployment configuration can serve a different purpose
 - Example, one can be used for testing and another for deploying to a production server

© Copyright IBM Corporation 2016

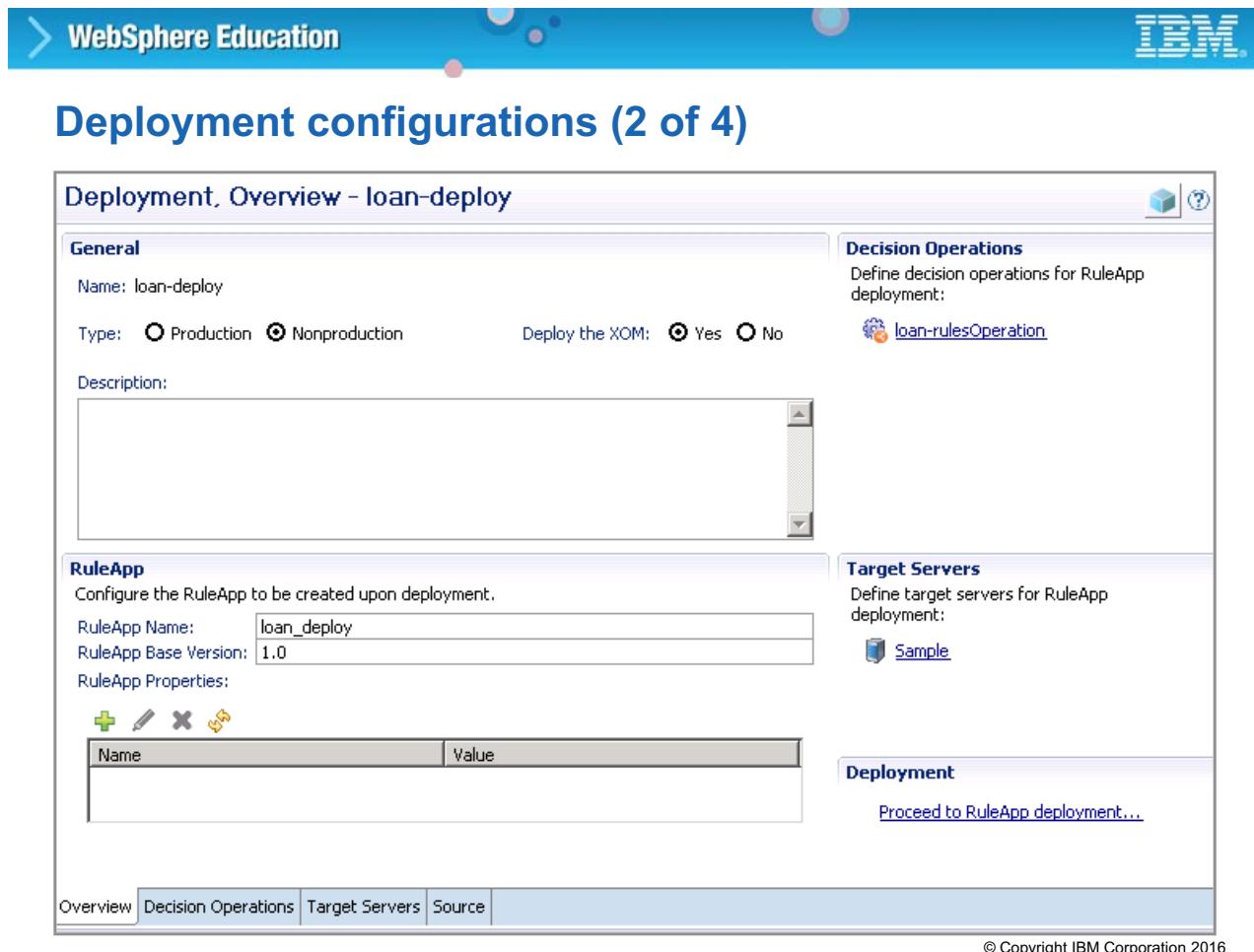
Figure 11-14. Deployment configurations (1 of 4)

WB3951.2

Notes:

To deploy your RuleApps from Rule Designer or from Decision Center Business console, you create a *deployment configuration* to tell Rule Designer which Rule Execution Server instance is your target server.

You indicate whether a deployment configuration is for a nonproduction or production environment. The nonproduction setting limits the target servers that can be used with the configuration. It also adds some restrictions on deployment from the Business console within the governance framework. When a release of a decision service is complete in the governance framework, you can deploy it to a production server.



Deployment, Overview - loan-deploy

General

Name: loan-deploy

Type: Production Nonproduction Deploy the XOM: Yes No

Description:

Decision Operations
Define decision operations for RuleApp deployment:
[loan-rulesOperation](#)

RuleApp
Configure the RuleApp to be created upon deployment.

RuleApp Name:
RuleApp Base Version:
RuleApp Properties:

Name	Value

Target Servers
Define target servers for RuleApp deployment:
[Sample](#)

Deployment
[Proceed to RuleApp deployment...](#)

Overview | Decision Operations | Target Servers | Source

© Copyright IBM Corporation 2016

Figure 11-15. Deployment configurations (2 of 4)

WB3951.2

Notes:

A deployment configuration includes the following information:

- A configuration type: production or nonproduction. This option can be used to limit deployment to certain servers.
- The RuleApp name that is used by the client application to request a decision.
- The decision operations with the rules to be deployed.
- The target servers.
- The ruleset version base number and version policy.



Deployment configurations (3 of 4)

Deployment, Decision Operations - loan-deploy

Configured Decision Operations
Select and configure decision operations for RuleApp deployment:

Ruleset General Behavior

Base Version:
 Enabled Debug Trace

Ruleset Properties
Add predefined or custom properties to the ruleset to specify ex information.

Name	Value

Ruleset Version Policy
Select the version policy that applies upon deployment:

- Increment minor version numbers
Updates the minor version for each ruleset. Makes the new version available but retains previous versions.
- Use the base version numbers
Uses the numbers provided in the deployment configuration. Replaces the latest version of each ruleset with this release. Used for hot fixes or
- The user can define the version numbers

Overview | Decision Operations | Target Servers | Source

© Copyright IBM Corporation 2016

Figure 11-16. Deployment configurations (3 of 4)

WB3951.2

Notes:

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

Figure 11-17. Deployment configurations (4 of 4)

WB3951.2

Notes:

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

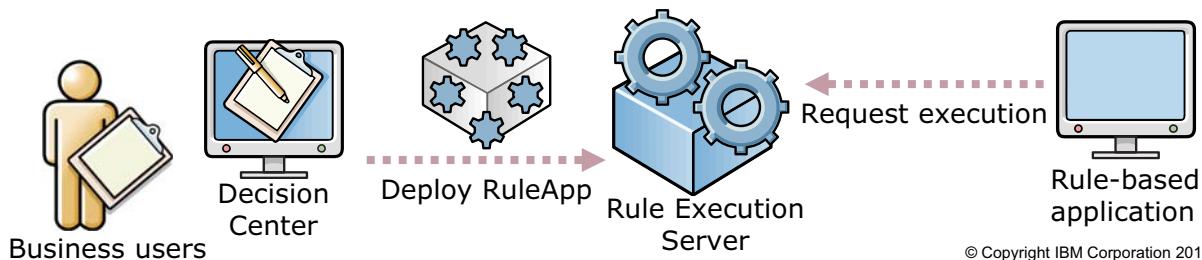


Figure 11-18. Deployment from Decision Center

WB3951.2

Notes:

Business users with administrator rights can deploy RuleApps from Decision Center to Rule Execution Server. They can create or edit deployment configurations.



Important

From Rule Designer, you define the decision operations. Decision operations cannot be edited in the Business console. Business users can edit which decision operations are referenced by a deployment configuration, but they cannot create or edit the decision operation.

If your decision service uses a managed Java XOM, deploy the Java XOM to the appropriate Rule Execution Server instance first. Then, publish your decision service or rule project from Rule Designer to Decision Center, and deploy from there.



Deployment from Rule Execution Server console

- Administrators and developers use the Rule Execution Server console to administer the Rule Execution Server
- As part of their administrative tasks, they can deploy RuleApps

© Copyright IBM Corporation 2016

Figure 11-19. Deployment from Rule Execution Server console

WB3951.2

Notes:

Administrators and developers use the Rule Execution Server console to monitor rule execution in Rule Execution Server. As part of their administrative tasks, they can deploy RuleApps. Less technical business users do not use Rule Execution Server console.



Deployment with Ant scripts

- WebSphere Operational Decision Management provides Ant tasks that you can use in scripts to automate your RuleApp management, in particular RuleApp deployment
- With these Ant tasks, you can:
 - Create RuleApps
 - Download a RuleApp archive
 - Deploy or remove RuleApp archives
 - Store RuleApp archives
 - Remove RuleApp archives

© Copyright IBM Corporation 2016

Figure 11-20. Deployment with Ant scripts

WB3951.2

Notes:

Operational Decision Manager provides Ant scripts that you can use to automate your RuleApp management, in particular RuleApp deployment.

- `res-jar`: To create RuleApps
- `res-fetch`: To download a RuleApp archive
- `res-fetch-all`: To download all your RuleApp archives
- `res-deploy`: To deploy RuleApp archives
- `res-undeploy`: To remove a RuleApp archive
- `res-write-file`: To store RuleApp archives as files
- `res-write-db`: To write a RuleApp to a database
- `res-delete-file`: To remove RuleApp archives that are stored as files
- `res-delete-db`: To remove RuleApp archives from a database

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

© Copyright IBM Corporation 2016

Figure 11-21. Deployed RuleApp and ruleset names

WB3951.2

Notes:

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*.



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>



11.3. Managing XOMs

Managing XOMs



© Copyright IBM Corporation 2016

Course materials may not be reproduced in whole or in part without the prior written permission of IBM.

10.1

Figure 11-22. Managing XOMs

WB3951.2

Notes:

In this topic, you learn how to manage XOMs with Rule Execution Server.

XOM deployment with decision services

- When you work with decision services, XOM management ensures that any RuleApp that you deploy to Rule Execution Server references the correct XOM
- With XOM management, when you deploy a RuleApp to Rule Execution Server, Rule Designer goes through the following 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 that is contained in the RuleApp. Specifically, Rule Designer copies the URI to a property of the ruleset called `ruleset.managedxom.uris`. For more information, see Managed Java XOM at https://www.ibm.com/support/knowledgecenter/SSQP76_8.8.0/com.ibm.odm.dserver.rules.deploying/topics/tpc_res_managed_jxom_intro.html
- With this mechanism, each ruleset in Rule Execution Server references the correct XOM

© Copyright IBM Corporation 2016

Figure 11-23. XOM deployment with decision services

WB3951.2

Notes:



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

© Copyright IBM Corporation 2016

Figure 11-24. Modifying the XOM

WB3951.2

Notes:



Managed Java XOM elements

- You can manage the Java XOM resources and *libraries* that are attached to a ruleset
- A Java XOM *resource* is a `.jar` file or a `.zip` file
 - You identify such a resource with a unique URI based on the `resuri` protocol
 - Example: `resuri://common-classes.jar/1.0`
- A Java XOM *library* is a set of resources or libraries, or both
 - You identify such a library with a unique URI based on the `reslib` protocol
 - Example: `reslib://loan-xom/1.4`

© Copyright IBM Corporation 2016

Figure 11-25. Managed Java XOM elements

WB3951.2

Notes:

The Java XOM resources are either:

- JAR files: Java archive files, with the `.jar` extension
- ZIP files: Compressed files that contain Java classes and the files that they use

The `resuri` URI is created with the name of the file and a version, which is incremented each time the file checksum changes.



Link from a ruleset to a managed Java XOM element

- The `ruleset.managedxom.uris` ruleset property defines the link between a specific ruleset and a specific managed Java XOM element
- The value of this ruleset property is a list of URIs to `.jar` files, `.zip` files, or libraries
- A specific ruleset is linked to a specific managed Java XOM element if:
 - This ruleset defines the `ruleset.managedxom.uris` ruleset property
 - The value of this ruleset property contains the URI to this Java XOM element

© Copyright IBM Corporation 2016

Figure 11-26. Link from a ruleset to a managed Java XOM element

WB3951.2

Notes:

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

© Copyright IBM Corporation 2016

Figure 11-27. Java XOM deployment

WB3951.2

Notes:

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 in Decision Center

- RuleApps that you deploy to Rule Execution Server from Decision Center reference the correct XOM in one of two ways:
 - In Rule Designer, select the **Enable XOM management in Decision Center** option. This option is in the properties of the main decision service project, under **Decision Service**.
 - When Rule Designer builds the XOM from the source files, it places the XOM binary under `resources/xom-libraries`.
 - You can then publish the XOM binary to Decision Center.
- If XOM management is not enabled:
 - When Rule Designer requests the URI of the XOM from Rule Execution Server, it copies the URI to a `deployment.xml` file in the resources of the decision service.
 - You can then publish the file to Decision Center.
 - With the XOM binary or `deployment.xml` file synchronized between Rule Designer and Decision Center, RuleApps deployed from either environment reference the correct XOM.

© Copyright IBM Corporation 2016

Figure 11-28. XOM deployment in Decision Center

WB3951.2

Notes:



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

© Copyright IBM Corporation 2016

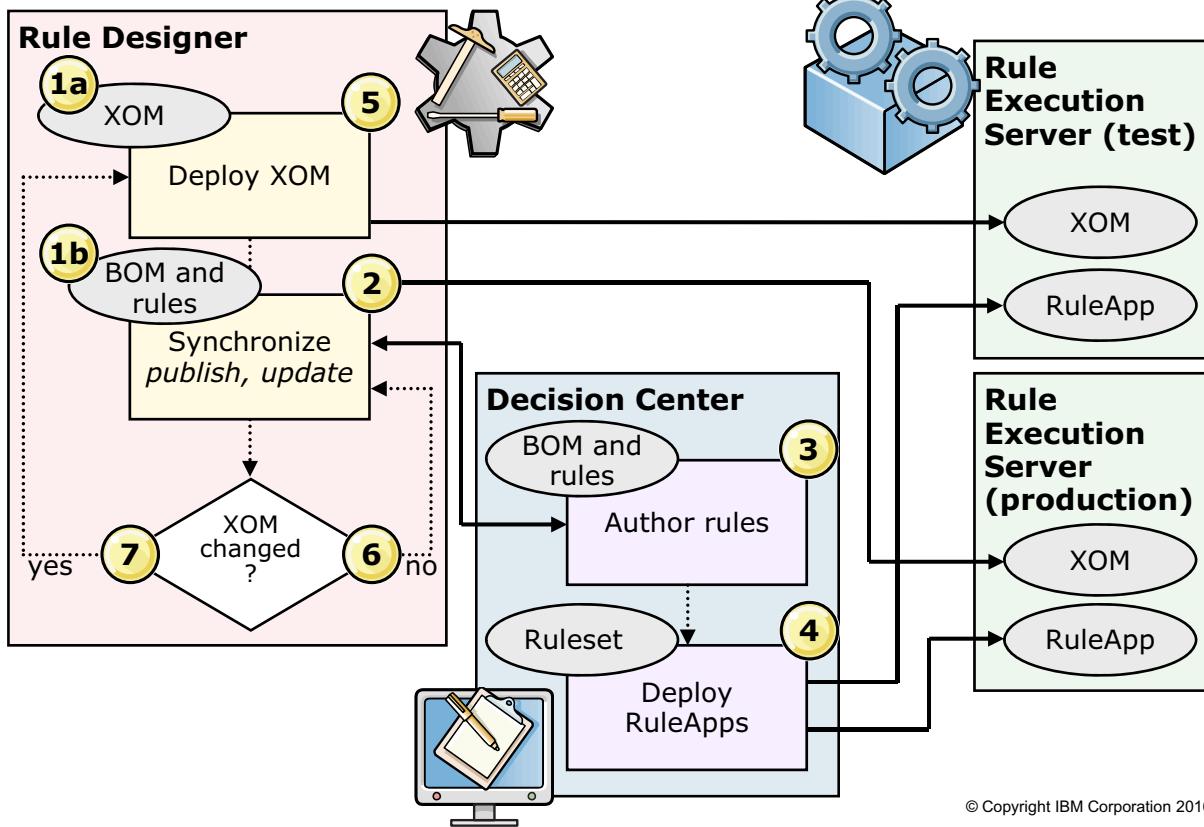
Figure 11-29. Managed XOM and Decision Center

WB3951.2

Notes:

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.

Managed XOM, Rule Designer, and Decision Center



© Copyright IBM Corporation 2016

Figure 11-30. Managed XOM, Rule Designer, and Decision Center

WB3951.2

Notes:

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. Then, 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. To test or execute the rulesets, deploy the decision service from Rule Designer or Decision Center by using the deployment configuration.
5. By default, the XOM is deployed with the decision service to the Rule Execution Servers that are used for tests or for production.

6. If the XOM does not evolve, then it is possible to synchronize Decision Center and Rule Designer environments at any time, thus ensuring that all actors of the rule application are using the same artifacts.
7. 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.



Managed XOM storage

- You can store the managed Java XOM in the persistence layer of Rule Execution Server independently of the ruleset
 - For example, in a test environment, you can store ruleset archives locally if they are too large to work with from a database

© Copyright IBM Corporation 2016

Figure 11-31. Managed XOM storage

WB3951.2

Notes:

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



Unit summary

Having completed this unit, you should be able to:

- Explain the basic features of Rule Execution Server
- Prepare deployment configurations
- Deploy RuleApps to Rule Execution Server
- Describe the basics of XOM management

© Copyright IBM Corporation 2016

Figure 11-32. Unit summary

WB3951.2

Notes:



Checkpoint questions

- 1. True or False:** Rulesets must be packaged as RuleApps for deployment.
- 2. True or False:** Deployment is an administrative task that cannot be performed in Decision Center.
- 3. What tools can you use to deploy rules to Rule Execution Server?**
 - a. Rule Designer
 - b. Decision Center Business console
 - c. Rule Execution Server console
 - d. Ant tasks
 - e. All of the above
- 4. True or False:** Deployment of decision services is managed with a deployment configuration.

© Copyright IBM Corporation 2016

Figure 11-33. Checkpoint questions

WB3951.2

Notes:

Write your answers here:

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



Checkpoint answers

1. True.

2. False: Business users can deploy from Decision Center.

3. What can you use to deploy rules to Rule Execution Server?

- e. All of a, b, c, and d

4. True.

© Copyright IBM Corporation 2016

Figure 11-34. Checkpoint answers

WB3951.2

Notes:

Exercise 15



Managing deployment

© Copyright IBM Corporation 2016

Course materials may not be reproduced in whole or in part without the prior written permission of IBM.

10.1

Figure 11-35. Exercise 15

WB3951.2

Notes:

Exercise objectives

After completing this exercise, you should be able to:

- Define a RuleApp and ruleset properties
- Use deployment configurations to deploy decision services
- Deploy the XOM for its management in Rule Execution Server

© Copyright IBM Corporation 2016

Figure 11-36. Exercise objectives

WB3951.2

Notes:

Unit 12. Executing rules with Rule Execution Server

What this unit is about

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.

What you should be able to do

After completing this unit, you should be able to:

- 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

How you will check your progress

- Checkpoint
- Exercises



Unit objectives

After completing this unit, you should be able to:

- 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 2016

Figure 12-1. Unit objectives

WB3951.2

Notes:



Topics

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

© Copyright IBM Corporation 2016

Figure 12-2. Topics

WB3951.2

Notes:

12.1.Managed execution with Rule Execution Server

Managed execution with Rule Execution Server



© Copyright IBM Corporation 2016

Course materials may not be reproduced in whole or in part without the prior written permission of IBM.

10.1

Figure 12-3. Managed execution with Rule Execution Server

WB3951.2

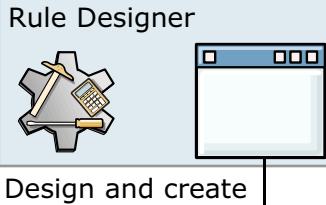
Notes:

Rule Execution Server in a production enterprise application

Design and create RuleApps



Deploy



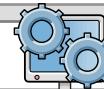
Design and create client application

Main node

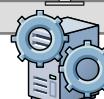
Failover node

Application server cluster

RES*
console



Execution
Unit (XU)



RuleApp
storage



Decision
Warehouse



RES*



HTDS**



Client
application



Application servers

XU

RuleApp
storage

Decision
Warehouse

RES*

HTDS**

Client
application

* RES = Rule Execution Server

** HTDS = Hosted transparent decision service

© Copyright IBM Corporation 2016

Figure 12-4. Rule Execution Server in a production enterprise application

WB3951.2

Notes:

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

© Copyright IBM Corporation 2016

Figure 12-5. Introducing managed execution with Rule Execution Server

WB3951.2

Notes:



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

© Copyright IBM Corporation 2016

Figure 12-6. Example of possible platforms

WB3951.2

Notes:

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.

12.2.Rule Execution Server modular architecture

Rule Execution Server modular architecture



© Copyright IBM Corporation 2016

Course materials may not be reproduced in whole or in part without the prior written permission of IBM.

10.1

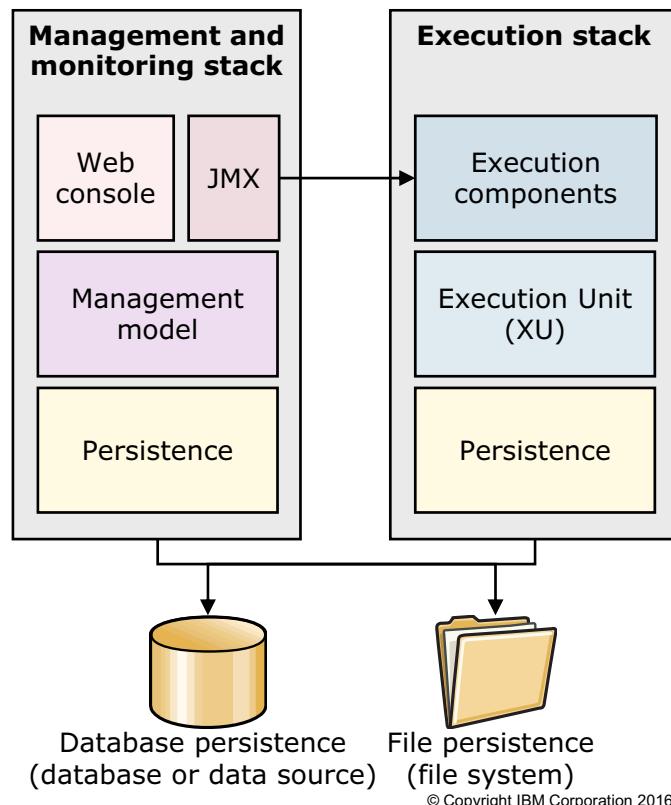
Figure 12-7. Rule Execution Server modular architecture

WB3951.2

Notes:

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



© Copyright IBM Corporation 2016

Figure 12-8. Rule Execution Server architecture

WB3951.2

Notes:

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.

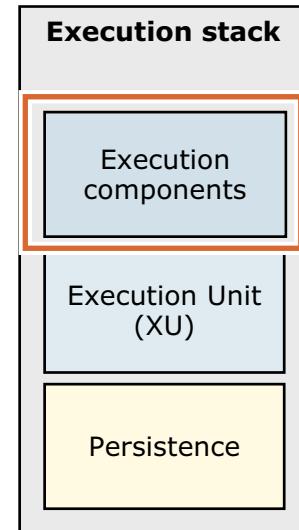
Although how to do this deployment is not covered in this course, you can find some information about this topic later in this unit.

- 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)



© Copyright IBM Corporation 2016

Figure 12-9. Execution stack: Execution components

WB3951.2

Notes:

The Execution stack comprises:

- Java execution components (Java SE and Java EE)
- JMX Execution Model components
- Execution Units (XU, see next slide)

Java SE execution components and Java EE execution components authorize the execution of a ruleset by the *Execution Unit* (XU). Java SE execution components comprise:

- Stateless ruleset sessions (further detailed next)
- Stateful ruleset sessions (further detailed next)
- Decision traces
- Ruleset execution interceptors

Java EE execution components comprise the same as Java SE execution components, plus:

- Message driven (asynchronous) rule beans (further detailed next)

- Remote invocation (RMI)

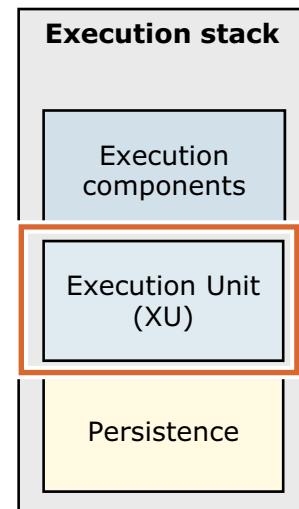


Information

The execution stack is in essence the same for both Java SE and Java EE. This similarity also exists for the management stack.

Execution Unit

- A resource adapter for Java EE Connector Architecture (JCA)
 - Handles the low-level aspects of ruleset execution
 - Collaborates with the server to provide several connector system-level contracts as a service provider interface
- An XU can also run 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
 - Execution trace
 - An XU container to create and pool connections of the XU (JCA)
 - Logging
 - Debugging
 - Notification that a RuleApp was modified
 - Statistics



© Copyright IBM Corporation 2016

Figure 12-10. Execution Unit

WB3951.2

Notes:

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.

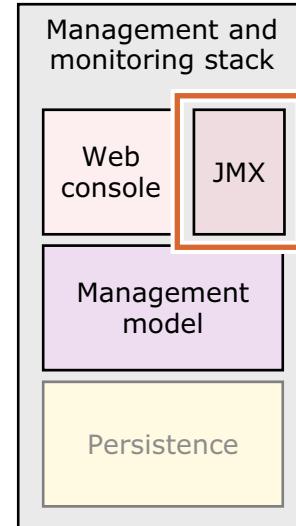
The XU manages rule engines, so the application server or application client uses the XU to connect to the rule engine. The XU retrieves and loads rulesets from persistence layer, passes data between the application and the rule engine, and manages the rule engines that are attached to loaded rulesets. 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 Execution Unit (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.

WebSphere Education

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
 - 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 database
- 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



© Copyright IBM Corporation 2016

Figure 12-11. JMX management and execution model

WB3951.2

Notes:

The management and monitoring stack comprises:

- Management console
- JMX Management Model components

Rule Execution Server uses the Java Management Extension (JMX) API as its underlying architecture.

The JMX API uses Java objects that are called MBeans to model system administration functions. Each MBean contains a set of attributes that define parameters for various management functions and operations that define administrative actions.

The management and monitoring model components constitute an interface that depends on your application server and that manages business logic, including remote updating and browsing. They are based on JMX, a part of Java SE.

The JMX components are responsible for:

- Hot update and deployment. When a new version of a ruleset is deployed, the JMX layer informs all XUs in the cluster that a new version of the ruleset is available in the database.

- Displaying execution statistics as MBeans Execution statistics so they can then be monitored by using JMX tools like Tivoli.

From the Rule Execution Server console, you can access these MBean attributes and operations through a convenient graphical user interface.

Management and monitoring stack: Console

- Web-based administration interface
- Application-specific interface to manage business logic, including remote browsing, updating, and deployment of RuleApps
- Central point of the Rule Execution Server architecture

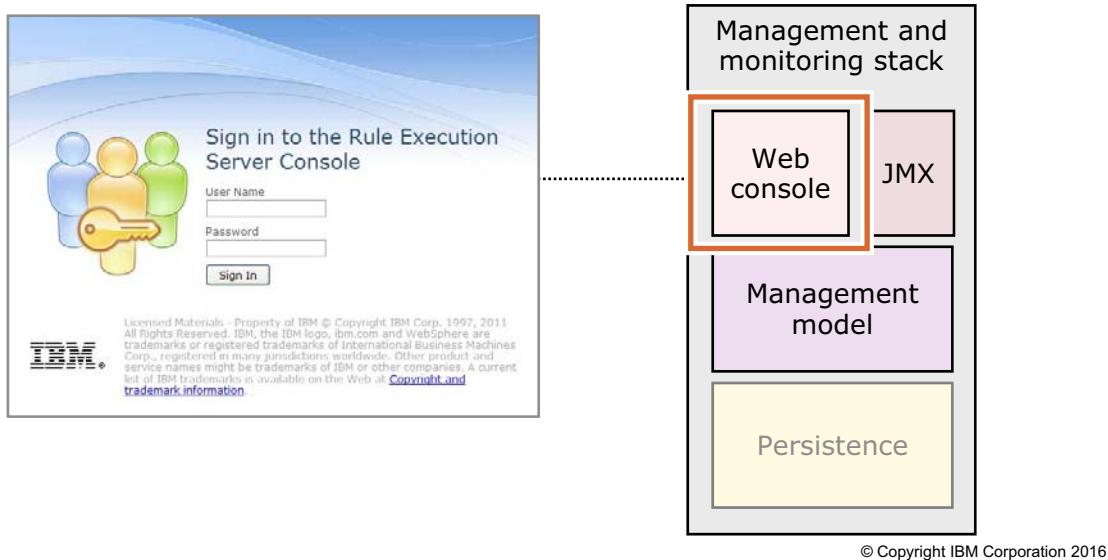


Figure 12-12. Management and monitoring stack: Console

WB3951.2

Notes:

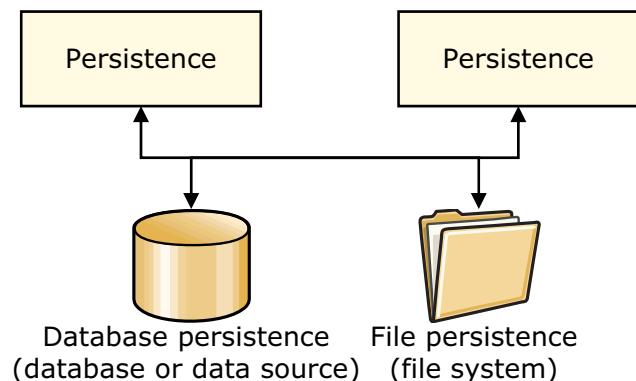
Rule Execution Server is accessed through the Rule Execution Server console, which is a web-based administration interface. The Rule Execution Server console is a graphical user interface that depends on your application server. Use it to manage business logic, including remote browsing, updating, and deployment of RuleApps.

The Rule Execution Server console is the central point of the Rule Execution Server architecture. Many features of Rule Execution Server do not work without the Rule Execution Server console.

Persistence layer

- Packaged in the management stack and the execution stack
 - Both the management and execution stacks can access the ruleset storage
 - When a new version of a ruleset is added to the management model, a notification of the update is sent through JMX to the XU
 - The XU receives the new resource
- The persistence layer provides a solution for file persistence (file system) or for database persistence (JDBC or data source)

- Possible values for persistence type:
 - `file` for a file persistence in Java SE
 - `datasource` for a database persistence in Java EE
 - `jdbc` for a database persistence in Java SE



© Copyright IBM Corporation 2016

Figure 12-13. Persistence layer

WB3951.2

Notes:

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.



Warning

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)

© Copyright IBM Corporation 2016

Figure 12-14. Ant tasks

WB3951.2

Notes:

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.

12.3.Managed execution in action

Managed execution in action



© Copyright IBM Corporation 2016

Course materials may not be reproduced in whole or in part without the prior written permission of IBM.

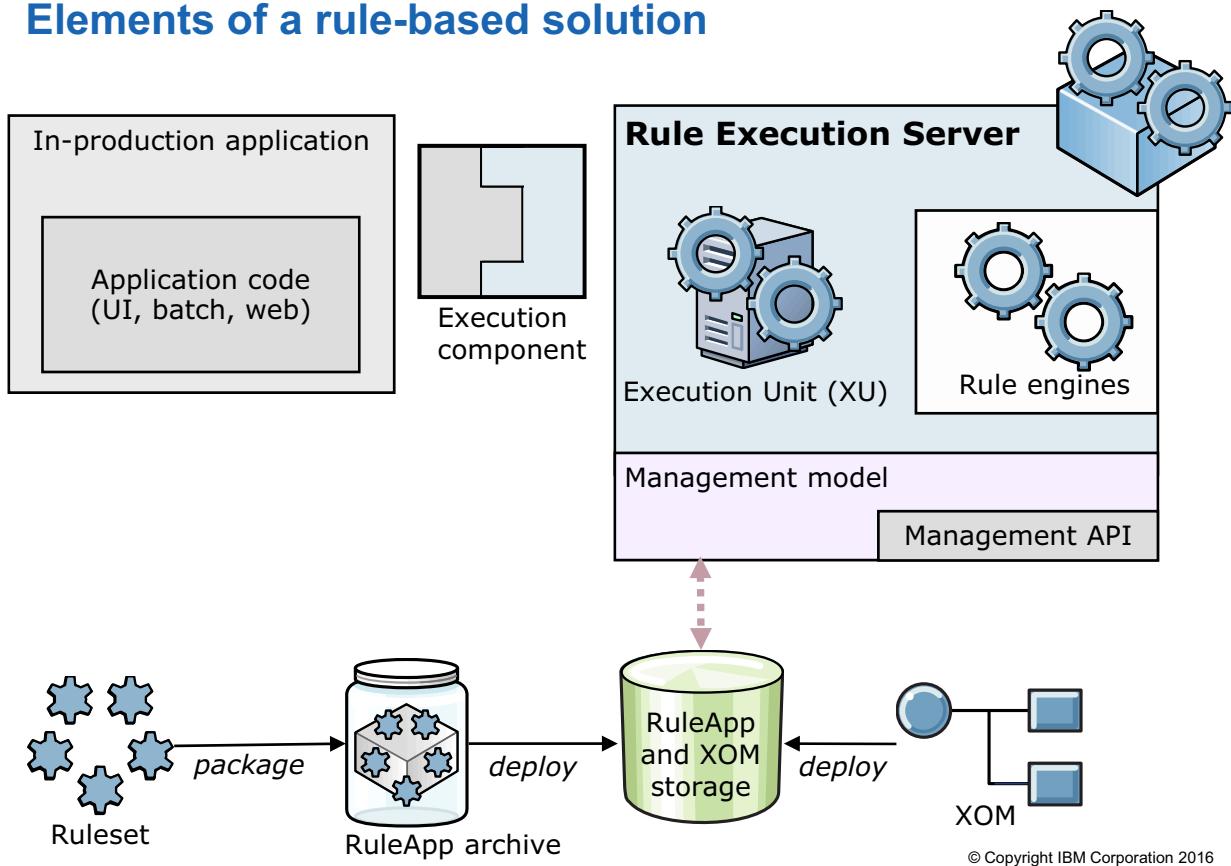
10.1

Figure 12-15. Managed execution in action

WB3951.2

Notes:

Elements of a rule-based solution



© Copyright IBM Corporation 2016

Figure 12-16. Elements of a rule-based solution

WB3951.2

Notes:

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

© Copyright IBM Corporation 2016

Figure 12-17. Ruleset parsing

WB3951.2

Notes:

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

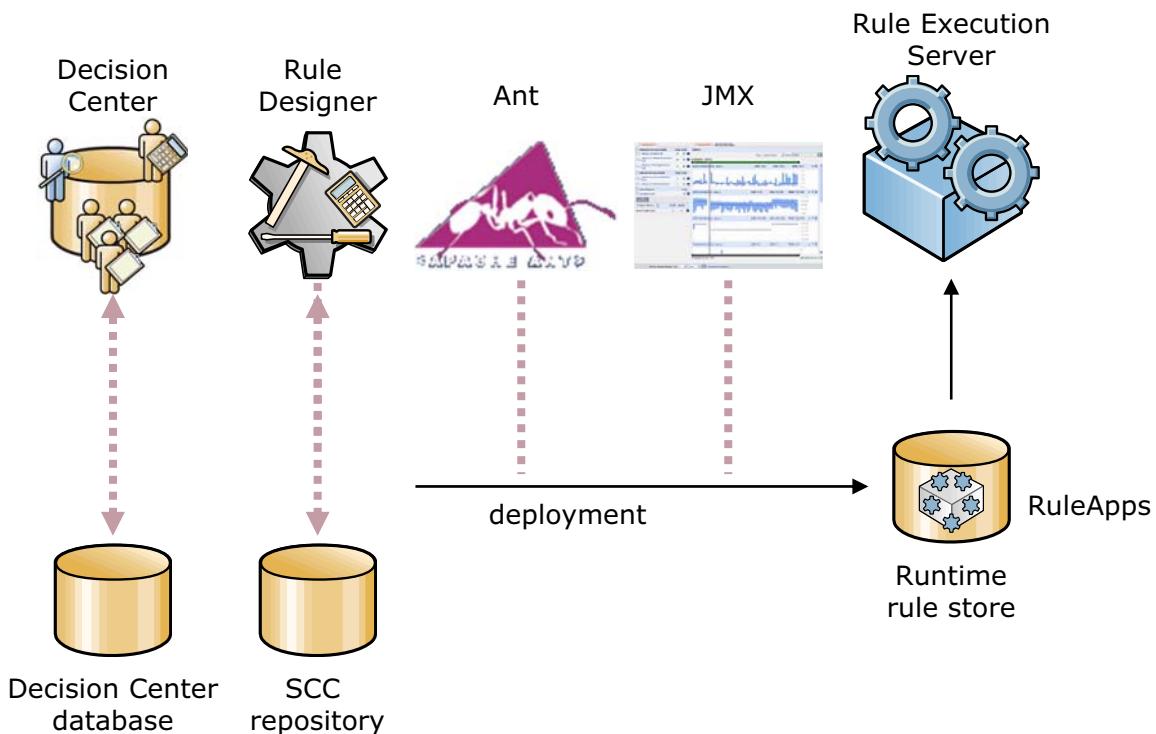
© Copyright IBM Corporation 2016

Figure 12-18. Ruleset execution

WB3951.2

Notes:

Ruleset update at run time



© Copyright IBM Corporation 2016

Figure 12-19. Ruleset update at run time

WB3951.2

Notes:

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

© Copyright IBM Corporation 2016

Figure 12-20. Runtime class loading

WB3951.2

Notes:

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.

12.4. Platforms for Rule Execution Server

Platforms for Rule Execution Server



© Copyright IBM Corporation 2016

Course materials may not be reproduced in whole or in part without the prior written permission of IBM.

10.1

Figure 12-21. Platforms for Rule Execution Server

WB3951.2

Notes:



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

© Copyright IBM Corporation 2016

Figure 12-22. Architecture questions

WB3951.2

Notes:

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

© Copyright IBM Corporation 2016

Figure 12-23. Possible choices: Introduction

WB3951.2

Notes:

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 and MTDS later in this unit.
- If your business rule application is not SOA-based, and you do not use your rules as a service, you can base your business rule application on *Java*.
- To deploy Rule Execution Server on a supported application server (for example WebSphere Application Server), use the appropriate Rule Execution Server *Java EE API* (synchronous or not, local or remote, with or without EJB) in your client application.
- If you want to deploy on a non-supported application server or in a web server (for example, Apache HTTP Server), use Java SE instead. In that case, use the *Java SE Rule Session API* in your client application.



Java SE

- Rule Execution Server can execute rulesets with 100% Java SE code
- It is used to:
 - Run rules from a web server like Tomcat
 - Run batches or run rules from a JMS provider or an enterprise service bus (ESB) that is not Java EE compliant
- When deployed in Java SE, your client application locally accesses the Rule Execution Server Execution components
- In this scenario, your client application uses the simple Java SE rule session API

© Copyright IBM Corporation 2016

Figure 12-24. Java SE

WB3951.2

Notes:

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)

© Copyright IBM Corporation 2016

Figure 12-25. Java EE

WB3951.2

Notes:

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.

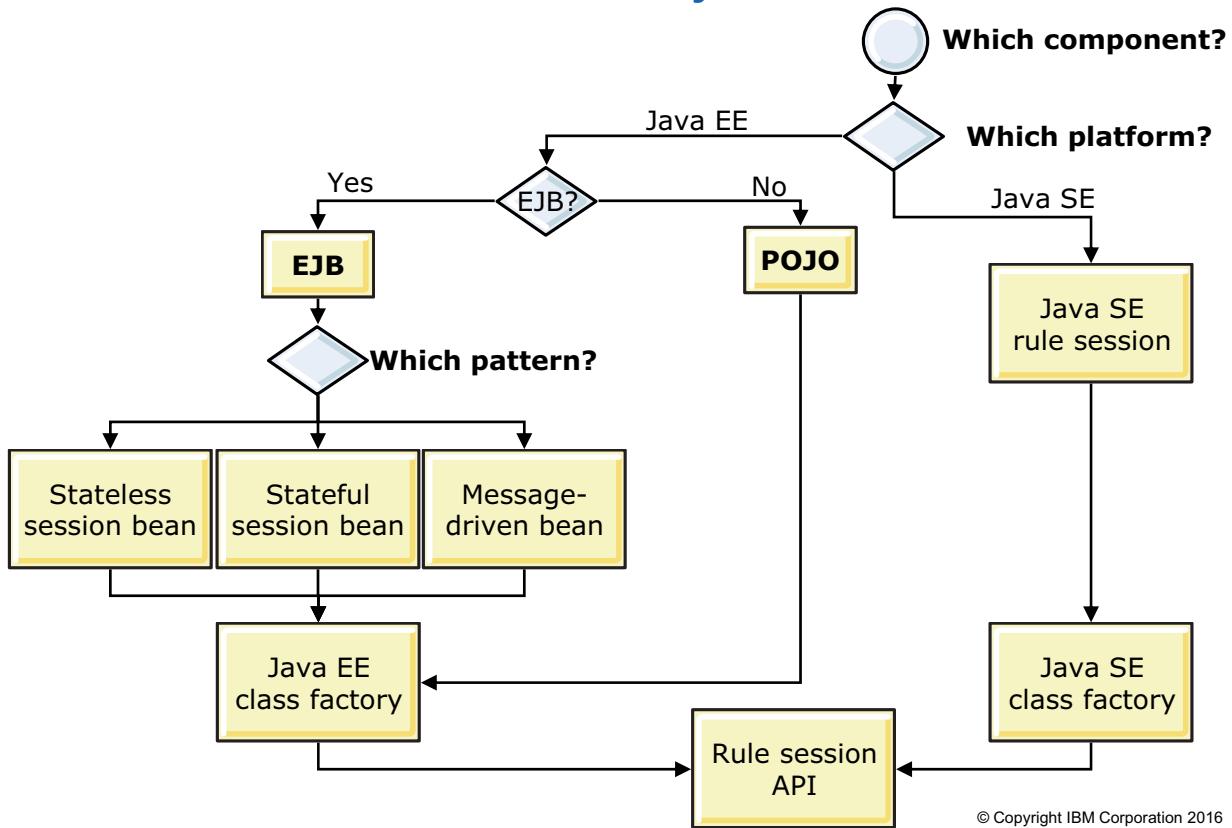
If your business rule application requires services such as transaction management, web containers, and security, then change to a Java EE application server, rather than add the necessary Java extensions to the Java SE platform. When deployed in Java EE, your client application can access the Rule Execution 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. You also learn how to write your client application.

Java SE and Java EE: A summary



© Copyright IBM Corporation 2016

Figure 12-26. Java SE and Java EE: A summary

WB3951.2

Notes:

The possible choices with the Java Platform (Java SE or Java EE) that were presented in the previous slides are summarized here.



Transparent decision services

- You can expose ruleset execution as a service
 - A web service with management capabilities
- Transparent:
 - Users do not have to know how it is implemented
 - Users must know only how to access it through HTTP with the XML or JSON formats
- Two types of transparent decision services:
 - Hosted transparent decision service (HTDS): Use when you want to create a web service with minimal configuration
 - Monitored transparent decision service (MTDS): Use when you want to create a web service that requires specific configuration or customization to meet your enterprise requirements

© Copyright IBM Corporation 2016

Figure 12-27. Transparent decision services

WB3951.2

Notes:

Operational Decision Manager provides ways to automatically expose ruleset execution as a transparent decision service. A transparent decision service is technically a web service with management capabilities. It drives rule execution, and enables users to access Rule Execution Server through a web service, rather than accessing it directly. It is said to be *transparent* because users do not have to know how it is implemented, just how to access it through HTTP with XML or JSON formats.

In Operational Decision Manager, a transparent decision service is either:

- *Hosted*: A hosted transparent decision service (HTDS) is a simple way for you to create a web service.
- *Monitored*: A monitored transparent decision service (MTDS) is a more complex yet more configurable way for you to create a web service.

The mechanism to create hosted transparent decision services (HTDS) requires only limited configuration. If you want a rapid deployment of a web service that is based on your rules, use that mechanism and create an HTDS. For monitored transparent decision services (MTDS), Operational Decision Manager generates a series of classes that you can then configure to match

your specific requirements. If you have specific needs, deploying your rules as an MTDS might be better than as an HTDS.

You can use HTDS and MTDS 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.

Similarly, if you are using a dynamic (XML-based) XOM, the XSD files that define the XML model are packaged within the ruleset by default (which is the default configuration in Rule Designer). If you modify this default behavior, you must package the XSD files that define the dynamic domain in the EAR file as well.

You learn more about transparent decision services later in this unit.

12.5.Introducing the Rule Execution Server API

Introducing the Rule Execution Server API



© Copyright IBM Corporation 2016

Course materials may not be reproduced in whole or in part without the prior written permission of IBM.

10.1

Figure 12-28. Introducing the Rule Execution Server API

WB3951.2

Notes:



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

© Copyright IBM Corporation 2016

Figure 12-29. Introduction

WB3951.2

Notes:

For Java environments, Rule Execution Server does not provide a ready-to-use client application. You must write the client application code to execute your ruleset, by using the *Rule Execution Server API* to command the Execution Unit (XU). This API is provided to bind an application and call the execution module.

Which part of the API to use depends on the execution pattern that your client application follows, which, in turn, depends on how Rule Execution Server is deployed in your enterprise. In this topic, you discover first the *execution patterns*, and then the API that these patterns are based on.



Execution patterns: Synchronous

In Java SE:

- Local access
 - Java SE rule session

In a Java EE container:

- Local access with or without transaction control
 - Plain old Java objects (POJO) rule session
- Local access with transaction control at the rule session level
 - Local interface of a stateful or a stateless EJB rule session
- Remote access with transaction control at the rule session level
 - Remote interface of a stateful or a stateless EJB rule session

© Copyright IBM Corporation 2016

Figure 12-30. Execution patterns: Synchronous

WB3951.2

Notes:

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. The way that you implement your client application and the way Rule Execution Server components are deployed are immediate consequences of this choice. 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)

© Copyright IBM Corporation 2016

Figure 12-31. Execution patterns: Asynchronous

WB3951.2

Notes:

You can do asynchronous access, either remote or local, with either:

- A message-driven rule bean (MDB)
- A stateless asynchronous execution, in Java SE and in Java EE POJO modes only



Rule session API overview (1 of 2)

- When using rule sessions to request ruleset execution within the managed environment of Rule Execution Server, your client application completes steps similar to the next ones:
 1. Get a rule session factory
 2. Create a rule session from the rule session factory
 3. Create a rule session request from the rule session
 4. Configure the rule session request
 5. Execute the rule session request from the rule session
 6. Retrieve the rule session response, and analyze it

© Copyright IBM Corporation 2016

Figure 12-32. Rule session API overview (1 of 2)

WB3951.2

Notes:



Rule session API overview (2 of 2)

- The exact classes that your client application requires to implement this series of steps depend on the chosen execution pattern
- These classes all rely on similar Java interfaces
 - See next slides

© Copyright IBM Corporation 2016

Figure 12-33. Rule session API overview (2 of 2)

WB3951.2

Notes:

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

© Copyright IBM Corporation 2016

Figure 12-34. Rule session factories

WB3951.2

Notes:

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

© Copyright IBM Corporation 2016

Figure 12-35. Rule sessions

WB3951.2

Notes:

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

© Copyright IBM Corporation 2016

Figure 12-36. Stateless rule sessions

WB3951.2

Notes:

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

© Copyright IBM Corporation 2016

Figure 12-37. Stateful rule sessions

WB3951.2

Notes:

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

© Copyright IBM Corporation 2016

Figure 12-38. Rule session requests

WB3951.2

Notes:

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 ruleset 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

© Copyright IBM Corporation 2016

Figure 12-39. Rule session requests: Decision ID

WB3951.2

Notes:

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

© Copyright IBM Corporation 2016

Figure 12-40. Rule session responses

WB3951.2

Notes:

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 ruleset 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

© Copyright IBM Corporation 2016

Figure 12-41. Ruleset path for execution (1 of 2)

WB3951.2

Notes:

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, as you learned in Unit 11, "Managing deployment". 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

© Copyright IBM Corporation 2016

Figure 12-42. Ruleset path for execution (2 of 2)

WB3951.2

Notes:

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 lets 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

© Copyright IBM Corporation 2016

Figure 12-43. Traces

WB3951.2

Notes:

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 `IlrSessionResponse.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

© Copyright IBM Corporation 2016

Figure 12-44. Decision traces: How to

WB3951.2

Notes:

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`

© Copyright IBM Corporation 2016

Figure 12-45. Decision traces: Complement

WB3951.2

Notes:

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.

You learned how to define a ruleset property in Unit 11, "Managing deployment".

12.6. Executing rules in Java SE

Executing rules in Java SE



© Copyright IBM Corporation 2016

Course materials may not be reproduced in whole or in part without the prior written permission of IBM.

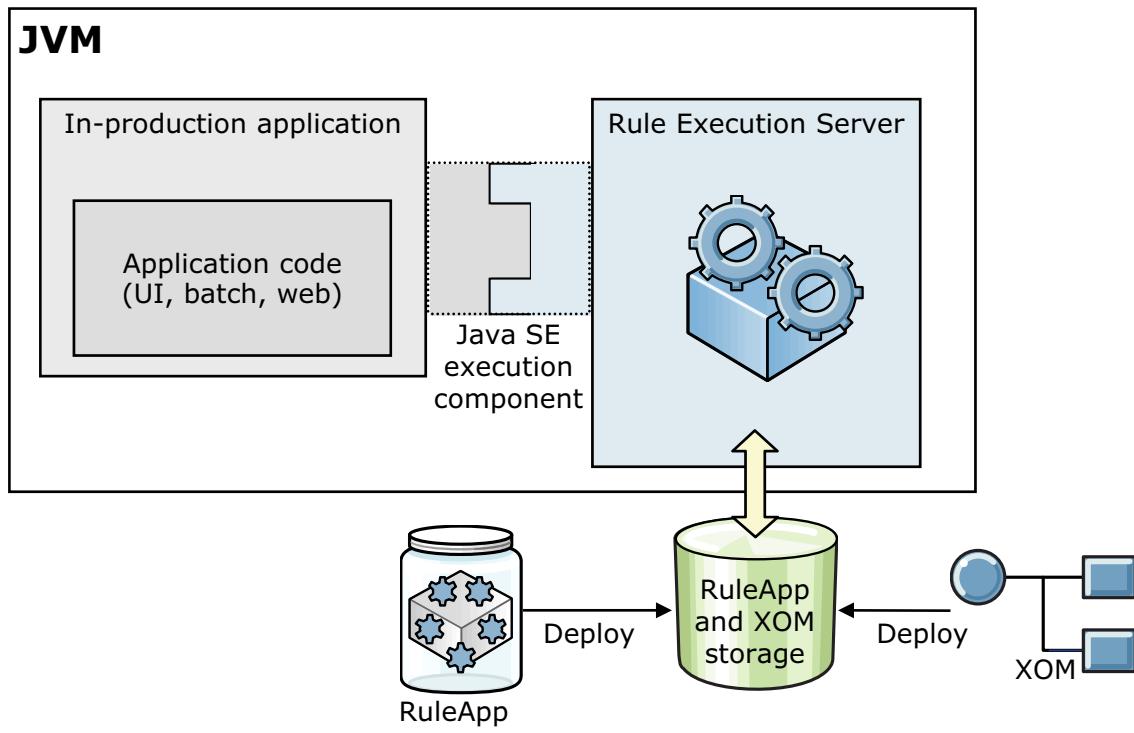
10.1

Figure 12-46. Executing rules in Java SE

WB3951.2

Notes:

Rule execution in Java SE



© Copyright IBM Corporation 2016

Figure 12-47. Rule execution in Java SE

WB3951.2

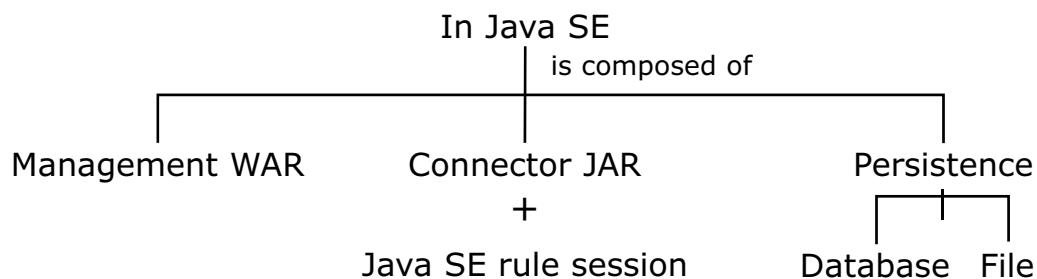
Notes:

When your client application asks for the managed execution of your ruleset with Rule Execution Server 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 Execution Unit (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



© Copyright IBM Corporation 2016

Figure 12-48. Deployed Rule Execution Server artifacts in Java SE

WB3951.2

Notes:

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

© Copyright IBM Corporation 2016

Figure 12-49. Required Rule Execution Server API in Java SE

WB3951.2

Notes:

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. You discover this wizard in the exercise that is associated with this unit, and further explore the Rule Execution Server API later in this unit.

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");
```

© Copyright IBM Corporation 2016

Figure 12-50. Example with a stateless rule session in Java SE

WB3951.2

Notes:

This example shows a possible skeleton for a client application that uses a stateless rule session in Java SE.

In this example, the ruleset has one IN OUT ruleset parameter that is called `report`, of class `Report`.

- Get a *rule session factory*:

```
IlrSessionFactory factory = new IlrJ2SESessionFactory();
```

- Create a *rule session* from the rule session factory:

```
IlrStatelessSession session = factory.createStatelessSession();
```

- Create a *rule session request* from the rule session:

```
IlrSessionRequest sessionRequest = factory.createRequest();
```

-
4. Configure the rule session request; in particular, set the initial value of the report ruleset parameter (IN):

```
sessionRequest.setRulesetPath(...);  
sessionRequest.setTraceEnabled(true);  
sessionRequest.getTraceFilter().  
setInfoAllFilters(true);  
Map inputParameters = new HashMap();  
Report in_report = new Report(...);  
inputParameters.put("report", in_report);  
sessionRequest.setInputParameters(inputParameters);
```

5. Execute the rule session request from the rule session and retrieve the *rule session responses*:

```
IlrSessionResponse sessionResponse = session.execute(sessionRequest);
```

6. Analyze the *rule session responses*; in particular, get the modified value of the report ruleset parameter (OUT):

```
Report out_report = (Report)  
sessionResponse.getOutputParameters().get("report");
```


12.7. Executing rules in Java EE

Executing rules in Java EE



© Copyright IBM Corporation 2016

Course materials may not be reproduced in whole or in part without the prior written permission of IBM.

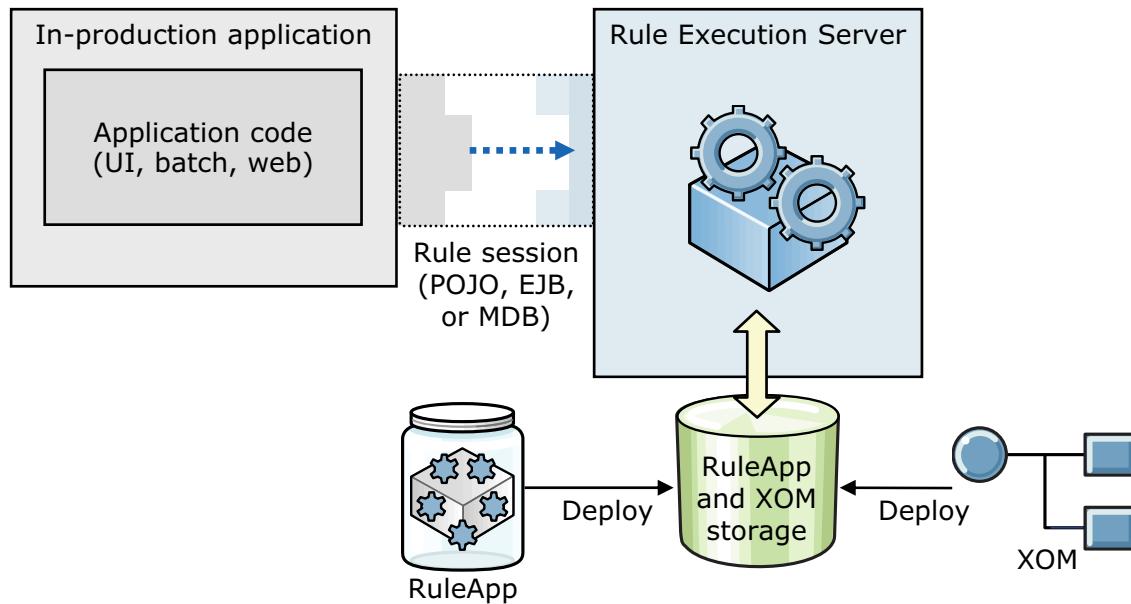
10.1

Figure 12-51. Executing rules in Java EE

WB3951.2

Notes:

Rule execution in Java EE



© Copyright IBM Corporation 2016

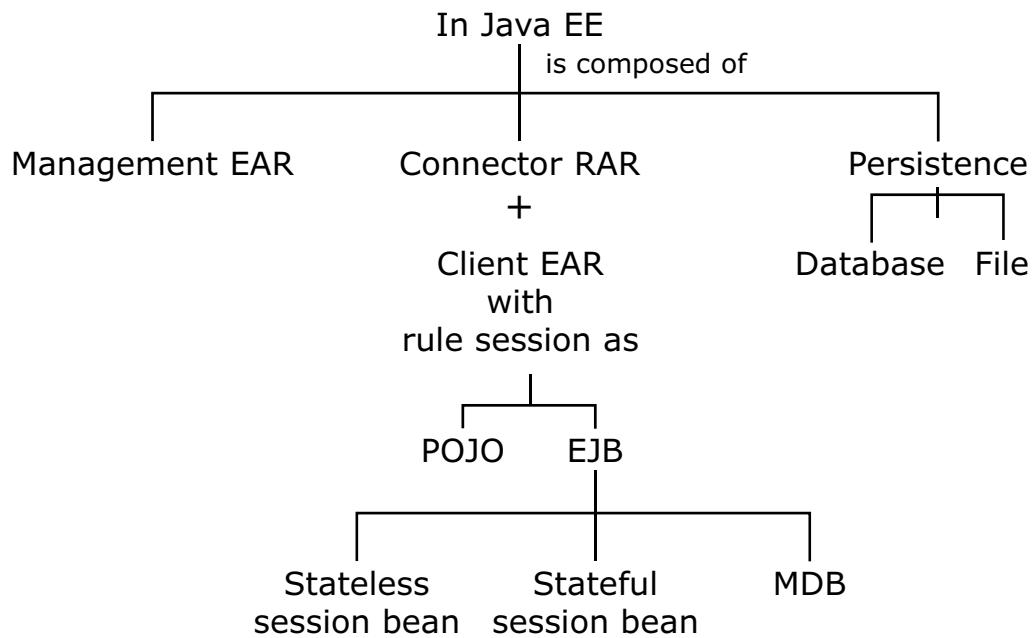
Figure 12-52. Rule execution in Java EE

WB3951.2

Notes:

When your client application asks for the managed execution of your ruleset with Rule Execution Server deployed in Java EE, the application and Rule Execution Server might run on different JVMs.

Deployed Rule Execution Server artifacts in Java EE



© Copyright IBM Corporation 2016

Figure 12-53. Deployed Rule Execution Server artifacts in Java EE

WB3951.2

Notes:

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

© Copyright IBM Corporation 2016

Figure 12-54. Required API in Java EE

WB3951.2

Notes:

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`)

© Copyright IBM Corporation 2016

Figure 12-55. Java EE POJO rule session

WB3951.2

Notes:

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`)

© Copyright IBM Corporation 2016

Figure 12-56. Java EE EJB rule session

WB3951.2

Notes:

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

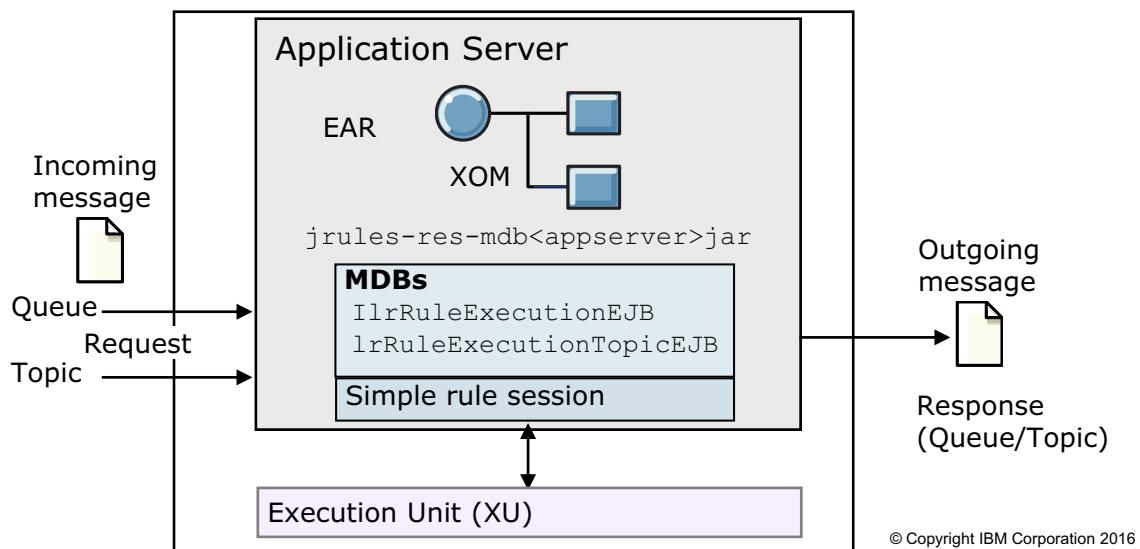


Figure 12-57. Java EE message-driven bean (MDB)

WB3951.2

Notes:

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 ruleset 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.

12.8. Executing rules as transparent decision services

Executing rules as transparent decision services



© Copyright IBM Corporation 2016

Course materials may not be reproduced in whole or in part without the prior written permission of IBM.

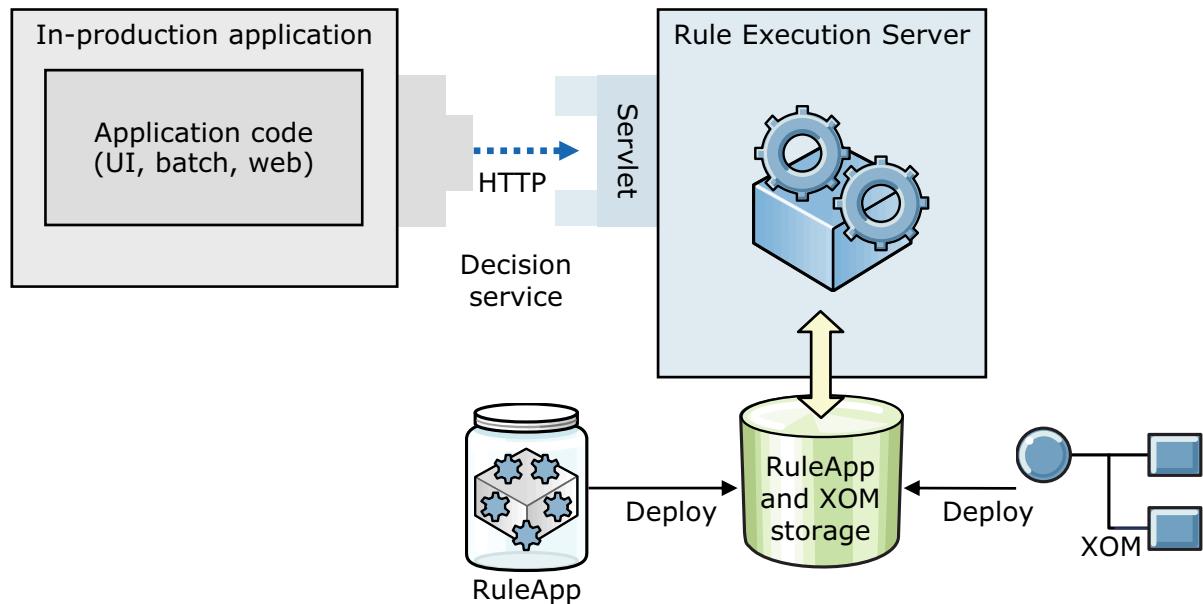
10.1

Figure 12-58. Executing rules as transparent decision services

WB3951.2

Notes:

Rules as transparent decision services



© Copyright IBM Corporation 2016

Figure 12-59. Rules as transparent decision services

WB3951.2

Notes:

By using hosted or monitored transparent decision services, you can access Rule Execution Server through a web service.

A transparent decision service helps development teams deploy business rules as fully formed web services and weave them into service-oriented architecture (SOA) platforms.

To execute rules as decision services, the application is installed remotely. Requests are made to Rule Execution Server through HTTP by using transparent decision services.

Operational Decision Manager provides two types of transparent decision services:

- Hosted transparent decision service (HTDS): An HTDS is essentially a ruleset that is deployed as a web service, which is installed on the same application server as Rule Execution Server.
- Monitored transparent decision service (MTDS): An MTDS is a service that is generated in Rule Designer and installed on the same application server as Rule Execution Server.



Hosted transparent decision service

- Is essentially a ruleset that is deployed as a web service, which is installed on the same application server as Rule Execution Server
- Is an execution component linked to a ruleset path with a JMX management bean (MBean)
- Rule Execution Server automatically exposes deployed rulesets as a web (decision) service
- Hosted transparent decision services can run synchronously or asynchronously
- To create an HTDS:
 - Install Rule Execution Server on your web or application server
 - Deploy the HTDS ruleset archive to the same server

© Copyright IBM Corporation 2016

Figure 12-60. Hosted transparent decision service

WB3951.2

Notes:

A hosted transparent decision service is an execution component that is linked to a ruleset path with a JMX management bean (MBean). To use a hosted transparent decision service, install Rule Execution Server on your web server or application server, and deploy the hosted transparent decision service archive to the same server.

Rule Execution Server automatically exposes any deployed rulesets as a web service, regardless of the underlying type of XOM (Java or dynamic). A *Web Services Description Language (WSDL)* file is generated for each deployed ruleset archive. These rulesets can be exposed as a web service without any code deployment.



Monitored transparent decision service

- Generated with Rule Designer and installed on the same application server as Rule Execution Server
- Exposes management and runtime information about the web service to Rule Execution Server
- To create an MTDS:
 - Create it as a project, by using the Rule Designer Client Project for RuleApps wizard
 - Install it on the web or application server

© Copyright IBM Corporation 2016

Figure 12-61. Monitored transparent decision service

WB3951.2

Notes:

A monitored transparent decision service exposes management and runtime information about the web service to Rule Execution Server. You can create a monitored transparent decision in Rule Designer and then install it on the web or application server.

Monitored transparent decision services can manage rulesets that use any type of XOM. However, with XML, notice that the XML ruleset parameters are handled as String, not as XML objects.



Unit summary

Having completed this unit, you should be able to:

- 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 2016

Figure 12-62. Unit summary

WB3951.2

Notes:

Checkpoint questions (1 of 2)

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. Which of the following services is a valid type of transparent decision service?
 - a. Hosted transparent decision service (HTDS)
 - b. Monitored transparent decision service (MTDS)
 - c. Web transparent decision service (WTDS)

© Copyright IBM Corporation 2016

Figure 12-63. Checkpoint questions (1 of 2)

WB3951.2

Notes:

Write your answers here:

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



Checkpoint questions (2 of 2)

5. 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

© Copyright IBM Corporation 2016

Figure 12-64. Checkpoint questions (2 of 2)

WB3951.2

Notes:

Write your answers here:

5.



Checkpoint answers

1. **True.**
2. **True.**
3. **True.**
4. Which of the following services is a valid type of transparent decision service?
 - a. Hosted transparent decision service (HTDS)
 - b. Monitored transparent decision service (MTDS)
5. 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

© Copyright IBM Corporation 2016

Figure 12-65. Checkpoint answers

WB3951.2

Notes:

Exercise 16



Exploring the Rule Execution Server
console

© Copyright IBM Corporation 2016

Course materials may not be reproduced in whole or in part without the prior written permission of IBM.

10.1

Figure 12-66. Exercise 16

WB3951.2

Notes:



Exercise objectives

After completing this exercise, you should be able to:

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

© Copyright IBM Corporation 2016

Figure 12-67. Exercise objectives

WB3951.2

Notes:

Exercise 17



Executing rules with Rule Execution Server in Java EE

© Copyright IBM Corporation 2016

Course materials may not be reproduced in whole or in part without the prior written permission of IBM.

10.1

Figure 12-68. Exercise 17

WB3951.2

Notes:



Exercise objectives

After completing this exercise, you should be able to:

- Use the basic Rule Execution Server API to request ruleset execution with Rule Execution Server in Java EE

© Copyright IBM Corporation 2016

Figure 12-69. Exercise objectives

WB3951.2

Notes:

Exercise 18



Executing rules as a hosted transparent decision service (HTDS)

© Copyright IBM Corporation 2016

Course materials may not be reproduced in whole or in part without the prior written permission of IBM.

10.1

Figure 12-70. Exercise 18

WB3951.2

Notes:



Exercise objectives

After completing this exercise, you should be able to:

- Deploy an HTDS that can be used in a service-oriented architecture (SOA)
- Create a client application that uses an HTDS
- Monitor HTDS execution statistics in the Rule Execution Server console

© Copyright IBM Corporation 2016

Figure 12-71. Exercise objectives

WB3951.2

Notes:

Unit 13. Auditing and monitoring ruleset execution

What this unit is about

In this unit, you learn how to audit and monitor ruleset execution with Decision Warehouse.

What you should be able to do

After completing this unit, you should be able to:

- Audit the execution of rulesets with Decision Warehouse
- Monitor ruleset execution with the Rule Execution Server console

How you will check your progress

- Checkpoint
- Exercise



Unit objectives

After completing this unit, you should be able to:

- Audit the execution of rulesets with Decision Warehouse
- Monitor ruleset execution with the Rule Execution Server console

© Copyright IBM Corporation 2016

Figure 13-1. Unit objectives

WB3951.2

Notes:



Topics

- Auditing ruleset execution
- Monitoring ruleset execution

© Copyright IBM Corporation 2016

Figure 13-2. Topics

WB3951.2

Notes:

13.1.Auditing ruleset execution

Auditing ruleset execution



© Copyright IBM Corporation 2016

Course materials may not be reproduced in whole or in part without the prior written permission of IBM.

10.1

Figure 13-3. Auditing ruleset execution

WB3951.2

Notes:

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

© Copyright IBM Corporation 2016

Figure 13-4. Auditing ruleset execution

WB3951.2

Notes:

This section introduces how you to use Decision Warehouse to audit ruleset execution.



Note

IBM ODM on Cloud

ODM on Cloud does not support 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

© Copyright IBM Corporation 2016

Figure 13-5. Decision Warehouse

WB3951.2

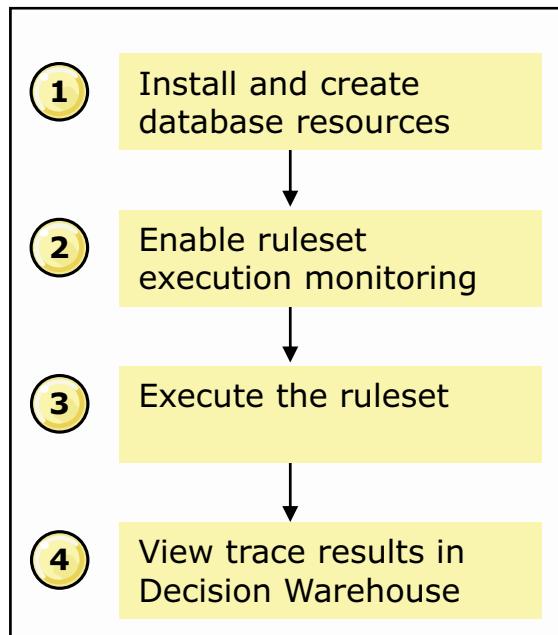
Notes:

Decision Warehouse allows users to view stored ruleset decision traces.

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. The trace contains information about how a decision was made. It records the executed ruleflow, the path of executed ruleflow tasks, and the rules that were fired. These details are intended to help users understand what happened as a result of executing a ruleset.

Default use of Decision Warehouse



© Copyright IBM Corporation 2016

Figure 13-6. Default use of Decision Warehouse

WB3951.2

Notes:

The following workflow illustrates the default use of Decision Warehouse, that is, without 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

© Copyright IBM Corporation 2016

Figure 13-7. Step 1: Install and create database resource

WB3951.2

Notes:



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`
 - `ruleset.sequential.trace.enabled` (if the ruleset contains rule tasks with the Sequential or Fastpath execution modes)
 - `ruleset.bom.enabled` (optional)

© Copyright IBM Corporation 2016

Figure 13-8. Step 2: Enable rule execution monitoring

WB3951.2

Notes:

To have your ruleset decision traces stored in Decision Warehouse, configure your ruleset to enable ruleset execution monitoring, by adding the following ruleset properties to the ruleset, and by setting their values to `true`:

- `monitoring.enabled`
- `ruleset.bom.enabled`

This property is optional. If it is not defined, Decision Warehouse acts as if it is defined and its value is `false`.

If this property is defined and its value is `true`, the input and output data is serialized and stored in a BOM representation of the data in XML format.

If this property is defined and its value is `false`:

- If the ruleset is based on a dynamic XOM, the input and output parameters are stored as XML code.
- If the ruleset is based on a Java XOM, the `toString` method of the ruleset parameter type stores the content.

- `rulesetSEQUENTIALTRACEenabled`

This property is required only if the ruleset contains rule tasks that use an execution mode other than RetePlus.

You can set these ruleset properties in:

- Rule Designer and Decision Center before you deploy a ruleset
- The Rule Execution Server console after you deploy the ruleset

You already learned how to manage (add, edit, remove) ruleset properties in Rule Designer, in Decision Center, and in the Rule Execution Server console. In the following topic, you learn a new way of defining these properties in the Rule Execution Server console.



Step 3: Execute the ruleset

- Execute your ruleset by using:
 - Decision Validation Services
 - A client application
 - A hosted transparent decision service

© Copyright IBM Corporation 2016

Figure 13-9. Step 3: Execute the ruleset

WB3951.2

Notes:

You can execute the ruleset by running tests or simulations, by using a client application, or by using a hosted transparent decision service.



Step 4-a: View stored decision traces

- Sign in to the Rule Execution Server console by using the `resMonitor` or `resAdmin` role
- In the Rule Execution Server console, click the **Decision Warehouse** tab
- Click **Search**

65 Decision(s) found Display by 10

Decision ID	Date	Ruleset Version	Number of rules fired	Decision Trace	Processing Time (ms)
5ab23322-97ef-4787-b892-58166e39cebc	2009-03-26 18:02:58	/prodra1238086969350_90b385fe_4e4c_4842_b408_4d1e9dcadfd6 /1.0/prodrs1238086969350_90b385fe_4e4c_4842_b408_4d1e9dcadfd6/1.0	1	View Decision details	416
50a3601b-20fa-4f78-abf7-6935fd4516c	2009-03-19 15:30:10	/prodra1237473006170_519ba66c_0ad6_4142_a3e9_3d7df42c7945 /1.0/prodrs1237473006170_519ba66c_0ad6_4142_a3e9_3d7df42c7945/1.0	1	View Decision details	35
6c44f221-50de-486f-89d9-ceab48042d84	2009-03-19 15:30:10	/prodra1237473006170_519ba66c_0ad6_4142_a3e9_3d7df42c7945 /1.0/prodrs1237473006170_519ba66c_0ad6_4142_a3e9_3d7df42c7945/1.0	0	View Decision details	37
a18b5329-ce90-4ea5-9a37-fb01e0b1e3e6	2009-03-19 15:30:08	/prodra1237473006170_519ba66c_0ad6_4142_a3e9_3d7df42c7945 /1.0/prodrs1237473006170_519ba66c_0ad6_4142_a3e9_3d7df42c7945/1.0	1	View Decision details	205
385ee32d-6bc8-49b1-a648-cd51c0c6f07d	2009-03-19 15:27:48	/prodra1237472858960_f57cfa85_481b_4e0e_a50a_9bd94a70e73e /1.0/prodrs1237472858960_f57cfa85_481b_4e0e_a50a_9bd94a70e73e/1.0	1	View Decision details	172
6602cfed-6c46-4fe5-8bc8-f1083940a28	2009-03-19 15:27:48	/prodra1237472858960_f57cfa85_481b_4e0e_a50a_9bd94a70e73e /1.0/prodrs1237472858960_f57cfa85_481b_4e0e_a50a_9bd94a70e73e/1.0	0	View Decision details	47
41565ac8-2456-47c4-b5ce-e8d925c8c9fa	2009-03-19 15:27:47	/prodra1237472858960_f57cfa85_481b_4e0e_a50a_9bd94a70e73e /1.0/prodrs1237472858960_f57cfa85_481b_4e0e_a50a_9bd94a70e73e/1.0	1	View Decision details	505
57705e85-ff38-45b3-a716-eb0a45fc676	2009-03-13 11:57:58	/prodra1236941862805_eee830e0_cefa_4105_a0b6_7851d0bca552 /1.0/prodrs1236941862805_eee830e0_cefa_4105_a0b6_7851d0bca552/1.0	4	View Decision details	1138
64efc105-58cd-4e71-8350-e550b957a610	2009-03-06 19:22:52	/prodra1236363766960_fad3c26c_ab69_46d9_9763_8d4ca0a422fc /1.0/prodrs1236363766960_fad3c26c_ab69_46d9_9763_8d4ca0a422fc/1.0	3	View Decision details	18
f91fe910-2585-4d18-987a-dbc9e06b4f77	2009-03-06 19:22:52	/prodra1236363766960_fad3c26c_ab69_46d9_9763_8d4ca0a422fc /1.0/prodrs1236363766960_fad3c26c_ab69_46d9_9763_8d4ca0a422fc/1.0	3	View Decision details	13

1 - 10 out of 65 results ◀◀◀ 1 2 3 4 5 6 7 ▶▶▶

© Copyright IBM Corporation 2016

Figure 13-10. Step 4-a: View stored decision traces

WB3951.2

Notes:

After your decision traces are stored in Decision Warehouse, you can use the **Decision Warehouse** tab of the Rule Execution Server console to query Decision Warehouse and view the stored decision traces.



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

© Copyright IBM Corporation 2016

Figure 13-11. Step 4-b: Filter stored decision traces

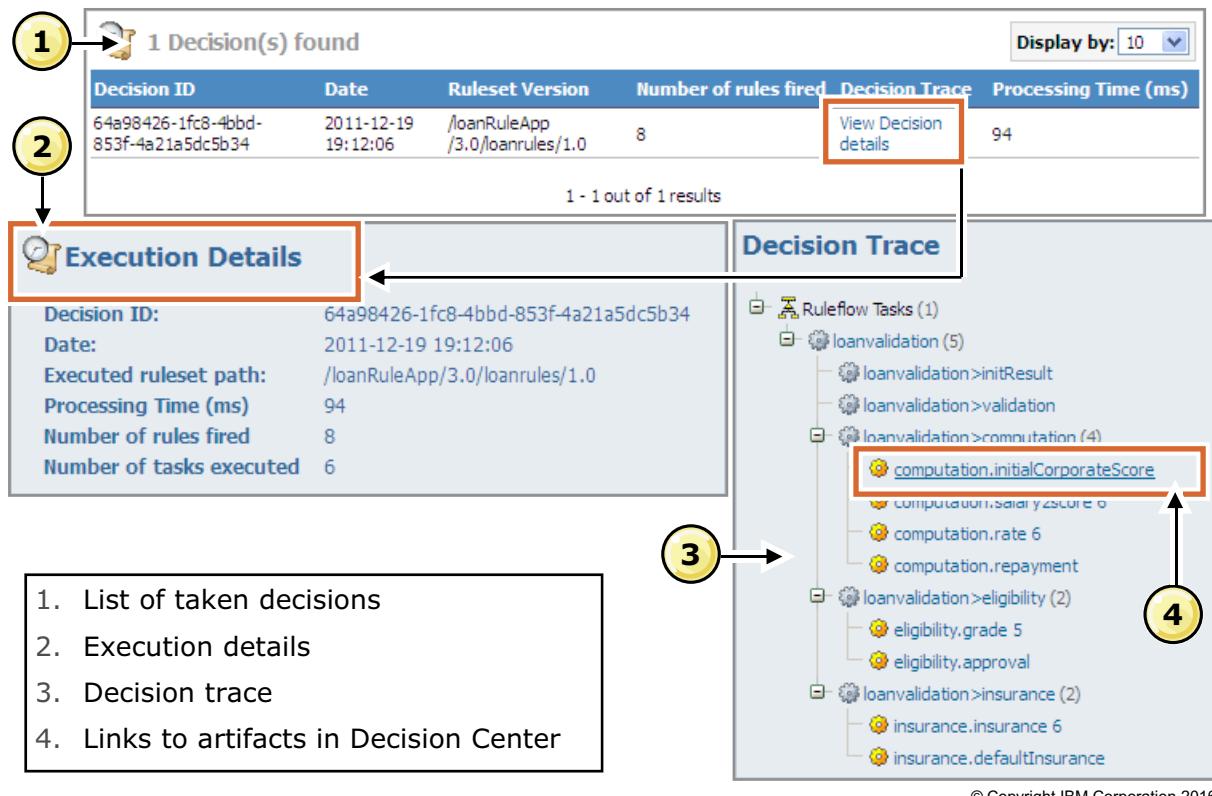
WB3951.2

Notes:

With the query feature, you can define filters on the specified data source so that only the events or decisions you are interested in are displayed. Use the **Decision ID** to locate a specific transaction and view the rules that were fired.

 WebSphere Education 

Step 4-c: View decision details and fired rules



Execution Details

Decision ID: 64a98426-1fc8-4bbd-853f-4a21a5dc5b34
Date: 2011-12-19 19:12:06
Executed ruleset path: /loanRuleApp/3.0/loanrules/1.0
Processing Time (ms): 94
Number of rules fired: 8
Number of tasks executed: 6

Decision Trace

- Ruleflow Tasks (1)
 - loanvalidation (5)
 - loanvalidation>initResult
 - loanvalidation>validation
 - loanvalidation>computation (4)
 - computation.initialCorporateScore
 - computation.salaryScore
 - computation.rate 6
 - computation.repayment
 - loanvalidation>eligibility (2)
 - eligibility.grade 5
 - eligibility.approval
 - loanvalidation>insurance (2)
 - insurance.insurance 6
 - insurance.defaultInsurance

1 1 Decision(s) found

Decision ID	Date	Ruleset Version	Number of rules fired	Decision Trace	Processing Time (ms)
64a98426-1fc8-4bbd-853f-4a21a5dc5b34	2011-12-19 19:12:06	/loanRuleApp/3.0/loanrules/1.0	8	View Decision details	94

1 - 1 out of 1 results

Display by: 10

© Copyright IBM Corporation 2016

Figure 13-12. Step 4-c: View decision details and fired rules

WB3951.2

Notes:

1. For each decision trace, you can click the **View Decision details** link to access the information details such as which rule tasks and rules were fired, and the input and output parameter values.
2. The Execution Details pane gives the overview information about the decision taken.
3. The Decision Trace pane gives the list of ruleflow tasks and rule artifacts that were executed.
4. If you deploy the RuleApp from Decision Center, the Decision Trace section contains links to the rule artifacts in Decision Center Repository. You can click these links to open Decision Center and view the corresponding rule artifact.

13.2. Monitoring ruleset execution

Monitoring ruleset execution



© Copyright IBM Corporation 2016

Course materials may not be reproduced in whole or in part without the prior written permission of IBM.

10.1

Figure 13-13. Monitoring ruleset execution

WB3951.2

Notes:



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

© Copyright IBM Corporation 2016

Figure 13-14. Monitoring ruleset execution

WB3951.2

Notes:



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

© Copyright IBM Corporation 2016

Figure 13-15. Debug mode

WB3951.2

Notes:

You enable or disable the debug mode as follows:

1. In the Navigator pane, click the relevant ruleset.
2. In the Ruleset View, click **Edit**.
3. The Ruleset Properties are now editable.
4. Select the check box next to the **Debug** property to enable the debug mode; clear this check box to disable the debug mode.
5. Click **Save** in the Ruleset View.



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

© Copyright IBM Corporation 2016

Figure 13-16. Ruleset monitoring options (1 of 5)

WB3951.2

Notes:

You learned about the properties indicated here in the previous topics. Rule Execution Server console also provides the Monitoring Options pane for you to define these properties.

To enable tracing in Decision Warehouse with Rule Execution Server console:

1. Click the **Explorer** tab.
2. In the Navigator pane, click the relevant ruleset.
3. In the Ruleset View, click **Show Monitoring Options**.
4. At the bottom of the monitoring options section, click the **Edit** icon.
5. Select the **Enable tracing in the Decision Warehouse** option.
6. Select the appropriate options (that you see next)

The options that you select are turned into ruleset properties, as you see next.

WebSphere Education

Ruleset monitoring options (2 of 5)

The figure consists of two side-by-side screenshots of a configuration interface. Both screenshots have a header with 'Hide Properties' and a gear icon, followed by a section for '2 properties' and '3 properties' respectively.

Left Screenshot (2 properties):

- Properties listed: 'monitoring.enabled' and 'ruleset.sequential.trace.enabled'.
- Section below: 'Upload properties from file' with a 'Choose file:' input field and a 'Browse...' button.
- Section below: 'Hide Monitoring Options' with a 'monitoring options' icon. It includes a checked checkbox for 'Enable tracing in the Decision Warehouse' and a radio button group for 'Store the values of the ruleset parameters to:' with 'Native format' selected.

Right Screenshot (3 properties):

- Properties listed: 'monitoring.enabled', 'ruleset.bom.enabled', and 'ruleset.sequential.trace.enabled'.
- Section below: 'Upload properties from file' with a 'Choose file:' input field and a 'Browse...' button.
- Section below: 'Hide Monitoring Options' with a 'monitoring options' icon. It includes a checked checkbox for 'Enable tracing in the Decision Warehouse' and a radio button group for 'Store the values of the ruleset parameters to:' with 'BOM format with optional filter:' selected.

A vertical double-headed arrow points between the 'Native format' radio button in the left screenshot and the 'BOM format with optional filter:' radio button in the right screenshot.

© Copyright IBM Corporation 2016

Figure 13-17. Ruleset monitoring options (2 of 5)

WB3951.2

Notes:

You can specify how to store the ruleset parameter values:

- **Native format:** To turn off the BOM serialization of the objects in the ruleset parameters (screen capture on the left of the graphic).

Native format is the default value that corresponds to the absence of the `ruleset.bom.enabled` property.

In native format, the ruleset parameters are stored as follows:

- Java XOM: The ruleset parameters are stored as string representations that are obtained by calling `java.lang.Object.toString`
- Dynamic XOM: The ruleset parameters are stored as XML.

- **BOM format with an optional filter:** To convert the list of objects in the ruleset parameters into a memory buffer by using BOM serialization (screen capture on the right of the graphic).

BOM format with an optional filter is set when you define the `ruleset.bom.enabled` property, and set its value to: `true`

With this format, the ruleset parameters are stored as BOM XML, with or without a filter.



Ruleset monitoring options (3 of 5)

Hide Properties

4 properties

Select All	Name	Value
<input type="checkbox"/>	monitoring.enabled	true
<input type="checkbox"/>	monitoring.inout.filters	borrower.some.heavy.property,loan.longfield
<input type="checkbox"/>	ruleset.bom.enabled	true
<input type="checkbox"/>	rulesetSEQUENTIAL.trace.enabled	true

properties 1 - 4 of 4

Upload properties from file

Choose file: Override ex

Hide Monitoring Options

monitoring options

Enable tracing in the Decision Warehouse

Store the values of the ruleset parameters to:

BOM format with optional filter: [borrower.some.heavy.property,loan.longfield]

Native format

© Copyright IBM Corporation 2016

Figure 13-18. Ruleset monitoring options (3 of 5)

WB3951.2

Notes:

When you use the **BOM format with an optional filter**, you might set a filter on the objects that the ruleset parameters use to remove information about these objects from the trace.

As an example, the following filter removes the `some.heavy.property` property from the `borrower` class, and the `longfield` field from the `loan` class:

`borrower.some.heavy.property,loan.longfield`




Ruleset monitoring options (4 of 5)

[Hide Monitoring Options](#)

 monitoring options

Enable tracing in the Decision Warehouse

Store the values of the ruleset parameters to:

BOM format with optional filter:

Native format

Select the execution traces to store in the Decision Warehouse:

Execution Date
 Execution Duration
 Total Number of Tasks Executed
 Total Number of Tasks Not Executed
 Total Number of Rules Fired
 Total Number of Rules Not Fired
 Execution Events
 List of All Tasks
 List of Tasks Not Executed
 List of All Rules
 List of Rules Not Fired
 Bound Object by Rule
 System Properties
 Working Memory (with optional filter:)

 Save  Cancel

- The possible trace options are as follows:
 - Execution Date
 - Execution Duration
 - Total Number of Tasks Executed
 - Total Number of Tasks Not Executed
 - Total Number of Rules Fired
 - Total Number of Rules Not Fired
 - Execution Events
 - List of All Tasks
 - List of Tasks Not Executed
 - List of All Rules
 - List of Rules Not Fired
 - Bound Object by Rule
 - System Properties
 - Working Memory (with optional filter)

© Copyright IBM Corporation 2016

Figure 13-19. Ruleset monitoring options (4 of 5)

WB3951.2

Notes:

The possible trace options are as follows:

- **Execution Date:** The execution start date
- **Execution Duration:** A measure of time that the engine took to execute the ruleset
- **Total Number of Tasks Executed:** The number of task instances that the engine executed
- **Total Number of Tasks Not Executed:** The number of ruleflow tasks that the engine did not execute
- **Total Number of Rules Fired:** The number of rule instances that the engine executed
- **Total Number of Rules Not Fired:** The number of rules that the engine did not execute
- **Execution Events:** The full execution tree for the executed ruleset
 The tree is composed of executed task instances and executed rule instances.
- **List of All Tasks:** The ruleflow tasks in the ruleset
- **List of Tasks Not Executed:** The ruleflow tasks that the engine did not execute

To get the list of rule instances that were executed, use the **Execution Events** trace option.

- **List of All Rules:** All the rules in the ruleset.
- **List of Rules Not Fired:** All the rules that the engine did not execute.

To get the list of task instances that were executed, use the Execution Events trace option.

- **Bound Object by Rule:** Used with the Execution Events trace option.

When set, the rules that are executed in the execution tree are bound to the objects to which they were applied.

- **System Properties:** The JRE system properties that Rule Execution Server uses.

For example: `JAVA_HOME`

- **Working Memory** (with optional filter): The contents of the working memory.

Use the optional filter to define specific classes that you want to record.

The following filter records the objects that are specified in the classes `MyClass` and `MyOtherClass` of the package `com.xyz`:

`com.xyz.MyClass, com.xyz.MyOtherClass`

WebSphere Education

Ruleset monitoring options (5 of 5)

Hide Properties

5 properties

Select All	Name	Value
<input type="checkbox"/>	monitoring.enabled	true
<input type="checkbox"/>	monitoring.filters	INFO_EXECUTION_DATE=true,INFO_EXECUTION_DURATION=true
<input type="checkbox"/>	monitoring.inout.filters	borrower.some.heavy.property,loan.longfield
<input type="checkbox"/>	ruleset.bom.enabled	true
<input type="checkbox"/>	rulesetSEQUENTIAL.trace.enabled	true

properties 1 - 5 of 5

prev 10 next 10

Upload properties from file

Choose file: [Browse...](#) [Proceed to update](#) [Preview update](#) Override existing properties

Hide Monitoring Options

monitoring options

Tracing in the Decision Warehouse is currently enabled

Ruleset parameters will be stored as BOM XML (with filter borrower.some.heavy.property,loan.longfield)

The following execution traces will be stored:

- Execution Date
- Execution Duration

[Edit](#)

© Copyright IBM Corporation 2016

Figure 13-20. Ruleset monitoring options (5 of 5)

WB3951.2

Notes:

If you do not select all the available options for the decision traces, you filter some of the decision traces. The `monitoring.filters` property is automatically added to the ruleset, with the list of decision traces that you select.

Removing the `monitoring.filters` property from the ruleset has the same effect as selecting all the possible decision traces (that is, setting no filters).

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

© Copyright IBM Corporation 2016

Figure 13-21. Ruleset statistics (1 of 3)

WB3951.2

Notes:

To generate statistics on the previous executions of a ruleset:

- In the Navigator pane, click the relevant ruleset.
- In the Ruleset View, click **View Statistics**.

The Ruleset Statistics View opens and provides a ruleset execution statistics table for each Execution Unit (XU) in the configuration, and consolidated statistics on the entire cluster.

You can see the following ruleset statistics in the Ruleset Statistics View:

- Count:** The number of times the ruleset was executed during this session
- Total time (ms):** The total time to execute the ruleset
- Average time (ms):** The average time to execute the ruleset
- Max / Min time (ms):** The longest and shortest ruleset execution times
- Last Execution Time (ms):** The time of the last ruleset execution
- First / Last Execution Date:** The dates and times of the first and last ruleset executions

The average time (Average time (ms)) is derived from the total execution time (Total time (ms)) and the number of executions completed (Count).

Notice that the ruleset statistics that you can see here are similar to the ones that you saw for transparent decision services.



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

© Copyright IBM Corporation 2016

Figure 13-22. Ruleset statistics (2 of 3)

WB3951.2

Notes:

The execution mode depends on the state of the request, that is, on how the instance of the `IlrSessionRequest` class is configured.

- If the method `IlrSessionRequest.setTaskName(java.lang.String taskName)` is called, the `IlrContext.executeTask` method is eventually called, and the execution mode is reported as a **Task Execution**.
- If this `setTaskName` method is not called, the execution mode is reported as a **Ruleset Execution**.

The following code sets the execution mode to **Task Execution**:

```
IlrSessionFactory sessionFactory = new IlrPOJOSessionFactory();
IlrSessionRequest request = sessionFactory.createRequest();
request.setRulesetPath(IlrPath.parsePath("/myRuleapp/myRuleset"));
request.setTaskName("myTask");
IlrStatelessSession session = sessionFactory.createStatelessSession();
IlrSessionResponse response = session.execute(request);
```



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

© Copyright IBM Corporation 2016

Figure 13-23. Ruleset statistics (3 of 3)

WB3951.2

Notes:

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

© Copyright IBM Corporation 2016

Figure 13-24. Test of ruleset execution

WB3951.2

Notes:

The web testing interface in the Rule Execution Server console is not a programmatic interface. It is not designed for automated execution.

You can use this feature only when the console is deployed in a Java EE environment where an Execution Unit (XU) is reachable through a JNDI lookup. You can verify that your deployment conforms to this configuration by checking that the result of the XU Lookup diagnostic test is green.



Warning

If you add the `ruleset.managedxom.uris` ruleset property to a ruleset, all execution requests use this same property value. In a production environment, this behavior is unlikely to be the intended one.



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

© Copyright IBM Corporation 2016

Figure 13-25. Logged events on Execution Units

WB3951.2

Notes:

To view logged events on Execution Units (XU):

1. Click the **Explorer** tab.
2. In the Navigator pane, click the relevant ruleset.
3. On the Ruleset View page, click **View Execution Units**.

The Execution Units page is displayed. The Execution Unit (XU) table provides the number of warnings and errors that are logged on each server.

4. Click the name of the server in the list of deployed Execution Units.

The events information can be displayed in different ways:

- To limit the displayed log, use the **Display by: 5, 10, 50, 100** filter.
- To sort the log messages by column (**Level, Creation Date, Message**) in ascending order, click the column name. To sort in descending order, click the column name again.

To update the model and the log, click **Refresh** at the top of the page.

To reset the log, click **Reset Execution Unit Messages**.



Unit summary

Having completed this unit, you should be able to:

- Audit the execution of rulesets with Decision Warehouse
- Monitor ruleset execution with the Rule Execution Server console

© Copyright IBM Corporation 2016

Figure 13-26. Unit summary

WB3951.2

Notes:



Checkpoint questions

1. **True or False:** To have decision traces, you must enable ruleset monitoring.
2. **True or False:** If you deploy the RuleApp from Rule Designer, you have links to rule artifacts in Rule Designer available from the selected decision in Decision Warehouse.
3. **True or False:** In the Rule Execution Server console, you can test any of the deployed rulesets.

© Copyright IBM Corporation 2016

Figure 13-27. Checkpoint questions

WB3951.2

Notes:

Write your answers here:

- 1.
- 2.
- 3.



Checkpoint answers

1. **True.**
2. **False:** *If you deploy the RuleApp from Decision Center, you have links to rule artifacts in Decision Center available from the selected decision in Decision Warehouse.*
3. **False:** *In the Rule Execution Server console, you can test only the deployed rulesets that are associated with a managed XOM.*

© Copyright IBM Corporation 2016

Figure 13-28. Checkpoint answers

WB3951.2

Notes:

Exercise 19



Auditing ruleset execution through
Decision Warehouse

© Copyright IBM Corporation 2016

Course materials may not be reproduced in whole or in part without the prior written permission of IBM.

10.1

Figure 13-29. Exercise 19

WB3951.2

Notes:

Exercise objectives

After completing this exercise, you should be able to:

- Enable monitoring for ruleset execution
- Retrieve decision traces through Decision Warehouse
- Optimize Decision Warehouse
- Delete trace data from Decision Warehouse

© Copyright IBM Corporation 2016

Figure 13-30. Exercise objectives

WB3951.2

Notes:

Unit 14. Working with the REST API

What this unit is about

This unit teaches you how to use the REST API for ruleset execution.

What you should be able to do

After completing this unit, you should be able to:

- Describe the ruleset execution REST API

How you will check your progress

- Checkpoint
- Exercise



Unit objectives

After completing this unit, you should be able to:

- Describe the ruleset execution REST API

© Copyright IBM Corporation 2016

Figure 14-1. Unit objectives

WB3951.2

Notes:



Topics

- Overview of the ruleset execution REST API
- Testing ruleset execution with the REST API
- Managing resources through the REST API

© Copyright IBM Corporation 2016

Figure 14-2. Topics

WB3951.2

Notes:

14.1.Overview of the ruleset execution REST API

Overview of the ruleset execution REST API



© Copyright IBM Corporation 2016

Course materials may not be reproduced in whole or in part without the prior written permission of IBM.

10.1

Figure 14-3. Overview of the ruleset execution REST API

WB3951.2

Notes:

REST API for ruleset execution

- Rule Execution Server provides a Representational State Transfer (REST) API for ruleset execution
 - Provides XML generation and XSD validation
 - Execute rulesets with XML format through the HTTP protocol
- The REST service for ruleset execution provides these benefits:
 - No client library or complex configuration is required to interact with a remote Rule Execution Server instance
 - Work across platforms or from various client applications
 - Easy to switch from local to remote Rule Execution Server execution
- You can test REST API ruleset execution requests in Rule Execution Server console

© Copyright IBM Corporation 2016

Figure 14-4. REST API for ruleset execution

WB3951.2

Notes:

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

The REST services include:

- XML payload sample generation
- XML and XSD validation
- Ruleset execution services

With these REST services, you do not need to add any client libraries or perform complex configuration, which is typically required with the JMX API.

By using the REST API, you can work across environments or from other client applications, such as JavaScript clients. Also, you can easily switch from local execution to a remote Rule Execution Server instance.

To verify that your ruleset execution requests use well-formatted XML, you can test the REST API requests in the Rule Execution Server console. You can also use REST API to integrate ruleset execution with other IBM products.



Endpoint URLs

- The REST API includes URIs that represent the rulesets
- URI format

`http://host:port/DecisionService/rest/v1/rulesetPath?options`

- *host:port/DecisionService*: Host address and port for the HTDS application
- */rest/v1*: Context root and version of the REST service
- *rulesetpath*: Can be either canonical or non-canonical
- *options*: Options for generation of WADL file
 - *inline*: Generates and displays a WADL file with the XSD schema included
 - *zip*: Generates a compressed file for download

© Copyright IBM Corporation 2016

Figure 14-5. Endpoint URLs

WB3951.2

Notes:

The endpoint URLs represent the rulesets, which are the Rule Execution Server resources for the REST execution service.

The first part of the URI consists of the host address and port of the hosted transparent decision service (HTDS) application.

In the next part, */rest/v1*, *rest* means the REST service context root and *v1* is the version number of the REST service.

In the ruleset path, the short ruleset path or the canonical ruleset path is expected. A canonical ruleset path includes the name and version number of the rule application, followed by the name and version number of the ruleset. A short ruleset path leaves out one or both version numbers, which would be the names of the rule application and ruleset.

The valid value for the last options can be *inline* or *zip* parameter for the Web Application Description Language (WADL) generation. If you set the *inline* parameter, the XSD schema is included in the WADL and displayed to you right away. Otherwise, use the *zip* option to download the compressed file and view it locally.

HTTP methods

- GET method
 - Generate a sample XML payload
 - Retrieve the WADL for a specified ruleset
- POST method
 - Create an execution request
 - Validate an XML payload
- HTTP header fields
 - The **Accept-Language** header field is equivalent to the **accept-language** URI parameter for a list of languages that are valid in response message
 - The **Content-Type** field accepts only: **application/xml**

© Copyright IBM Corporation 2016

Figure 14-6. HTTP methods

WB3951.2

Notes:

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

You use the POST method to validate an XML payload and to execute the ruleset. The request body for the execution request contains the XML payload, so before sending the request, it is good practice to first validate it.

For the HTTP header, the **Accept-Language** field is the same as the **accept-language** URI parameter. It takes a list of languages as valid values.

Only XML format is supported, so the **Content-Type** field accepts only the **application/xml** value.

Request and response schema

- Request contains:
 - The ruleset IN and INOUT parameters in alphabetical order
 - A Decision ID (optional)
 - A trace filter (optional)
- Execution response (XML) contains:
 - The ruleset INOUT and OUT parameters, in alphabetical order
 - The Decision ID
 - The trace, if specified in the request
- Validation responses are returned in JSON format
 - Responses to valid requests contains an empty JSON list []
 - Responses to invalid requests describe the type and location of the error
- Example of validation response to invalid request:

```
{"type": "Error", "line": 9, "column": 22, "message": "cvc-
datatype-valid.1.2.1: '5d' is not a valid value for
'integer'."}
```

© Copyright IBM Corporation 2016

Figure 14-7. Request and response schema

WB3951.2

Notes:

In the ruleset execution REST service, requests and responses follow different schemas, depending on whether the ruleset XOM is based on XML classes or on Java classes. The schema determines how the types are serialized.

With the ruleset signature, the request part is composed of the following elements:

- The IN and INOUT parameters of the ruleset, in alphabetical order.
- An optional decision ID if you want to set it to a specific value and an optional trace filter.

The execution response consists of:

- The INOUT and OUT parameters of the ruleset, in alphabetical order.
- The decision ID, either the default value or the value that you set in the request.
- If you set trace filter in request, the trace is returned in response.

Notice that the execution response is sent in XML format. The XML payload is analyzed against the generated XSD files.

The validation response is returned in JSON format.

- If the request is valid, the response is an empty JSON list [].
- If the request is invalid, the tool returns the list of errors.
- Each response to an invalid request includes:
 - **Type:** Error, Fatal, or Warning
 - **Line:** The number of the line that contains the error in the XML file
 - **Column:** The column number that contains the error in the XML file
 - **Message:** Description of error

The example on the slide shows an error in line 9, column 22. Based on the XML schema data type validation rule, the value 5d is not a valid integer.



XML serialization of the XOM

- Primitive types: The ruleset parameter name (in the parameter namespace) is used in XML element serialization
- Dynamic XSD types
 - Ruleset parameter is an XML element, and that element is used as the root element name of the parameter in requests and responses
 - Ruleset parameter is a complex or primitive Java type; the parameter name is used as the root element name of the parameter in requests and responses, within the configured param namespace
- Java classes:
 - The ruleset parameter name (in the parameter namespace) is used as the name of the XML root element
 - Arrays are supported, including simple and multi-dimensional arrays

© Copyright IBM Corporation 2016

Figure 14-8. XML serialization of the XOM

WB3951.2

Notes:

In the REST service for ruleset execution, primitive Java types, XSD types, and Java XOM classes are serialized differently.

For the primitive types, the REST service XML element that is used to serialize the primitive type is the name of the ruleset parameter in the parameter namespace. Examples of primitive types include boolean, string, or simple classes like java.lang.Boolean.

For dynamic XSD type XOMs, the request and response XML root element depends on how the ruleset parameters are defined. It is serialized from the original XSD schema.

If the dynamic XOM ruleset parameter is an XML element, then this XML element name becomes the XML request root element name. See the example MyConference XSD schema on the left. And on the right, the ruleset parameter is declared from the MyConference XML element.

If the dynamic XOM ruleset parameter is the Java type, the parameter name is used as the root element name in the request and response.

For Java XOMs, basically the ruleset parameter name is in use. The Java array type is also supported, for both simple and multidimensional arrays.



WADL representation

You generate WADL representation to write the XML payload of a specific ruleset execution request

- URI:

```
http://{host}:{port}/DecisionService/rest/v1/
{rulesetPath}/wadl
```

A set of XSD files are also generated, but separated from WADL file

- Optional parameters:
 - inline: The .xsd files are included in the .wadl file
 - zip: The .xsd files and .wadl file are generated in a compressed file

Response result:

- The WADL file
 - One GET method function: XML payload sample generation
 - Two POST method functions: XML payload validation and ruleset execution
- XSD files to describe XML representation of the input and output objects

© Copyright IBM Corporation 2016

Figure 14-9. WADL representation

WB3951.2

Notes:

The WADL representation contains all the information of the request and response elements. You can use the WADL representation as a reference while you develop your client application.

By default, a set of XSD files are generated and they are separated from the WADL file. If you specify the `inline` option, the XSD file contents are included in the WADL file. Alternatively, to get WADL code and its XSD files to a compressed file, add the `zip` parameter.

If it is a valid request, you get a WADL representation of the request and response documentation. It contains one GET method function for XML payload sample generation, and two POST method functions for XML payload validation and ruleset execution. The attached XSD files describe the XML representation of the input and output objects.



Using the REST execution API for ruleset execution

1. Ensure that the ruleset is deployed
2. Generate a sample XML payload
 - Example:
GET `http://host:port/DecisionService/rest/v1/rulesetPath/xml`
 - Use the sample XML fragment as a starting point for writing an XML request
3. Validate the XML structure
 - Example:
POST `http://host:port/DecisionService/rest/v1/rulesetPath/validate`
4. Send the execution request
 - Example:
POST `http://host:port/DecisionService/rest/v1/rulesetPath`
5. Expect the execution response to be returned in XML format

© Copyright IBM Corporation 2016

Figure 14-10. Using the REST execution API for ruleset execution

WB3951.2

Notes:

First, you make sure that the ruleset is deployed.

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

When your XML request is ready, you can validate its XML structure by posting a “validate” request.

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

14.2. Testing ruleset execution with the REST API

Testing ruleset execution with the REST API



© Copyright IBM Corporation 2016

Course materials may not be reproduced in whole or in part without the prior written permission of IBM.

10.1

Figure 14-11. Testing ruleset execution with the REST API

WB3951.2

Notes:



Executing a ruleset (1 of 2)

- Open Rule Execution Server console to the **Explorer** tab

The screenshot shows the IBM Rule Execution Server console interface. At the top, there's a navigation bar with tabs: Home, Explorer (which is highlighted with a red box), Decision Warehouse, Diagnostics, Server Info, and REST API. Below the navigation bar, the URL path is visible: Explorer > RuleApp > RuleApp > Ruleset. On the right side of the page, there's a link "Skip to main content".

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

The screenshot shows the Ruleset View page for the ruleset '/miniloanruleapp/1.0/miniloanrules/1.0'. The page includes various buttons like Test Ruleset, View Statistics, etc., and a prominent 'Retrieve HTDS Description File' button, which is highlighted with a red box. The URL in the address bar is '/miniloanruleapp/1.0/miniloanrules/1.0'.

© Copyright IBM Corporation 2016

Figure 14-12. Executing a ruleset (1 of 2)

WB3951.2

Notes:



Executing a ruleset (2 of 2)

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

The screenshot shows a dialog box titled "Retrieve HTDS Description File". At the top, there is a gear icon and the URL `/miniloanruleapp/1.0/miniloanrules/1.0`. Below the URL, there is a section labeled "Decision service type" with two radio buttons: "SOA" and "REST", where "REST" is selected. There are four checkboxes below this section:

- Latest ruleset version
- Latest RuleApp version
- Decision trace information
- Inline types in separate XSD files

At the bottom of the dialog box are four buttons: "Cancel", "View", "Download", and "Test", with "Test" being highlighted by a red box.

© Copyright IBM Corporation 2016

Figure 14-13. Executing a ruleset (2 of 2)

WB3951.2

Notes:

Validating the XML execution request

- Create the execution request in XML
- Before sending the request, you validate the XML structure

Decision Service : /miniloanruleapp/1.0/miniloanrules/1.0 REST Service

Execution Request:



```

1 <par:Request xmlns:par="http://www.ibm.com/rules/decisionservice/Miniloanruleapp/Miniloanrules/param">
2   <!--Optional:-->
3   <par:DecisionID>string</par:DecisionID>
4   <!--Optional:-->
5   <par:borrower>
6     <creditScore>600</creditScore>
7     <yearlyIncome>80000</yearlyIncome>
8   </par:borrower>
9   <!--Optional:-->
10  <par:loan>
11    <amount>2000000</amount>
12    <approved>true </approved>
13    <duration>240</duration>
14    <yearlyInterestRate>.05</yearlyInterestRate>
15  </par:loan>
16 </par:Request>

```

© Copyright IBM Corporation 2016

Figure 14-14. Validating the XML execution request

WB3951.2

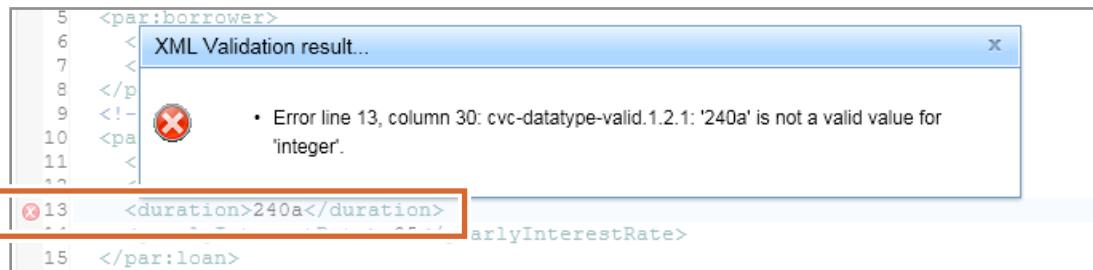
Notes:

The test tool opens where you can create the execution request. Before you send the request, you can validate the XML structure in this interface.



Validation response in JSON

- If an error is detected in the request, the validation response shows the type and location of the error in JSON format



The screenshot shows a window titled "XML Validation result..." with XML code and an error message. The XML code is:

```
5 <par:borrower>
6   < XML Validation result...
7   <
8   </p
9   <!--
10  <pa
11  <
12  <
13  <duration>240a</duration>
14  --> <earlyInterestRate>
15  </par:loan>
```

An error message is displayed in a callout box:

- Error line 13, column 30: cvc datatype-valid.1.2.1: '240a' is not a valid value for 'integer'.

- In this case, the expected value is an integer, but a letter was accidentally added in line 13, which caused an error

© Copyright IBM Corporation 2016

Figure 14-15. Validation response in JSON

WB3951.2

Notes:



Execution response

- After sending the validated request, the execution response is returned in XML format
 - In this case, the **approved** attribute indicates successful results

```
Execute Request  
  
Server Response:  
  
<?xml version="1.0" encoding="UTF-8"?><par:Response  
    <par:DecisionID>string</par:DecisionID>  
    <par:loan>  
        <approved>false</approved>  
        <yearlyInterestRate>0.05</yearlyInterestRate>  
    </par:loan>  
</par:Response>
```

- Understanding how to test ruleset execution REST API can be useful to develop your client application

© Copyright IBM Corporation 2016

Figure 14-16. Execution response

WB3951.2

Notes:

During the exercise, you learn how the REST API can be integrated into a client application to test execution.

14.3.Managing resources through the REST API

Managing resources through the REST API



© Copyright IBM Corporation 2016

Course materials may not be reproduced in whole or in part without the prior written permission of IBM.

10.1

Figure 14-17. Managing resources through the REST API

WB3951.2

Notes:



REST API tool in Rule Execution Server console

- Use the **REST API** tab in the Rule Execution Server console to manage REST resources through HTTP methods

REST API tool

You can use this test tool to test the management REST API. To test the ruleset execution REST API, click the Explorer tab and navigate to the Ruleset View.

REST API WADL file: /res/apiauth/v1/DecisionServer.wadl

/ruleapps /rulesets /libraries /xoms

GET /ruleapps
getRuleApps Returns all the RuleApps contained in the repository.

GET /ruleapps?count=true
getCountOfRuleApps Counts the number of elements in this list.

POST /ruleapps
deployRuleAppArchive Deploys a RuleApp archive in the repository, based on the merging and versioning policies passed as parameters. The RuleApp archive request body.

POST /ruleapps
addRuleApp Adds a new RuleApp in the repository. The RuleApp representation is passed in the request body in JSON or XML format. If the RuleApp already exists in the repository, the response body contains a specific error description and the HTTP status 202 is returned.

© Copyright IBM Corporation 2016

Figure 14-18. REST API tool in Rule Execution Server console

WB3951.2

Notes:

From the **REST API** tab, you can manage resources through HTTP methods.



Unit summary

Having completed this unit, you should be able to:

- Describe the ruleset execution REST API

© Copyright IBM Corporation 2016

Figure 14-19. Unit summary

WB3951.2

Notes:

Checkpoint questions

1. **True or False:** As a starting point for writing a ruleset execution request, generate a sample XML payload by retrieving the WADL file.
2. **True or False:** To make sure that you create a valid ruleset execution request in well-formatted XML, first send a “validate” request.
3. **True or False:** All execution and validation responses are returned in JSON format.

© Copyright IBM Corporation 2016

Figure 14-20. Checkpoint questions

WB3951.2

Notes:

Write your answers here:

- 1.
- 2.
- 3.



Checkpoint answers

1. **True.**
2. **True.**
3. **False.** *Execution responses are returned in XML. Responses to validation requests are returned in JSON format.*

© Copyright IBM Corporation 2016

Figure 14-21. Checkpoint answers

WB3951.2

Notes:

Exercise 20



Working with the REST API

© Copyright IBM Corporation 2016

Course materials may not be reproduced in whole or in part without the prior written permission of IBM.

10.1

Figure 14-22. Exercise 20

WB3951.2

Notes:



Exercise objectives

After completing this exercise, you should be able to:

- Use the REST API to test ruleset execution and manage RuleApp resources

© Copyright IBM Corporation 2016

Figure 14-23. Exercise objectives

WB3951.2

Notes:

Unit 15. Introducing decision governance

What this unit is about

In this unit, you learn how to identify governance issues and use Operational Decision Manager features to support decision governance.

What you should be able to do

After completing this unit, you should be able to:

- Explain governance issues and good practices
- Identify Operational Decision Manager features that support decision governance
- Describe how to implement the decision governance framework

How you will check your progress

- Checkpoint
- Exercise

Unit objectives

After completing this unit, you should be able to:

- 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 2016

Figure 15-1. Unit objectives

WB3951.2

Notes:



Topics

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

© Copyright IBM Corporation 2016

Figure 15-2. Topics

WB3951.2

Notes:

15.1.What is decision governance?

What is decision governance?



© Copyright IBM Corporation 2016

Course materials may not be reproduced in whole or in part without the prior written permission of IBM.

10.1

Figure 15-3. What is decision governance?

WB3951.2

Notes:

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

© Copyright IBM Corporation 2016

Figure 15-4. What is decision governance?

WB3951.2

Notes:

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 changing business policies so that updates can be implemented, tested, and deployed to production with minimal dependence on IT. However, when implementing 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

© Copyright IBM Corporation 2016

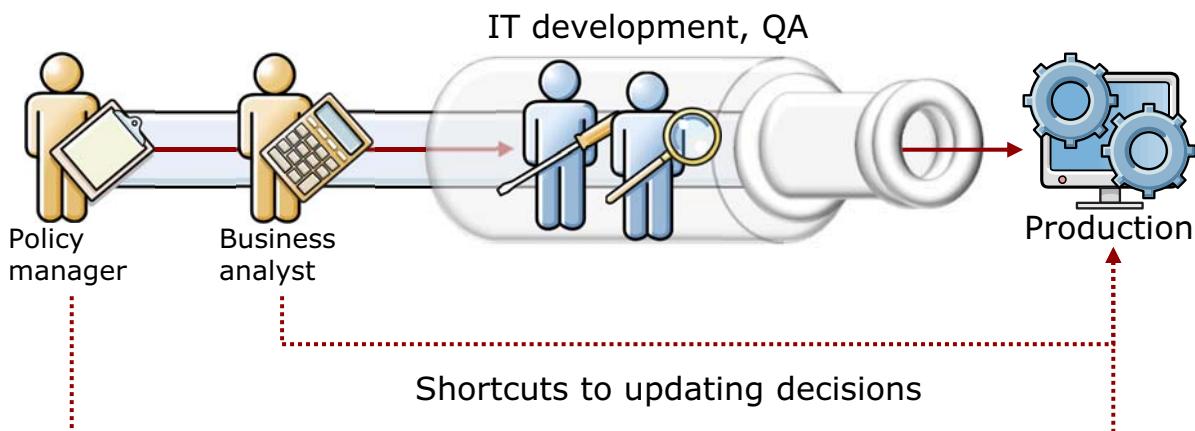
Figure 15-5. Why decision governance is required

WB3951.2

Notes:

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



© Copyright IBM Corporation 2016

Figure 15-6. Traditional approach to maintenance

WB3951.2

Notes:

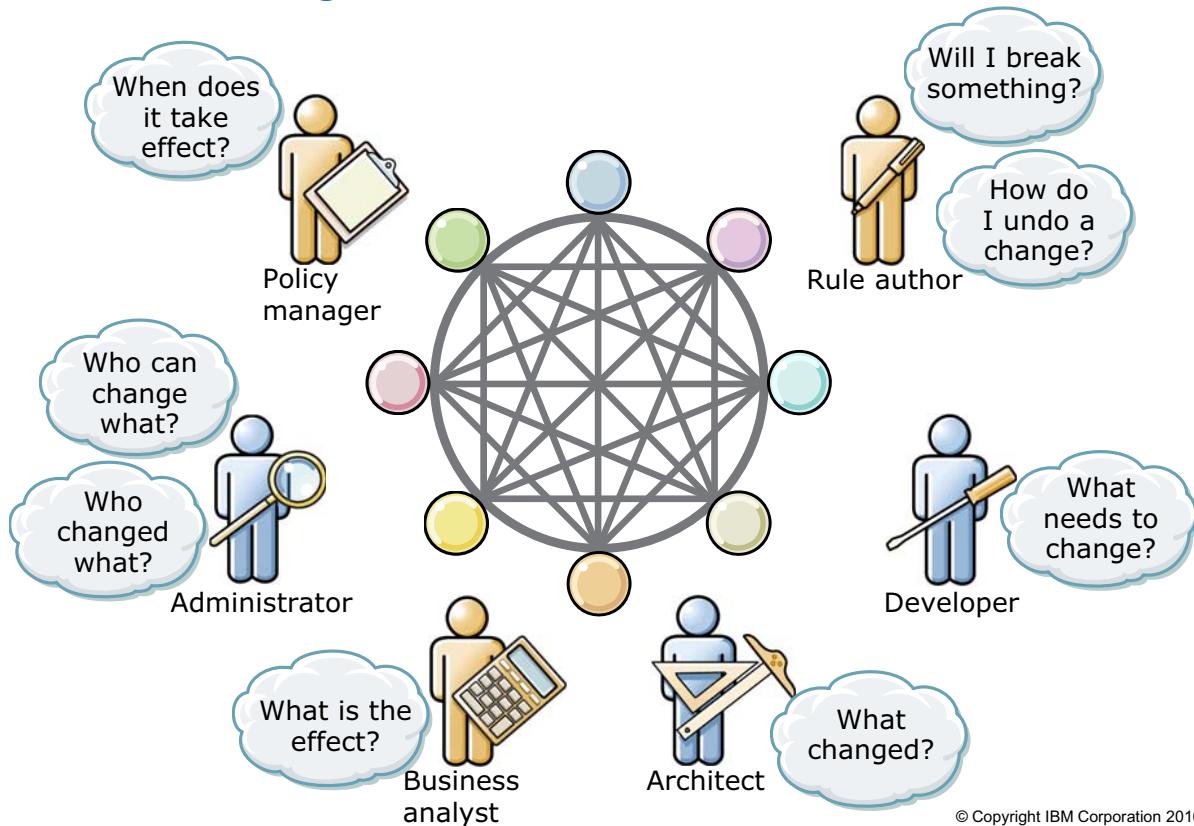
In the traditional approach, the IT development-QA pair can represent a bottleneck. Business policy updates, which are the new requirements that come from the business team, must go through that bottleneck.

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



© Copyright IBM Corporation 2016

Figure 15-7. Decision management increases communication

WB3951.2

Notes:

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 requires frequent short questions that must be quickly answered and implemented.

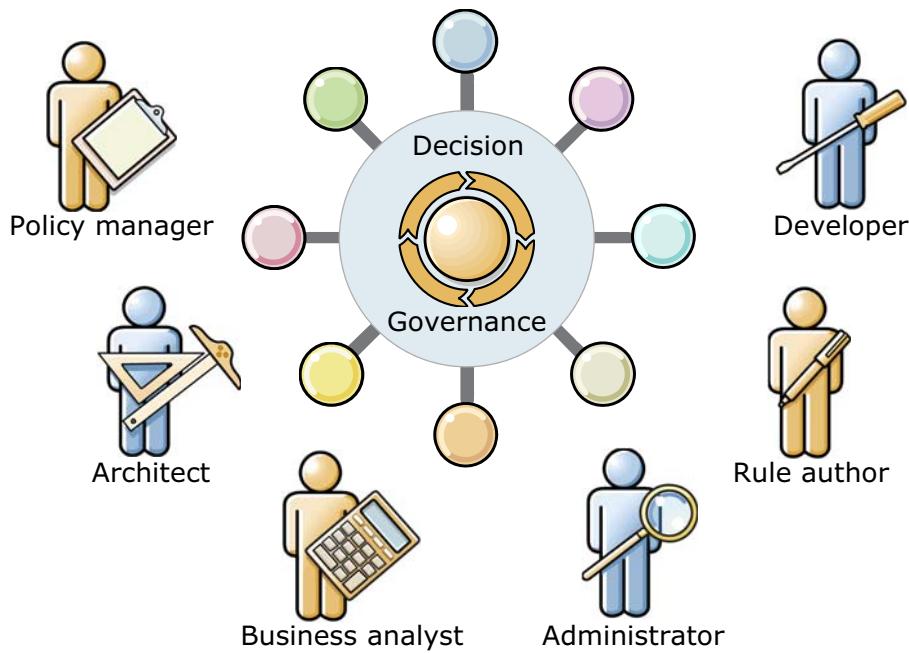
Examples:

- 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



© Copyright IBM Corporation 2016

Figure 15-8. Decision governance goal

WB3951.2

Notes:

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

© Copyright IBM Corporation 2016

Figure 15-9. Definition of project governance

WB3951.2

Notes:

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 Decision 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

© Copyright IBM Corporation 2016

Figure 15-10. Business Decision Management group

WB3951.2

Notes:

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.

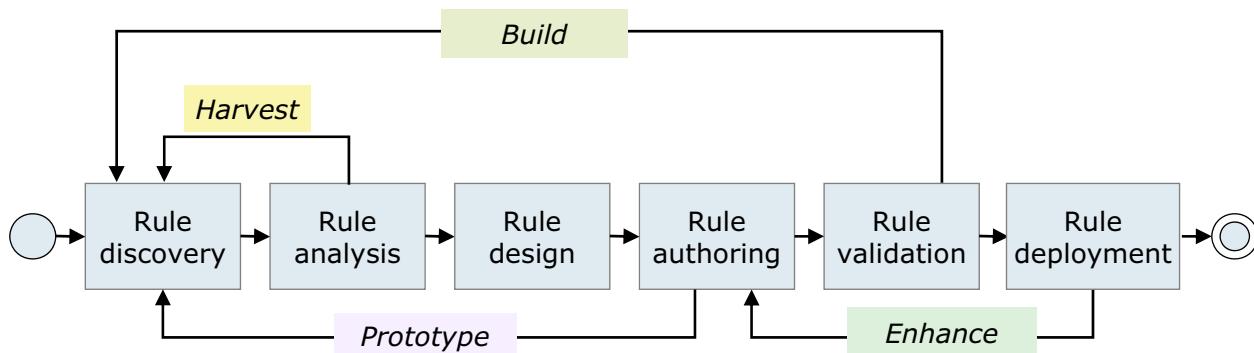
The group is formed from core team members who are involved in initial development of the application, including business analysts and policy managers, who can contribute to the governance objectives. Include members that represent various entities and facilitate communication between these entities 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
 - Recall graphic from unit 2



© Copyright IBM Corporation 2016

Figure 15-11. Governance and agile development

WB3951.2

Notes:

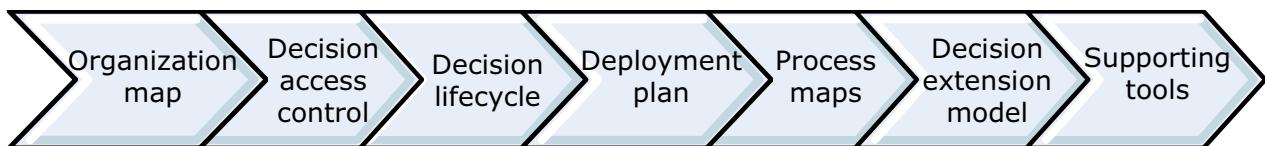
As you saw earlier in this course, the overall development of the decision follows an agile and iterative approach.

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



© Copyright IBM Corporation 2016

Figure 15-12. Implementing governance

WB3951.2

Notes:

- 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.

15.2. Operational Decision Manager support for governance

Operational Decision Manager support for governance



© Copyright IBM Corporation 2016

Course materials may not be reproduced in whole or in part without the prior written permission of IBM.

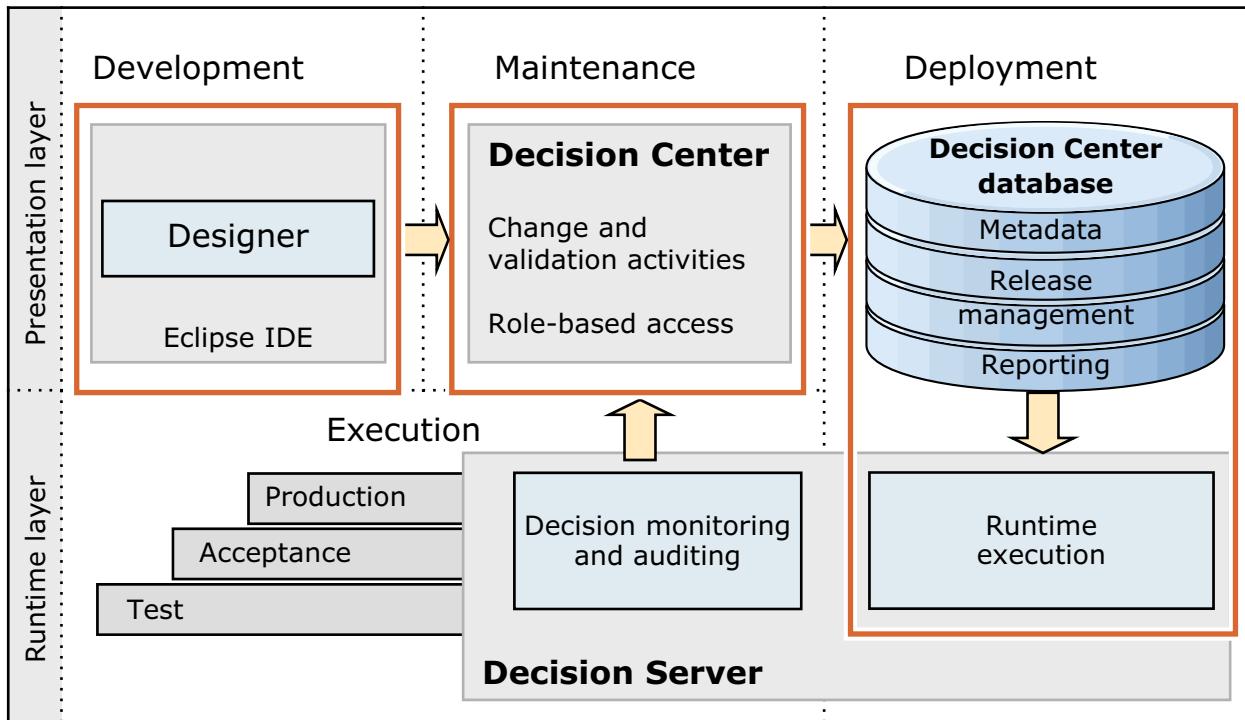
10.1

Figure 15-13. Operational Decision Manager support for governance

WB3951.2

Notes:

Decision lifecycle in Operational Decision Manager (1 of 2)



© Copyright IBM Corporation 2016

Figure 15-14. Decision lifecycle in Operational Decision Manager (1 of 2)

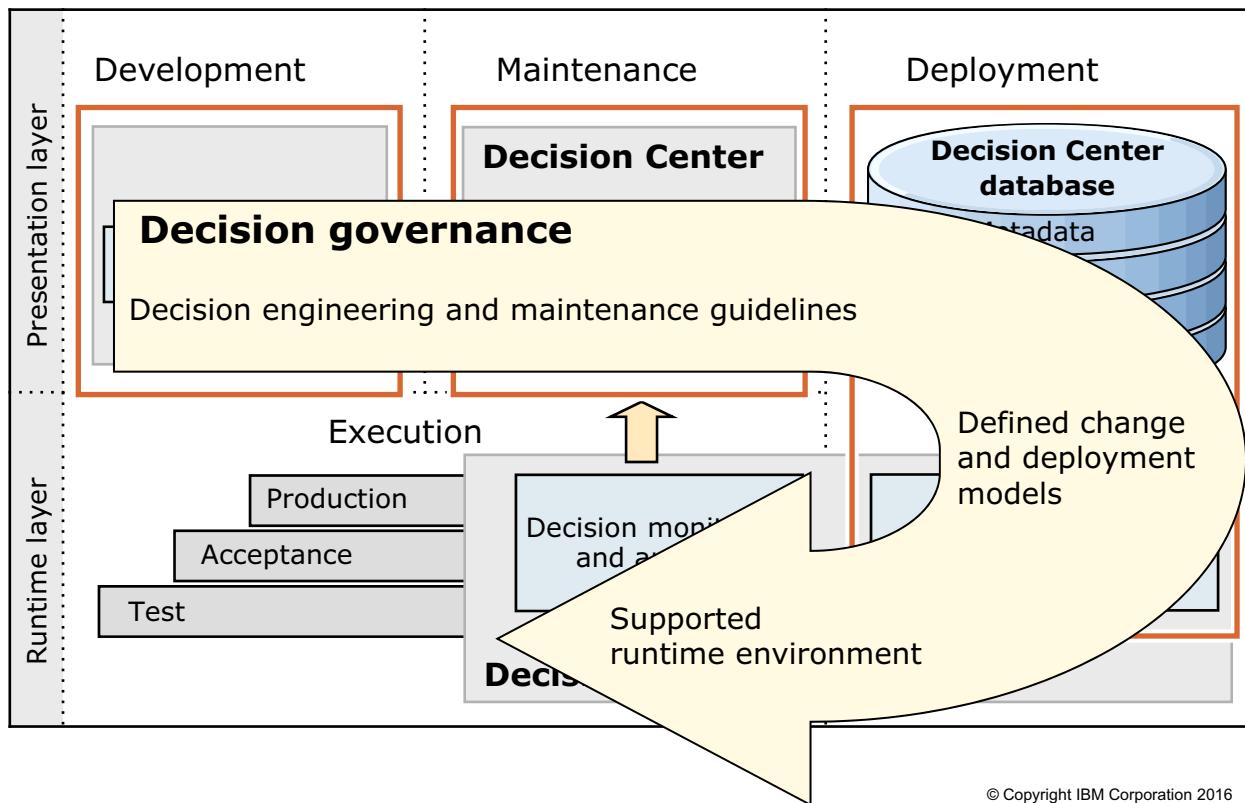
WB3951.2

Notes:

As seen throughout the course, ODM provides a set of mechanisms and tools to facilitate change while enforcing governance at the level of the individual decision artifacts and at the level of the overall decision.

- Developers can work in a single Eclipse-based environment.
- Decision Center provides a complete operational decision management environment for business users. You can easily manage the level of access and control to enable various stakeholders to author, edit, transition, and deploy rules that are stored in the Decision Center database.
- The Decision Center database provides integrated storage for rule and event artifacts as a central “source of truth.” A rich metadata layer is provided for decision logic that specifies all the custom properties that define how rules are used and how they interact with other rules in generating decisions.
- Decision Server provides a centralized execution environment to support streamlined change deployments.

Decision lifecycle in Operational Decision Manager (2 of 2)



© Copyright IBM Corporation 2016

Figure 15-15. Decision lifecycle in Operational Decision Manager (2 of 2)

WB3951.2

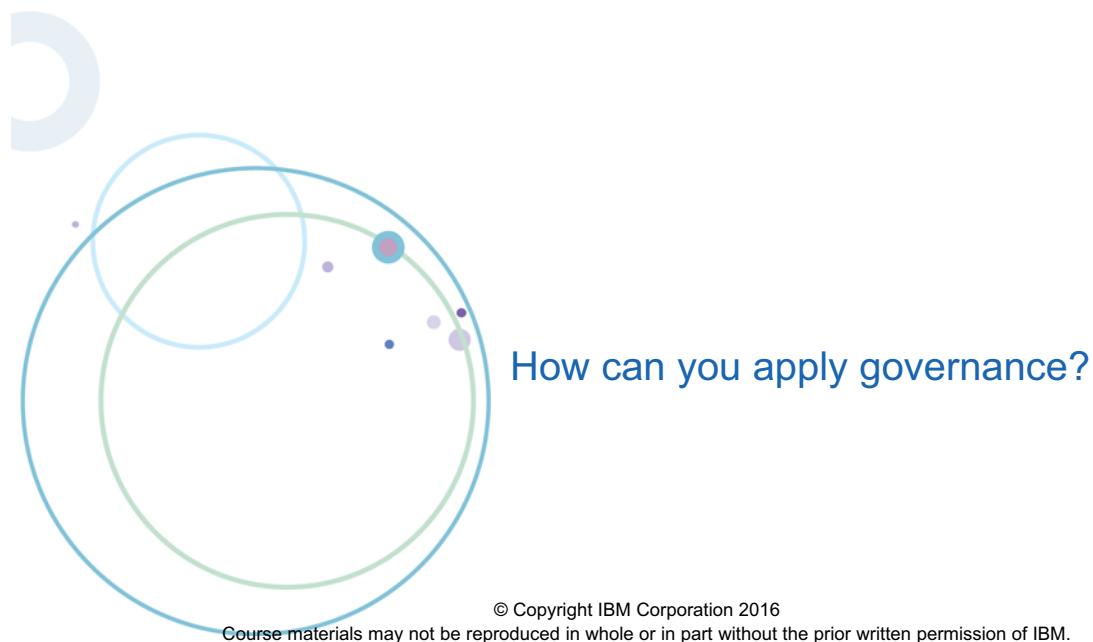
Notes:

Coupled with the tools and deployment cycle that are depicted here, the suggested practices that are outlined in ABRD methodology ensure project success.

The lifecycle for rule and event artifacts is implemented through status management, access control, and permissions. Notice that this fine-grained level of governance must be defined carefully so that it does not become a hindrance to the change process.

Governance at the artifact level, while enforcing a division of responsibilities for the artifact authoring task, does not provide a vision of what happens at the decision level to manage change. Therefore, on top of artifact-level governance, a decision management solution must define the change lifecycle and governance at the level of the decision itself. The goal is to define who is responsible for which task, at what phase, and in which environment. The process defines change management of a business policy update, from its initial request by the policy manager to its deployment in the production environment, through to evaluation of the newly deployed business decision implementation. Decision changes follow a cycle of *define*, *deploy*, *measure*, and *update*.

Discussion



© Copyright IBM Corporation 2016

Course materials may not be reproduced in whole or in part without the prior written permission of IBM.

10.1

Figure 15-16. Discussion

WB3951.2

Notes:

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?

15.3. Decision governance framework

Decision governance framework



© Copyright IBM Corporation 2016

Course materials may not be reproduced in whole or in part without the prior written permission of IBM.

10.1

Figure 15-17. Decision governance framework

WB3951.2

Notes:

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
 - Approve change and validation activities
 - Approve releases for deployment
 - Deploy completed releases to production
- Decision Center Enterprise console
 - Perform validation tasks (simulations)

© Copyright IBM Corporation 2016

Figure 15-18. Decision governance framework overview

WB3951.2

Notes:

The decision governance framework uses tools in Rule Designer and Decision Center to provide a ready-to-use, prescriptive approach to change management and rule governance. It helps business users manage, report, and govern changes to rules and decisions.

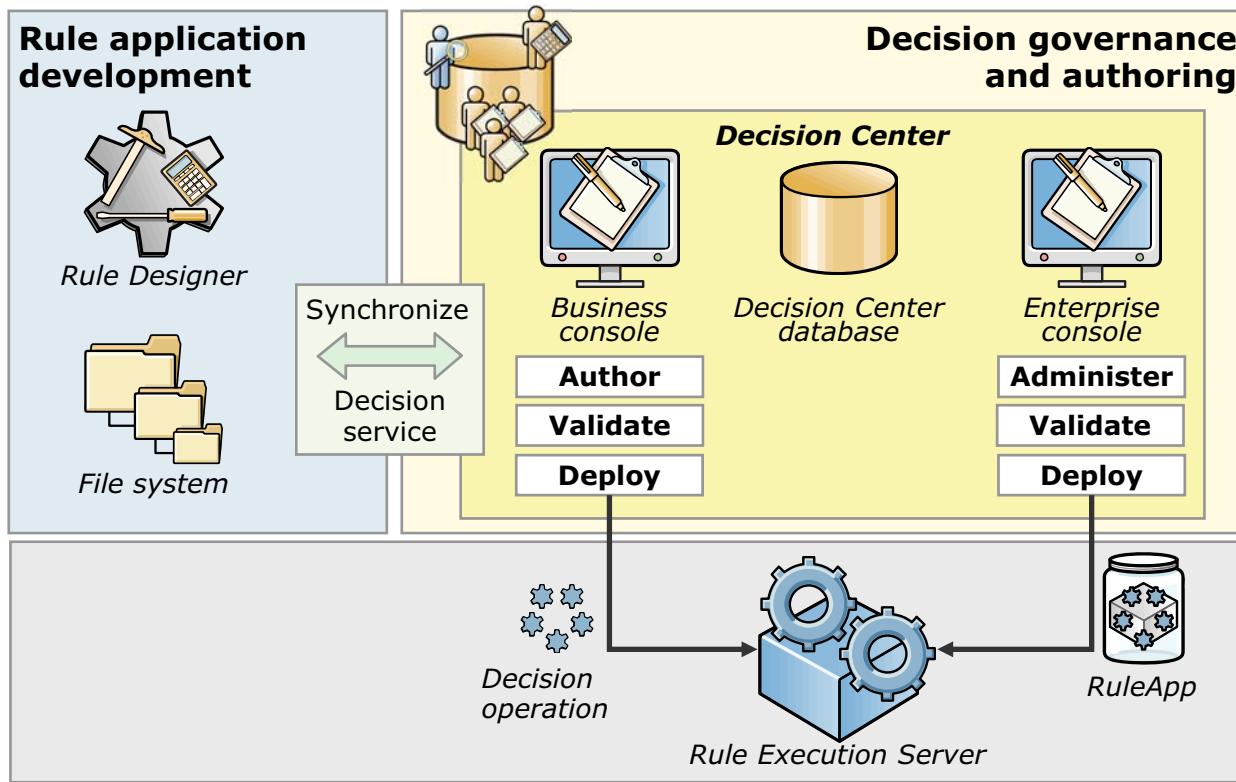
The decision governance framework is based on decision service *releases*, which follow a well-defined change process that is supported by Decision Center tools. Tasks that business users complete depend on user roles, such as rule author or tester.

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

Governance framework includes:

- Decision service releases
- Change activities
- Validation activities (testing)
- Deployment of releases

Recall: Operational Decision Manager tools



© Copyright IBM Corporation 2016

Figure 15-19. Recall: Operational Decision Manager tools

WB3951.2

Notes:

The decision governance framework in Operational Decision Manager encompasses the following tasks:

Synchronizing

- IT users use Rule Designer to publish a set of rule projects as a decision service that business users can access through Business console

Authoring

- Business analysts and rule authors use Business console to create *change activities* and author rules within a release

Validating

- Policy managers use Business console to create *validation activities* to track and manage test plans for a release
- Assigned testers create and run tests and simulations in Business console

Deploying

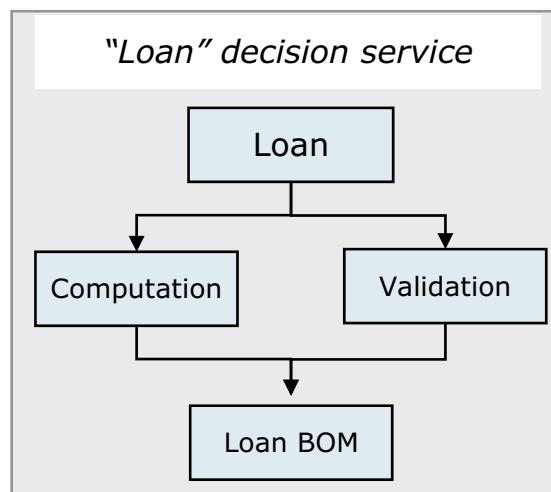
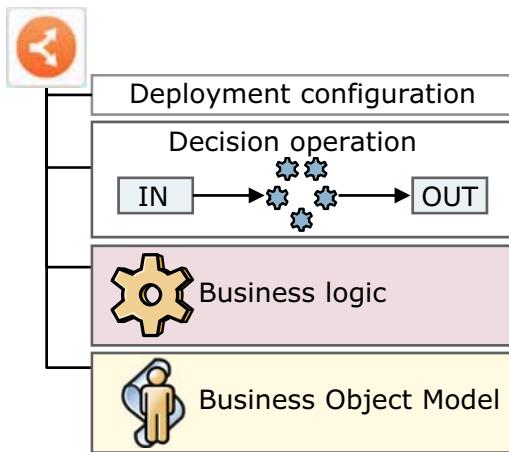
- After all change and validation activities are complete and approved, policy managers approve the completion of the release
- Release is ready for deployment
- Release owners or administrators can create deployment configurations and deploy decision services from Business console to Rule Execution Server

Administering

- Manage roles and permissions for Business console users
- Configure the Decision Center database, which provides snapshot and version features that support the auditing and rollback of business rules

Decision service (1 of 2)

- Simplified unit of work for change management and rule governance
 - Contains one or more rule projects
 - Name of decision service corresponds to its top-level project



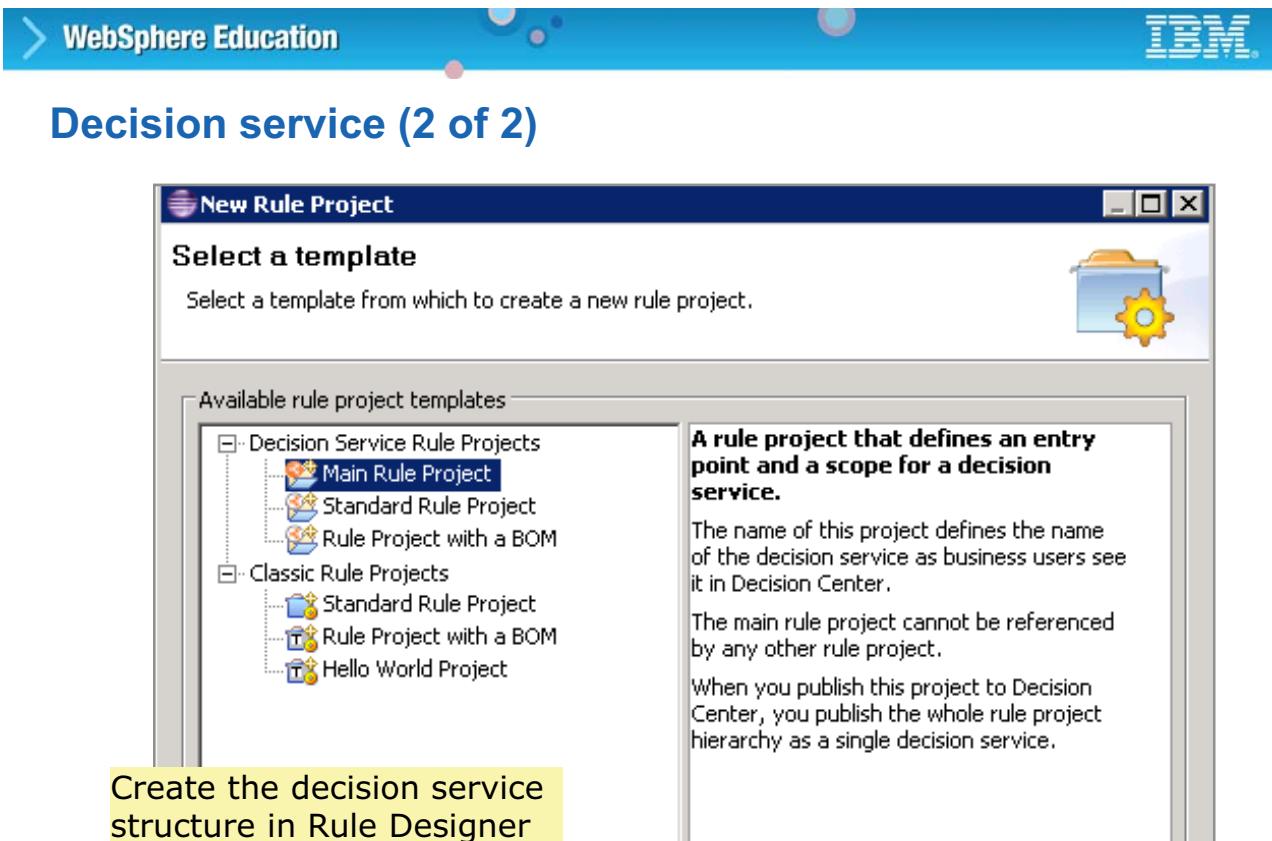
- Contains artifacts to support governance from authoring to testing and deployment, including:
 - Decision operation
 - Deployment configuration

© Copyright IBM Corporation 2016

Figure 15-20. Decision service (1 of 2)

WB3951.2

Notes:



© Copyright IBM Corporation 2016

Figure 15-21. Decision service (2 of 2)

WB3951.2

Notes:



Decision service properties

- Use the rule project properties dialog box to define decision service properties
 - Designate top-level main rule project of the decision service

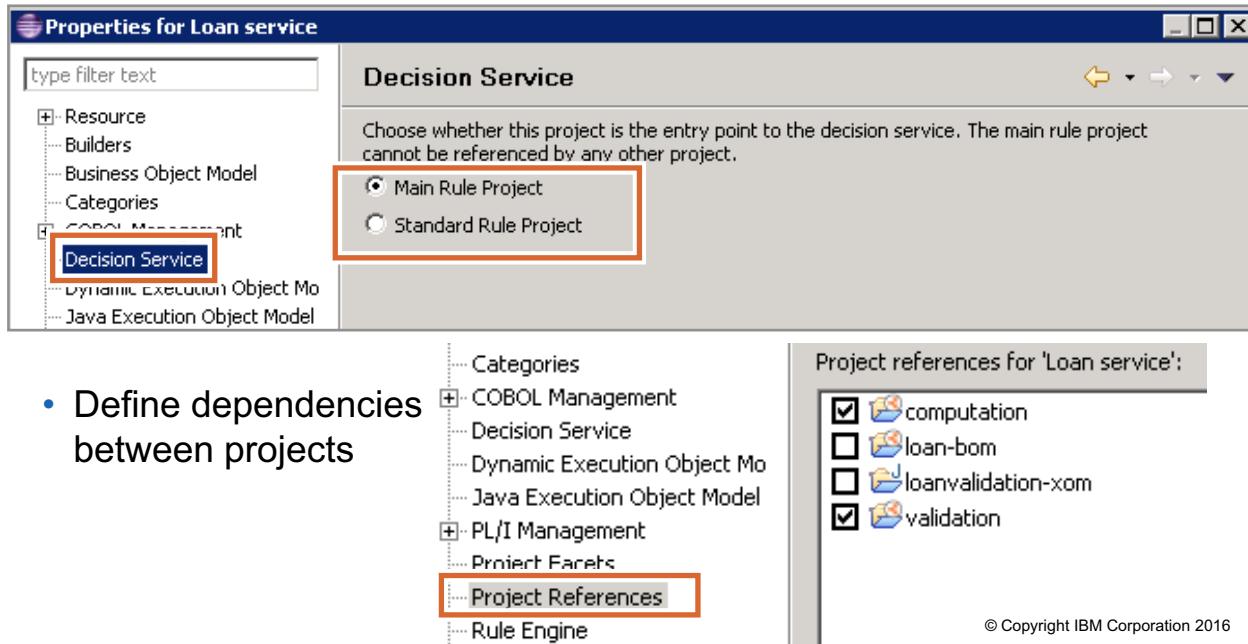


Figure 15-22. Decision service properties

WB3951.2

Notes:



Decision operations (1 of 2)

- Create a decision operation to define the ruleset to use, the ruleset parameters, and the ruleflow
 - Stored in the `deployment` folder of the project

Decision Operation Overview – myDecisionServiceOperation

General
Signature

Name: myDecisionServiceOperation
Description:

Signature
 Define the input and output parameters for the ruleset.
 Input: borrower
 Input - output: loan
 Output: none

Ruleset Name
 Enter the name of the ruleset part of the ruleset path that is called by Rule Execution Server.
 Ruleset name: myDecisionServiceRuleset

Ruleflow
 Define the main ruleflow that is used as the entry point for execution.
 Use main ruleflow: miniloan
 Do not use a ruleflow.

Overview
Signature
Source

© Copyright IBM Corporation 2016

Figure 15-23. Decision operations (1 of 2)

WB3951.2

Notes:

The ruleset is contained in a decision operation that includes the ruleset parameters.



Decision operations (2 of 2)

- Ruleset parameters are defined on the **Signature** page

Decision Operation Signature - myDecisionServiceOperation

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

Refresh Add as ruleset parameter

myDecisionService

- myDecisionServiceParameters
 - borrower
 - loan

Input Parameters
Define the parameters required to call the execution.

Parameter name	Verbalization	Type
borrower	the borrower	miniloan.Borrower

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

Parameter name	Verbalization	Type
loan	the loan	miniloan.Loan

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

Parameter name	Verbalization	Type

Overview Signature Source

© Copyright IBM Corporation 2016

Figure 15-24. Decision operations (2 of 2)

WB3951.2

Notes:

The decision operation can use any ruleset parameters or ruleset variables that are defined in any project within the decision service.

Run configurations for a decision operation

- To run a decision operation, create a decision operation configuration

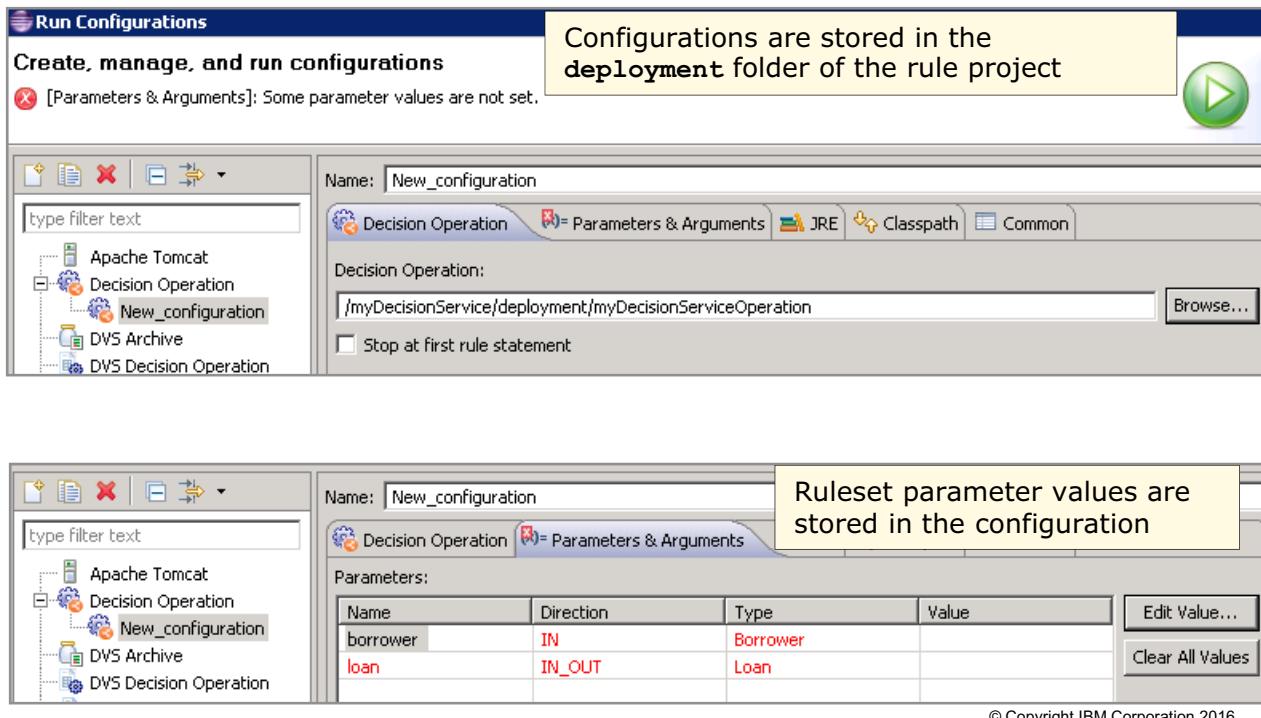


Figure 15-25. Run configurations for a decision operation

WB3951.2

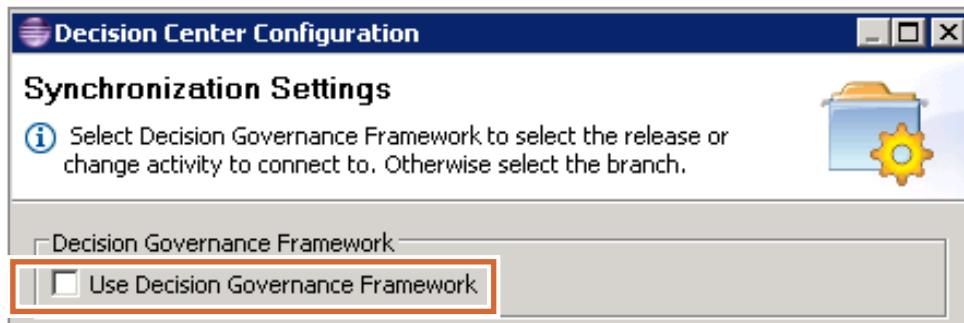
Notes:

After you create a decision operation, you create a deployment configuration. The deployment configuration can contain more than one decision operation.



Publishing decision services from Rule Designer

- When you publish a decision service to Decision Center, connect to Decision Center from the top-level main rule project
- Select **Use Decision Governance Framework** check box



© Copyright IBM Corporation 2016

Figure 15-26. Publishing decision services from Rule Designer

WB3951.2

Notes:

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, such as running test suites and simulations
- Deploy new decision service releases

© Copyright IBM Corporation 2016

Figure 15-27. Decision Center: Governance in Business console

WB3951.2

Notes:

Through Decision Center, business users access decision insight capabilities that help them understand how and where change affects decisions within operational systems. They can also simulate how decisions affect the outcomes of transactions, orders, customer interactions, or processes.

- Role-based access control supports concurrent access of decision artifacts while enforcing security

Decision Center provides several lifecycle control features:

- Permission management
 - User roles and associated permissions determine the ability to create, change, or see rules
- Version control and history
 - Every version is kept in the database (even after deletion)
 - Ensures traceability
- Workflow control through rule status properties

- Depending on permissions, users can change the status property as the rule passes through the lifecycle phases
- Deployment to Rule Execution Server
- 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

© Copyright IBM Corporation 2016

Figure 15-28. States and user roles

WB3951.2

Notes:

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

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

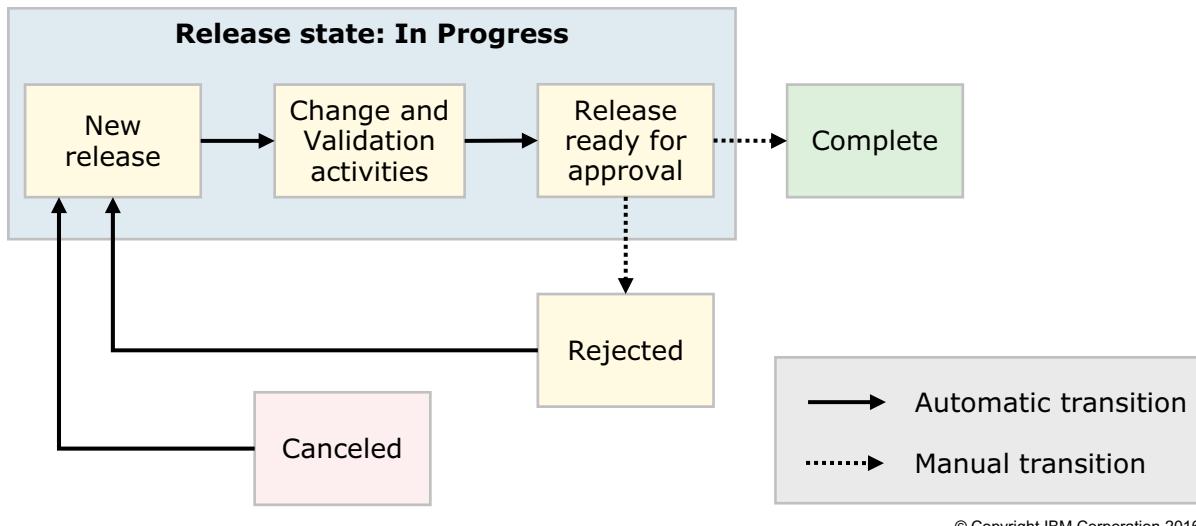


Figure 15-29. Release governance

WB3951.2

Notes:

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.

- However, some release-related tasks are manual

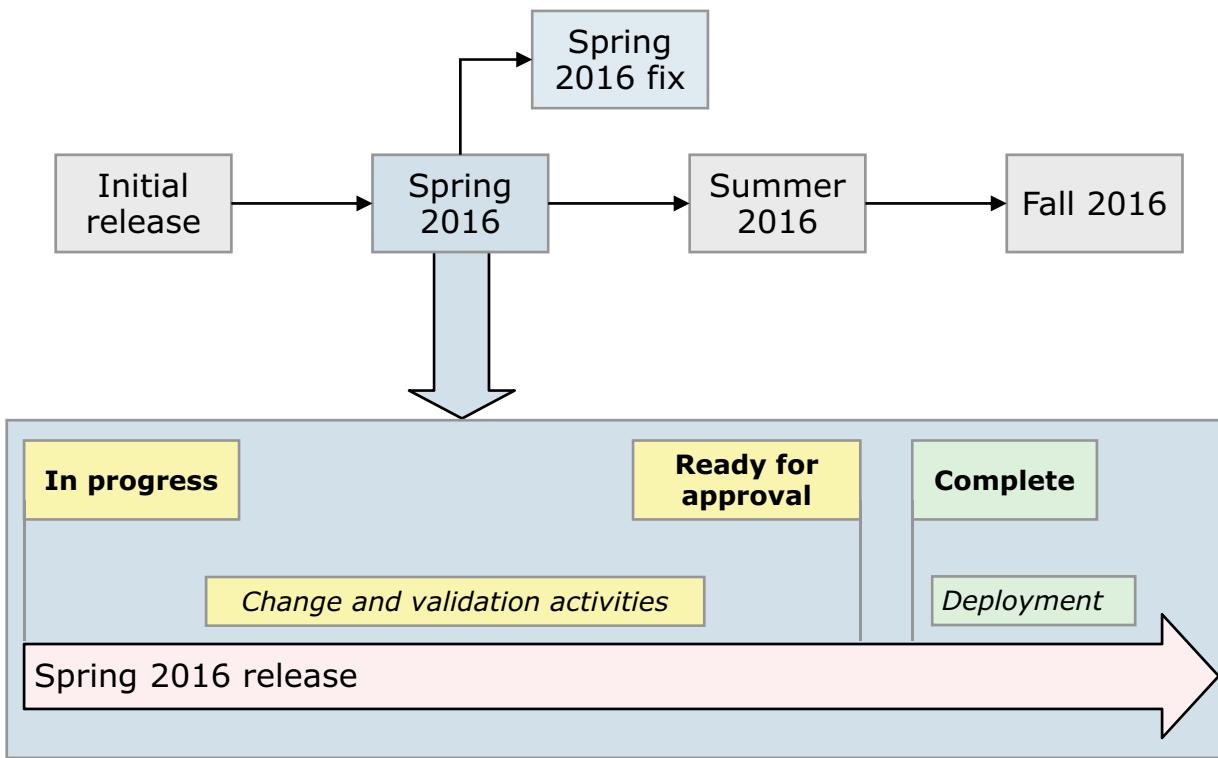
When the release is in the **In Progress** state, the owner can:

- Change the owner of the release
- Change the goals of the release
- Change the due date of the release
- Create change and validation activities

After all release activities are complete:

- The release owner changes the state of the release to **Ready for Approval**
- Approvers can then approve or reject the changes to the release

Release sequence



© Copyright IBM Corporation 2016

Figure 15-30. Release sequence

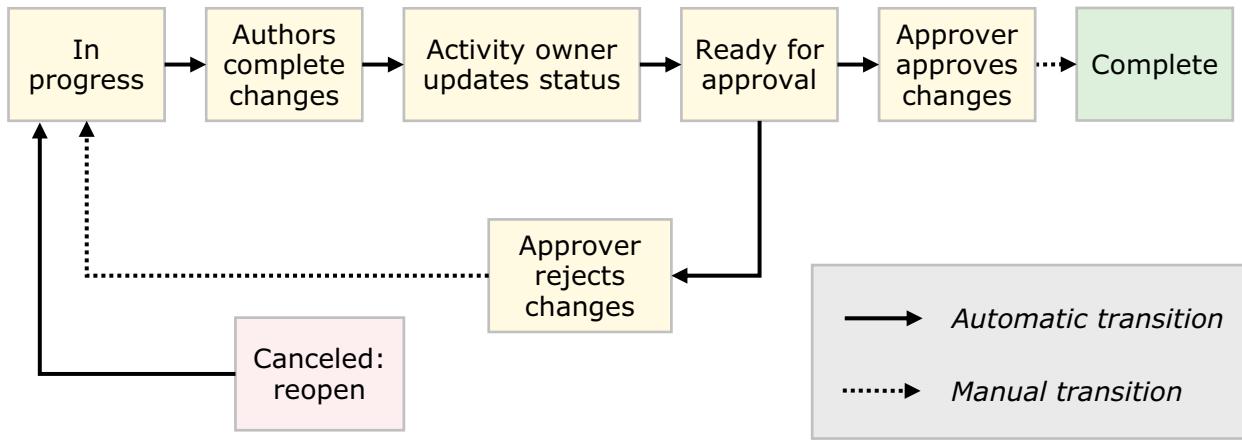
WB3951.2

Notes:

An initial release is automatically created when a decision service is published from Rule Designer. The content of the new release is based on the content of another complete release.

Change activity governance

- Changes to the release are initiated through the creation and completion of change activities
 - When a change activity is created, Decision Center takes a snapshot to record the starting state of the activity
 - Change activities also have states and transitions that are predefined



© Copyright IBM Corporation 2016

Figure 15-31. Change activity governance

WB3951.2

Notes:

Rule authors are responsible for editing and updating the rules so that they align with the goals of the release. After rule authors finish their work, they can change *their* status to **Finished**.

When all the authors finish their work, the owner of the change activity sets the state of the change activity to **Ready for Approval**.

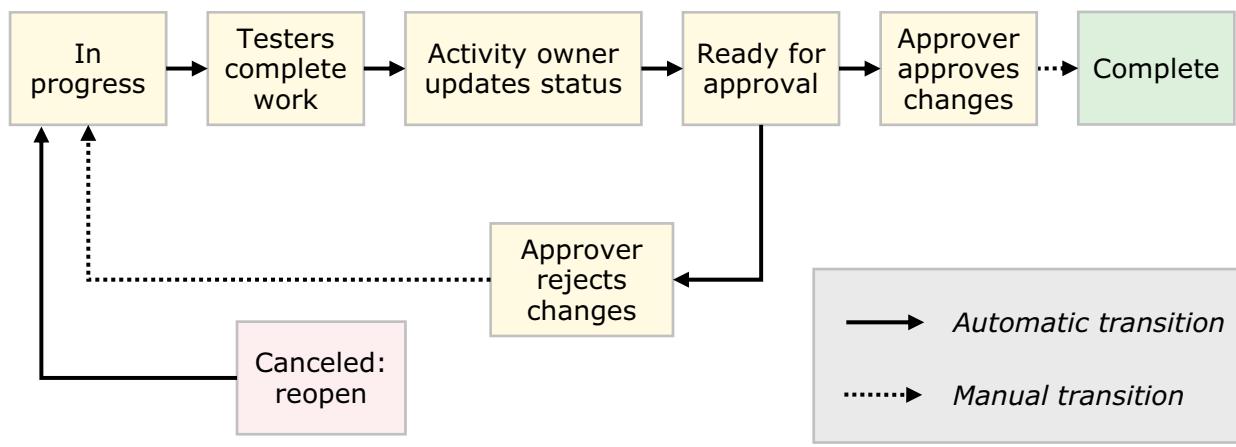
Then, the assigned approvers approve or reject the changes. If the change activity was not assigned approvers by the release owner, the change activity is automatically approved.

After all approvers approve the change activity, Decision Center takes another snapshot to record the state of the rules. Decision Center then merges the changes back into the release. The change activity is then marked **Complete**.

If a change activity is rejected, it returns to the state of **In Progress**.

Validation activity governance

- After change activities are completed, the release is validated through the creation and completion of validation activities
 - Policy managers create validation activities to track and manage test plans and test results for release to production
 - Validation tasks, such as running test suites and simulations, are completed in either the Business console or the Enterprise console



© Copyright IBM Corporation 2016

Figure 15-32. Validation activity governance

WB3951.2

Notes:

Testers run different tests and simulations that are aimed at validating a release, and note the result in the test plan. After testers complete their work, and all the change activities of the release are **Complete**, testers change their status to **Finished**.

The owner of the validation activity sets the validation activity state to **Ready for Approval**, at which point the approvers approve or reject the activity.

After all the approvers approve the validation activity, Decision Center sets the state of the validation activity to **Complete**.



Release deployment

- After all activities in a release are complete, the release is approved by the release owner
- Approved releases can be deployed from Business console

© Copyright IBM Corporation 2016

Figure 15-33. Release deployment

WB3951.2

Notes:

The **Deployments** tab lists the deployment configurations that are available for the decision service. Changes from change activities can be deployed to non-production servers. However, only completed releases can be deployed to a production environment.

When the decision service is deployed, a deployment snapshot is created. Snapshots can be consulted, compared, and restored, but not edited.



Unit summary

Having completed this unit, you should be able to:

- 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 2016

Figure 15-34. Unit summary

WB3951.2

Notes:

Checkpoint 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 the following tasks? Select all that apply.
 - a. Defining expectations and assigning responsibilities
 - b. Establishing efficient collaboration between the business and IT teams
 - c. Selecting a group of productive developers to be in charge of business rule management and reporting their decisions to business stakeholders

3. **True or False:** Implementing decision governance is outside the scope of the IT department.

4. **True or False:** With the decision governance framework, business users cannot deploy decision services from Business console.

© Copyright IBM Corporation 2016

Figure 15-35. Checkpoint questions

WB3951.2

Notes:

Write your answers here:

- 1.

- 2.

- 3.

- 4.



Checkpoint answers

1. **True.**
2. Governance encompasses which of the following tasks?
 - a. **True.**
 - b. **True.**
 - c. **False:** *Include business and developer stakeholders to ensure balanced representation for decision making.*
3. **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. **False:** *Business users with administrative permissions can create or edit deployment configurations and deploy decision services from Business console.*

© Copyright IBM Corporation 2016

Figure 15-36. Checkpoint answers

WB3951.2

Notes:

Unit 16. Course summary

What this unit is about

This unit summarizes the course and provides information for future study.

What you should be able to do

- Explain how the course met its learning objectives
- Access the IBM Training website
- Identify other IBM Training courses that are related to this topic
- Locate appropriate resources for further study

Unit objectives

After completing this unit, you should be able to:

- Explain how the course met its learning objectives
- Access the IBM Training website
- Identify other IBM Training courses that are related to this topic
- Locate appropriate resources for further study

© Copyright IBM Corporation 2016

Figure 16-1. Unit objectives

WB3951.2

Notes:

Course learning objectives (1 of 2)

After completing this course, you should be able to:

- Describe the benefits of implementing a decision management solution with Operational Decision Manager
- Identify the key user roles that are involved in designing and developing a decision management solution, and the tasks that are associated with each role
- Describe the development process of building a business rule application and the collaboration between business and development teams
- Set up and customize the business object model (BOM) and vocabulary for rule authoring
- Implement the execution object model (XOM) that enables rule execution
- Orchestrate rule execution through ruleflows
- Author rule artifacts to implement business policies

© Copyright IBM Corporation 2016

Figure 16-2. Course learning objectives (1 of 2)

WB3951.2

Notes:

Course learning objectives (2 of 2)

- Debug business rule applications to ensure that the implemented business logic is error-free
- Set up the testing and simulation environment for business users
- Package and deploy decision services to testing and production environments
- Integrate decision services for managed execution within an enterprise environment
- Build client applications to invoke ruleset execution
- Test, monitor, and audit execution of decision services
- Work with Operational Decision Manager features that support decision governance, including the decision governance framework for decision services

© Copyright IBM Corporation 2016

Figure 16-3. Course learning objectives (2 of 2)

WB3951.2

Notes:



To learn more on the subject

- IBM Training website:
www.ibm.com/training
- For more information about Decision Server Insights and decision management
 - Redpaper #TIPS-0940 IBM Operational Decision Manager: Enabling the Rule Engine to Augment the Decision Data:
www.redbooks.ibm.com/Redbooks.nsf/RedbookAbstracts/redp5333.html
 - Redbooks #SG24-8127-00 Governing Operational Decisions in an Enterprise Scalable Way:
www.redbooks.ibm.com/abstracts/sg248127.html
 - Redbooks #REDP-4836-00 Making Better Decisions Using IBM WebSphere Operational Decision Management:
www.redbooks.ibm.com/abstracts/redp4836.html

© Copyright IBM Corporation 2016

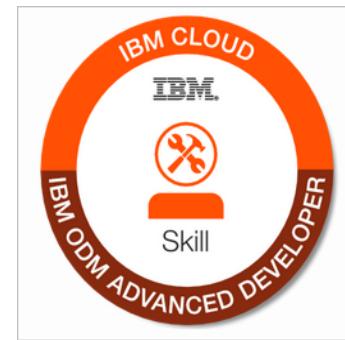
Figure 16-4. To learn more on the subject

WB3951.2

Notes:

Earn an IBM Badge

- After completing this course, you might be ready to take an IBM Badge test
- Use IBM Badges to share verified proof of your IBM credentials
- Find your Badge test on this site:
 - www.ibm.com/services/learning/ites.wss/zz-en?pageType=badgesearch
- The *IBM Operational Decision Manager Advanced V8.8 Developer Badge* test requires these courses:
 - **WB395/ZB395: Developing Rule Solutions with IBM Operational Decision Manager V8.8**
 - **WB399/ZB399: Developing Solutions with IBM Decision Server Insights V8.8**
 - After completing **both** courses, take the Badge test



© Copyright IBM Corporation 2016

Figure 16-5. Earn an IBM Badge

WB3951.2

Notes:

Enhance your learning with IBM resources

Keep your IBM Cloud skills up-to-date

- IBM offers resources for:
 - Product information
 - Training and certification
 - Documentation
 - Support
 - Technical information



- To learn more, see the IBM Cloud Education Resource Guide:
 - www.ibm.biz/CloudEduResources

© Copyright IBM Corporation 2016

Figure 16-6. Enhance your learning with IBM resources

WB3951.2

Notes:



Unit summary

Having completed this unit, you should be able to:

- Explain how the course met its learning objectives
- Access the IBM Training website
- Identify other IBM Training courses that are related to this topic
- Locate appropriate resources for further study

© Copyright IBM Corporation 2016

Figure 16-7. Unit summary

WB3951.2

Notes:

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
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
IDE	integrated development environment

IP	Internet Protocol
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
MTDS	monitored transparent decision service
ODM	Operational Decision Manager
ODM	Optimization Decision Manager (ILOG)
POJO	plain old Java object
POS	Point of sale
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

RSO	Rule Solutions for Office
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
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
z/OS	Z Series Operating System

Appendix B. Appendix: IBM ODM on Cloud

What this unit is about

This unit describes Operational Decision Manager on Cloud.

What you should be able to do

After completing this unit, you should be able to:

- Describe Operational Decision Manager on Cloud.



Introduction to IBM ODM on Cloud

- Enterprise-grade ODM cloud service for development, testing, and production
- Cloud-based, collaborative, and role-based environment
 - Capture, automate, and manage frequently occurring, repeatable rules-based business decisions
- Ready-to-use development, test, and production environments are available
- Monthly subscription plans
- Available exclusively on IBM Cloud infrastructure
- Managed by IBM
- Artifacts that are created with IBM ODM on Cloud are compatible with IBM ODM on-premises product

© Copyright IBM Corporation 2016

Figure B-1. Introduction to IBM ODM on Cloud

WB3951.2

Notes:



IBM ODM on Cloud

- Targets born-on-the-cloud projects and hybrid cloud scenarios
- Born-on-the-cloud project:
 - Development, testing, and production in the cloud
 - Prefer cloud solutions to on-premises solutions
 - Urgency for implementation
- Pilot project
 - Proving business value
 - Try a newer version of IBM Operational Decision Manager
- Development
 - Organization can manage the production solution on-premises, but needs to get started quickly to meet go-live dates

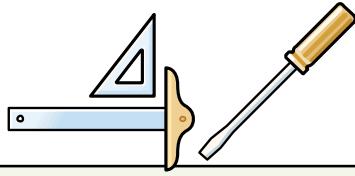
© Copyright IBM Corporation 2016

Figure B-2. IBM ODM on Cloud

WB3951.2

Notes:

Accelerate decision management solution deployment with IBM ODM on Cloud



Build-it-yourself

- Separately procure and install software, service, and hardware
- Sign multiple contracts
- Requires capital investment to procure software, hardware, and implementation services
- Ongoing management, maintenance, and upgrades



IBM ODM on Cloud

- Integrated, fast, and flexible
- Get started right away
- One business solution, one contract, and one subscription price
- Minimal capital investment
- Scale when needed
- Reduce time and effort that is needed for maintenance
- Keep up-to-date with the latest releases

© Copyright IBM Corporation 2016

Figure B-3. Accelerate decision management solution deployment with IBM ODM on Cloud

WB3951.2

Notes:

ODM on Cloud customer focus: Manage and automate decisions

IBM manages:

- Uptime
- Monitoring
- Backup
- High availability
- Disaster recovery
- Updates
- Maintenance



Customers manage:

- Application development
- Application integration
- Application support



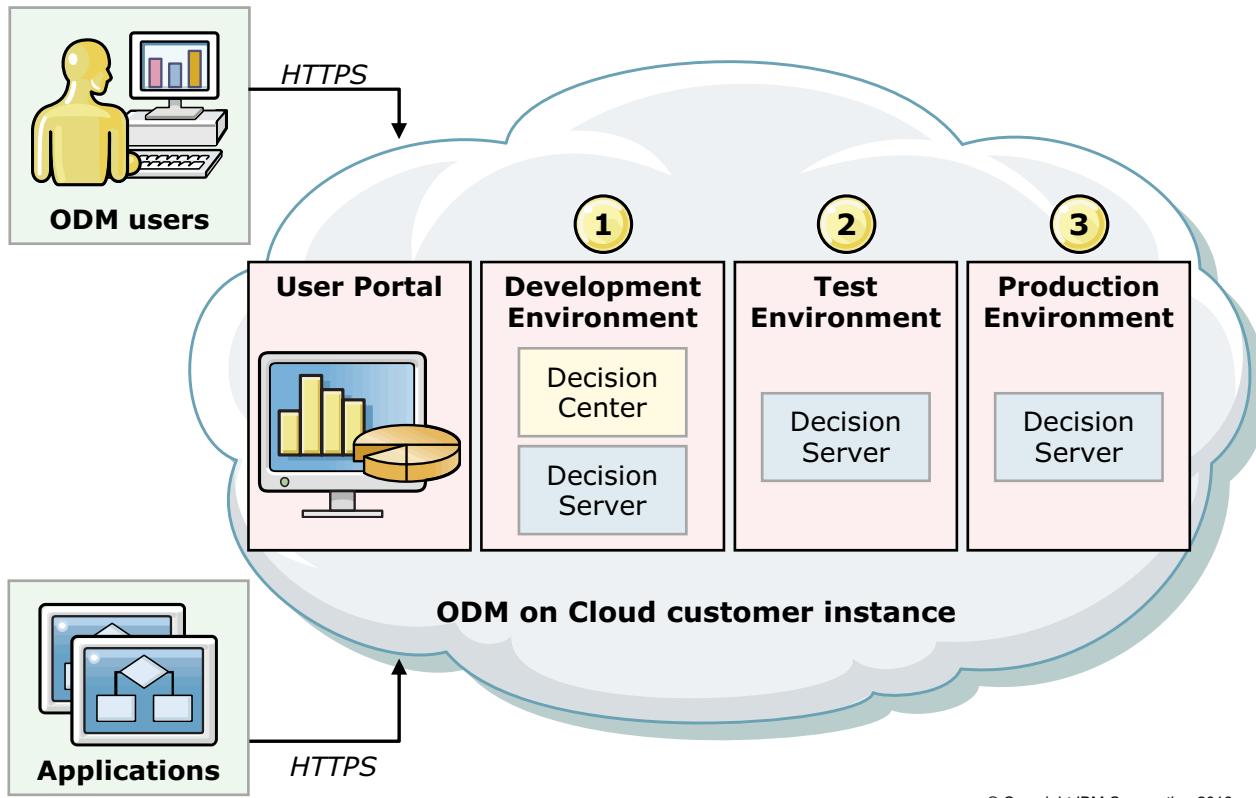
© Copyright IBM Corporation 2016

Figure B-4. ODM on Cloud customer focus: Manage and automate decisions

WB3951.2

Notes:

IBM ODM on Cloud: Three runtime environments



© Copyright IBM Corporation 2016

Figure B-5. IBM ODM on Cloud: Three runtime environments

WB3951.2

Notes:

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

1. Development
2. Test
3. Production

In this diagram:

- **ODM users** include developers, business analysts, business users, and rule authors who access Rule Designer, Decision Center, and the various user consoles.
- **Applications** are applications that call deployed decision services.

Workflow

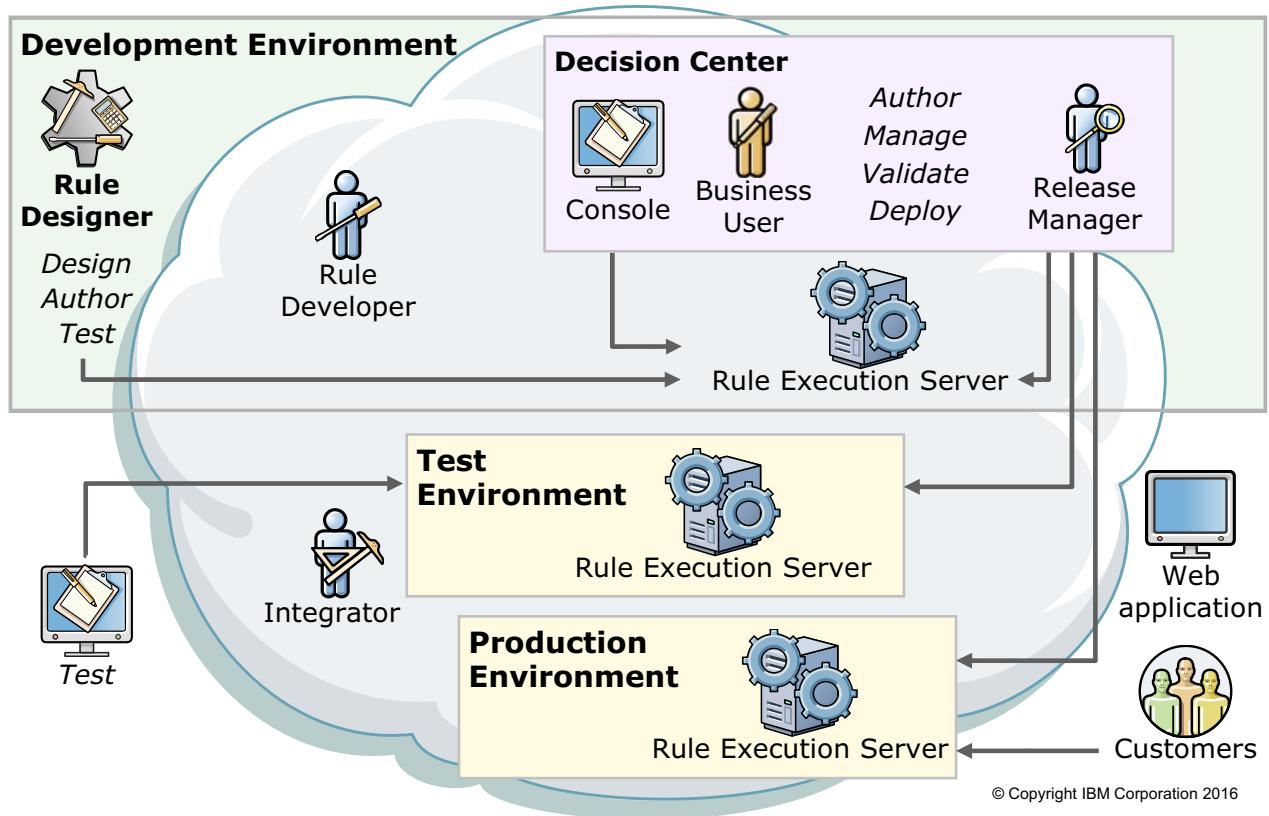


Figure B-6. Workflow

WB3951.2

Notes:

This diagram illustrates how the predefined user roles in IBM ODM on Cloud interact with the product components during the lifecycle of a business rules application.



IBM ODM on Cloud free trial

- Free trial for IBM ODM on Cloud is available
- Go to the following website and click **Try for free** to sign up:
www.bpm.ibmcloud.com/odm

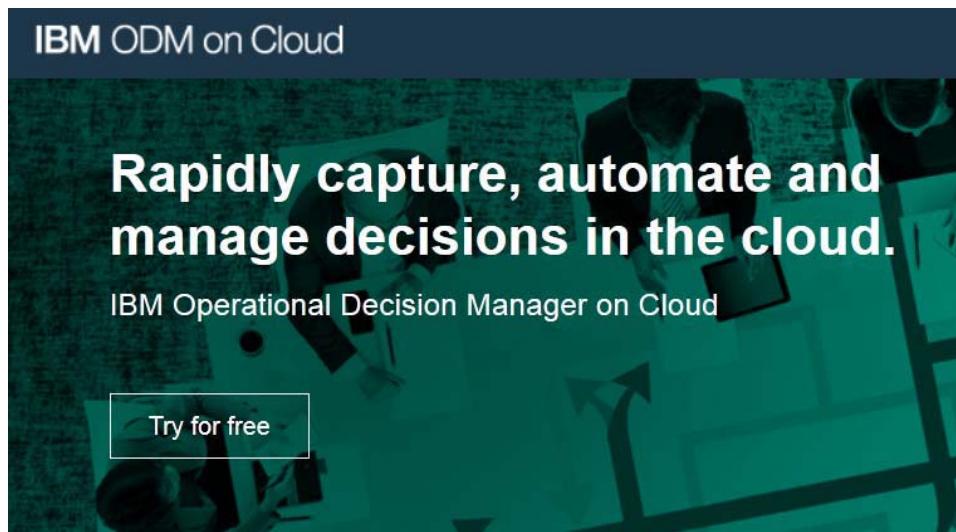


Figure B-7. IBM ODM on Cloud free trial

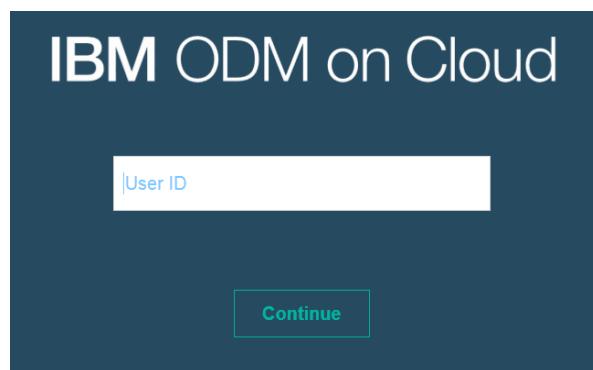
WB3951.2

Notes:



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



© Copyright IBM Corporation 2016

Figure B-8. Activating access and logging in to IBM ODM on Cloud

WB3951.2

Notes:

The screenshot shows the IBM ODM on Cloud user portal interface. At the top, there's a navigation bar with 'WebSphere Education' and the IBM logo. Below the navigation bar, the title 'IBM ODM on Cloud user portal' is displayed. The main content area is organized into three horizontal sections: 'Development Environment', 'Test Environment', and 'Production Environment'. Each section contains links to different applications:

- Development Environment:**
 - Rule Designer: Build a decision service to author and run the business rules that implement your business decisions. Includes 'Download' and 'More info' buttons.
 - Decision Center Business console: Author, test, govern and deploy decision services. Includes 'Launch' and 'More info' buttons.
 - Decision Center Enterprise console: Perform occasional administration or advanced management activities. Includes 'Launch' and 'More info' buttons.
 - Rule Execution Server console: Manage and monitor decision services deployed from Rule Designer or Decision Center as part of your development activities. Includes 'Launch' and 'More info' buttons.
- Test Environment:**
 - Rule Execution Server console: Monitor decision services deployed from Decision Center used for performance, system, and integration testing activities. Includes 'Launch' and 'More info' buttons.
- Production Environment:**
 - Rule Execution Server console: Monitor decision services deployed from Decision Center used in the context of a production application. Includes 'Launch' and 'More info' buttons.

At the bottom right of the portal interface, there is a copyright notice: © Copyright IBM Corporation 2016.

Figure B-9. IBM ODM on Cloud user portal

WB3951.2

Notes:

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`

Development Environment

Rule Designer



Build a decision service to author and run the business rules that implement your business decisions.

[!\[\]\(8923e61ddad490193ef183620621fd77_img.jpg\) Download](#) [!\[\]\(810a77b0c4608d59ce27cc76488be201_img.jpg\) More info](#)

Figure B-10. Using Rule Designer

© Copyright IBM Corporation 2016

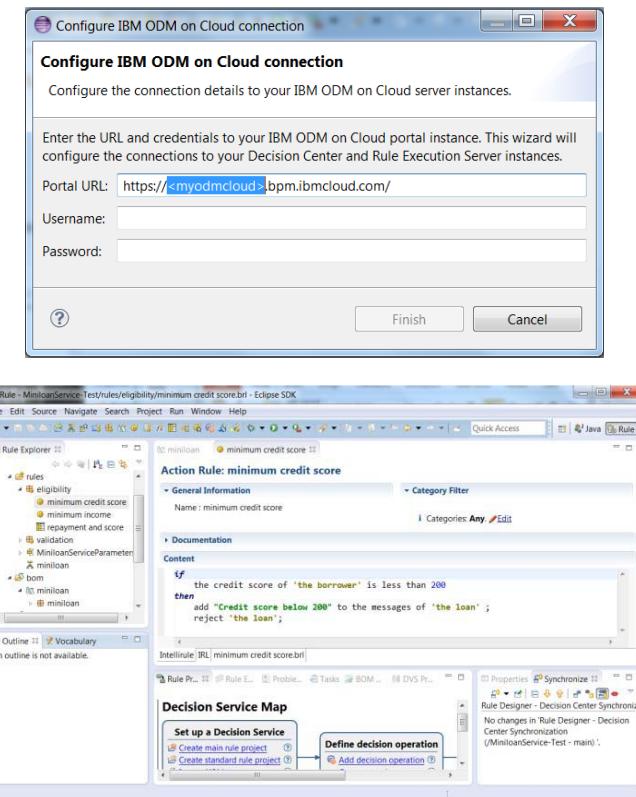
WB3951.2

Notes:



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



© Copyright IBM Corporation 2016

Figure B-11. Using Rule Designer (2 of 2)

WB3951.2

Notes:



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

Decision Center Business console  Author, test, govern and deploy decision services  Launch  More info	Decision Center Enterprise console  Perform occasional administration or advanced management activities.  Launch  More info	Rule Execution Server console  Manage and monitor decision services deployed from Rule Designer or Decision Center as part of your development activities.  Launch  More info
---	---	--

© Copyright IBM Corporation 2016

Figure B-12. Using Decision Center and Rule Execution Server (1 of 3)

WB3951.2

Notes:

The IBM ODM on Cloud modules can be started from the user portal.



Using Decision Center and Rule Execution Server (2 of 2)

- Log in using your ODM on Cloud user name and password
- Module opens in web browser window
 - Interface is the same as on-premises version

A screenshot of the Decision Center and Rule Execution Server interface. The top navigation bar includes 'Decision Center', 'HOME', 'LIBRARY', 'WORK', and a question mark icon. The main area is divided into sections: 'What's New' (containing 'New Rules' like 'maximum amount', 'repayment and score', 'minimum income', and 'minimum credit score'), 'Stream' (showing no new updates), and 'Followed Rules' (listing 'maximum amount (1)', 'minimum credit score (1)', 'minimum income (1)', and 'repayment and score (1)'). A sidebar on the right shows 'Rules Recently Worked On' with a note: 'You have not worked on any rules yet.' At the bottom right, a copyright notice reads '© Copyright IBM Corporation 2016'.

Figure B-13. Using Decision Center and Rule Execution Server (2 of 2)

WB3951.2

Notes:

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 on Cloud user portal also has direct links to the documentation

- IBM ODM Support Portal

https://www.ibm.com/support/entry/portal/product/websphere/ibm_operational_decision_manager

- Support Portal provides tools and resources for help with IBM Operational Decision Manager
- Open service requests, view fix lists, access community resources, and more

© Copyright IBM Corporation 2016

Figure B-14. Finding help for IBM ODM on Cloud

WB3951.2

Notes:

For more information about how to use IBM ODM on Cloud, see the IBM Knowledge Center for IBM ODM on Cloud.

The IBM ODM Support Portal provides general information on IBM Operational Decision Manager and access to other resources that you might find of interest. You can also use the Support Portal to open a service request.

Appendix C. Resource guide

Completing this WebSphere Education course is a great first step in building your WebSphere, CICS, and SOA skills. Beyond this course, IBM offers several resources to keep your WebSphere skills on the cutting edge. Resources available to you range from product documentation to support websites and social media websites.

Training

- **IBM Training website**
 - Bookmark the IBM Training website for easy access to the full listing of IBM training curricula. The website also features training paths to help you select your next course and available certifications.
 - For more information, see: <http://www.ibm.com/training>
- **IBM Training News**
 - Review or subscribe to updates from IBM and its training partners.
 - For more information, see: <http://bit.ly/IBMTrainEN>
- **IBM Certification**
 - You can demonstrate to your employer or clients your new WebSphere, CICS, or SOA mastery through achieving IBM Professional Certification. WebSphere certifications are available for developers, administrators, and business analysts.
 - For more information, see: <http://www.ibm.com/certify>
- **Training paths**
 - Find your next course easily with IBM training paths. Training paths provide a visual flow-chart style representation of training for many WebSphere products and roles, including developers and administrators.
 - For more information, see: <http://www.ibm.com/services/learning/sites.wss/us/en?pageType=page&c=a0003096>

Social media links

You can keep in sync with WebSphere Education, including new courses and certifications, course previews, and special offers, by visiting any of the following social media websites.

- **Twitter**
 - Receive short and concise updates from WebSphere Education a few times each week.
 - Follow WebSphere Education at: twitter.com/websphere_edu

- **Facebook:**

- Become a fan of IBM Training on Facebook to keep in sync with the latest news and career trends, and to post questions or comments.
- Find IBM Training at: facebook.com/ibmtraining

- **YouTube:**

- Visit the IBM Training YouTube channel to learn about IBM training programs and courses.
- Find IBM Training at: youtube.com/IBMTTraining

Support

- **WebSphere Support portal**

- The WebSphere Support website provides access to a portfolio of support tools. From the WebSphere Support website, you can access several downloads, including troubleshooting utilities, product updates, drivers, and Authorized Program Analysis Reports (APARs). To collaboratively solve issues, the support website is a clearing house of links to online WebSphere communities and forums. The IBM support website is now customizable so you can add and delete portlets to the information most important to the WebSphere products you work with.
- For more information, see: <http://www.ibm.com/software/websphere/support>

- **IBM Support Assistant**

- The IBM Support Assistant is a local serviceability workbench that makes it easier and faster for you to resolve software product issues. It includes a desktop search component that searches multiple IBM and non-IBM locations concurrently and returns the results in a single window, all within IBM Support Assistant.
- IBM Support Assistant includes a built-in capability to submit service requests; it automatically collects key problem information and transmits it directly to your IBM support representative.
- For more information, see: <http://www.ibm.com/software/support/isa>

- **WebSphere Education Assistant**

- IBM Education Assistant is a collection of multimedia modules that are designed to help you gain a basic understanding of IBM software products and use them more effectively. The presentations, demonstrations, and tutorials that are part of the IBM Education Assistant are an ideal refresher for what you learned in your WebSphere Education course.
- For more information, see:
<http://www.ibm.com/software/info/education/assistant/>

WebSphere documentation and tips

- **IBM Redbooks**
 - The IBM International Technical Support Organization develops and publishes IBM Redbooks publications. IBM Redbooks are downloadable PDF files that describe installation and implementation experiences, typical solution scenarios, and step-by-step “how-to” guidelines for many WebSphere products. Often, Redbooks include sample code and other support materials available as downloads from the site.
 - For more information, see: <http://www.ibm.com/redbooks>
- **IBM documentation and libraries**
 - Information centers and product libraries provide an online interface for finding technical information on a particular product, offering, or product solution. The information centers and libraries include various types of documentation, including white papers, podcasts, webcasts, release notes, evaluation guides, and other resources to help you plan, install, configure, use, tune, monitor, troubleshoot, and maintain WebSphere products. The WebSphere information center and library are located conveniently in the left navigation on WebSphere product web pages.
- **developerWorks**
 - IBM developerWorks is the web-based professional network and technical resource for millions of developers, IT professionals, and students worldwide. IBM developerWorks provides an extensive, easy-to-search technical library to help you get up to speed on the most critical technologies that affect your profession. Among its many resources, developerWorks includes how-to articles, tutorials, skill kits, trial code, demonstrations, and podcasts. In addition to the WebSphere zone, developerWorks also includes content areas for Java, SOA, web services, and XML.
 - For more information, see: <http://www.ibm.com/developerworks>

WebSphere Services

- IBM Software Services for WebSphere are a team of highly skilled consultants with broad architectural knowledge, deep technical skills, expertise on suggested practices, and close ties with IBM research and development labs. The WebSphere Services team offers skills transfer, implementation, migration, architecture, and design services, plus customized workshops. Through a worldwide network of services specialists, IBM Software Service for WebSphere makes it easy for you to design, build, test, and deploy solutions, helping you to become an on-demand business.
- For more information, see:
<http://www.ibm.com/developerworks/websphere/services/>

IBM
®