

Systems of Insight for Digital Transformation

Using IBM Operational Decision Manager Advanced
and Predictive Analytics

Whei-Jen Chen

Rajeev Kamath

Alexander Kelly

Hector H. Diaz Lopez

Matthew Roberts

Yee Pin Yheng



 **Analytics**

Big Data



International Technical Support Organization

**Systems of Insight for Digital Transformation: Using
IBM Operational Decision Manager Advanced
and Predictive Analytics**

December 2015

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

First Edition (December 2015)

This edition applies to IBM Operational Decision Manager Advanced Version 8.7.1.

© Copyright International Business Machines Corporation 2015. All rights reserved.

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

Contents

Notices	vii
Trademarks	viii
IBM Redbooks promotions	ix
Preface	xi
Authors	xi
Now you can become a published author, too!	xiii
Comments welcome	xiv
Stay connected to IBM Redbooks	xiv
Chapter 1. Introduction	1
1.1 Decision making	2
1.1.1 Systems of record	3
1.1.2 Systems of engagement	3
1.1.3 Systems of insight	4
1.1.4 Types of decisions	7
1.2 Motivation and business challenges	8
1.2.1 Decision-making process in general	8
1.3 Decision automation	9
1.3.1 Decision automation versus decision optimization	10
1.4 Business use cases	10
1.5 Example of system of insight provided in this book	12
Chapter 2. Systems of insight solutions	13
2.1 Real-time solutions	14
2.1.1 Event-driven situation detection	14
2.1.2 Decision automation	16
2.1.3 Business activity monitoring	17
2.1.4 Stream processing	18
2.1.5 Next best action	20
2.1.6 Advanced patterns	21
2.2 Retroactive solutions	21
2.2.1 Manual analytic investigations	21
2.2.2 Big Data analytics	22
2.3 Proactive solutions	23
2.3.1 Predictive analytics	23
2.3.2 Prescriptive analytics	24
Chapter 3. “A model for systems of insight	27
3.1 Systems of insight capabilities	28
3.2 Consume and collect	30
3.2.1 Consume	30
3.2.2 Collect	31
3.3 Analyze and report	31
3.3.1 Analyze	31
3.3.2 Report	32
3.4 Detect and decide	32
3.4.1 Detect	32

3.4.2 Decide	33
3.5 Integration	33
3.5.1 Data and transactions	33
3.5.2 Integration within a system of insight	34
3.5.3 Action execution	34
Chapter 4. Consume and collect	37
4.1 Consuming data in motion	38
4.1.1 Event sources	39
4.1.2 Event distribution and filtering	44
4.2 Collecting data at rest	47
4.2.1 Batch processing	47
4.2.2 Database triggers	48
4.2.3 Big data and business rules	49
Chapter 5. Analyze and report	51
5.1 Descriptive analytics	53
5.2 Predictive analytics	53
5.2.1 The predictive analytics process: The CRISP-DM data mining process.	54
5.2.2 Predictive analytics with SPSS	55
5.2.3 SPSS Modeler capabilities	56
5.2.4 The predictive modeling process with SPSS	57
5.3 Real-time analytics	62
5.4 Prescriptive analytics	63
5.5 Cognitive analytics	63
5.6 Reporting and monitoring	63
5.6.1 Reports	64
5.6.2 Dashboards.	64
Chapter 6. Detect and decide	67
6.1 Introduction	68
6.1.1 Detect	68
6.1.2 Decide.	68
6.1.3 Interaction between detect and decide	69
6.1.4 Decision management	69
6.1.5 ODM in the context of systems of insight	71
6.2 Request-driven decisions in ODM Decision Server Rules	73
6.2.1 Solution components	74
6.2.2 Decision Service projects	78
6.2.3 Modeling and authoring artifacts.	79
6.2.4 Rule project artifacts	86
6.3 Situation-driven decisions in ODM Decision Server Insights	88
6.3.1 Decision Server Insights components.	88
6.3.2 Business model definition	92
6.3.3 Agents.	94
6.3.4 Lifecycle of an event	97
6.3.5 Building context.	98
6.3.6 Using context	100
6.3.7 Scheduling rule execution.	103
6.3.8 Integration and connectivity	103
Chapter 7. Taking action.	107
7.1 Actionable insight	108
7.2 Common actions	109

7.2.1 Asynchronous Interactions	109
7.2.2 Synchronous interactions	113
7.2.3 Feedback	113
7.3 Continuous improvement	115
Chapter 8. Integration examples with ODM Advanced	117
8.1 Scenario overview	118
8.2 Architecture	119
8.3 Solution setup	121
8.3.1 Solution definition	122
8.3.2 Model definition	124
8.3.3 Agents definition	128
8.4 Consume and collect example	129
8.4.1 Message queue configuration	130
8.4.2 InfoSphere Streams configuration	133
8.4.3 DSI configuration (connectivity and server configuration)	137
8.5 Analyze and report example	143
8.5.1 Create scenario model	144
8.5.2 Store model as a stream on the server	148
8.5.3 Scoring model configuration	152
8.5.4 Predictive analytics agent implementation	159
8.6 Detect and decide example	165
8.6.1 Creating the decision service	165
8.6.2 Creating the OSGi service	171
8.6.3 Mapping the OSGi service to a BOM entry	179
8.6.4 Starting the service in a rule agent	179
8.7 Taking action example	180
8.7.1 Decision Server Insights outbound integration with IBM Integration Bus	181
8.7.2 Decision Server Insights outbound integration with IBM Business Process Manager (IBM BPM)	186
Chapter 9. Building systems of insight with ODM Advanced	197
9.1 Adoption	198
9.1.1 Identify use case	198
9.1.2 Maturity model	199
9.2 Request-driven decisions development cycle	201
9.2.1 Rule analysis	201
9.2.2 Modeling	202
9.2.3 Authoring	202
9.2.4 Validation	203
9.2.5 Deployment	207
9.2.6 Monitoring	207
9.2.7 Execution patterns	208
9.2.8 Concept of operations	210
9.2.9 Applicability in a service-oriented architecture (SOA)	211
9.3 Situation-driven decisions development cycle	212
9.3.1 Analysis	213
9.3.2 Model	213
9.3.3 Solution development	214
9.3.4 Connect	214
9.3.5 Test	214
9.3.6 Deploy	216
9.3.7 Execute	217

9.3.8 Monitor	217
9.3.9 Concept of operations	218
9.3.10 Applicability in an event-driven architecture	218
9.4 Considerations for planning and designing DSI deployments	220
9.4.1 Designing for performance	220
9.4.2 Configuring for performance	222
9.5 ODM Decision Server Insights topologies	222
9.5.1 Component definitions	222
9.5.2 Minimum configuration	224
9.5.3 HA and CA configuration	225
9.5.4 Selecting your topology	226
9.6 Transactionality and rollback	228
Appendix A. Integration of ODM Decision Server Insights with custom services . .	231
A.1 OSGi service implementation	232
A.1.1 Setting up	232
A.1.2 Creating the OSGi bundle project	232
A.1.3 Coding the service	234
A.1.4 Configuring the manifest	235
A.1.5 Configuring the blueprint	238
A.2 XOM to BOM mapping	239
A.2.1 Creating a rule project	239
A.2.2 Creating a BOM entry	241
A.2.3 Configuring the BOM entry	242
A.2.4 Verbalizing the service call	243
A.2.5 Final setup	244
Related publications	245
IBM Redbooks	245
Online resources	245
Help from IBM	246

Notices

This information was developed for products and services offered in the U.S.A.

IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not grant you any license to these patents. You can send license inquiries, in writing, to:

IBM Director of Licensing, IBM Corporation, North Castle Drive, Armonk, NY 10504-1785 U.S.A.

The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law: INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to non-IBM websites are provided for convenience only and do not in any manner serve as an endorsement of those websites. The materials at those websites are not part of the materials for this IBM product and use of those websites is at your own risk.

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

Any performance data contained herein was determined in a controlled environment. Therefore, the results obtained in other operating environments may vary significantly. Some measurements may have been made on development-level systems and there is no guarantee that these measurements will be the same on generally available systems. Furthermore, some measurements may have been estimated through extrapolation. Actual results may vary. Users of this document should verify the applicable data for their specific environment.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental.

COPYRIGHT LICENSE:

This information contains sample application programs in source language, which illustrate programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to IBM, for the purposes of developing, using, marketing or distributing application programs conforming to the application programming interface for the operating platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs.


Trademarks

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

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

BigInsights™
Bluemix™
Cognos®
CPLEX®
DataPower®
DB2®
developerWorks®
i2®
IBM MobileFirst™

IBM PureData™
IBM Watson™
IBM z™
IBM®
ILOG®
InfoSphere®
Insight™
Insights™
Maximo®

PureData®
Redbooks®
Redbooks (logo) ®
SPSS®
Symphony®
WebSphere®
z Systems™
z/OS®

The following terms are trademarks of other companies:

Evolution, Kenexa, and Inc. device are trademarks or registered trademarks of Kenexa, an IBM Company.

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

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

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

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

Find and read thousands of IBM Redbooks publications

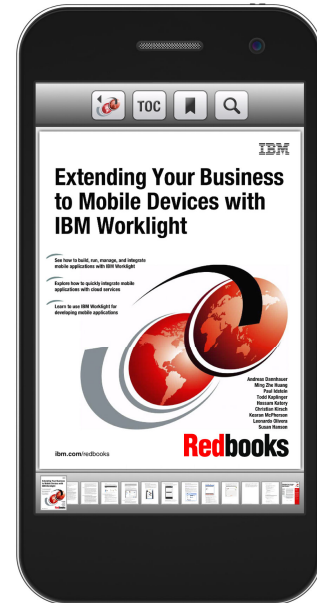
- ▶ Search, bookmark, save and organize favorites
- ▶ Get up-to-the-minute Redbooks news and announcements
- ▶ Link to the latest Redbooks blogs and videos

Get the latest version of the Redbooks Mobile App



Download
Now

iOS



Promote your business in an IBM Redbooks publication

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

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



ibm.com/Redbooks

About Redbooks → Business Partner Programs

THIS PAGE INTENTIONALLY LEFT BLANK

Preface

Systems of record (SORs) are engines that generate value for your business. Systems of engagement (SOE) are always evolving and generating new customer-centric experiences and new opportunities to capitalize on the value in the systems of record. The highest value is gained when systems of record and systems of engagement are brought together to deliver insight.

Systems of insight (SOI) monitor and analyze what is going on with various behaviors in the systems of engagement and information being stored or transacted in the systems of record. SOIs seek new opportunities, risks, and operational behavior that needs to be reported or have action taken to optimize business outcomes. Systems of insight are at the core of the Digital Experience, which tries to derive insights from the enormous amount of data generated by automated processes and customer interactions. Systems of Insight™ can also provide the ability to apply analytics and rules to real-time data as it flows within, throughout, and beyond the enterprise (applications, databases, mobile, social, Internet of Things) to gain the wanted insight. Deriving this insight is a key step toward being able to make the best decisions and take the most appropriate actions. Examples of such actions are to improve the number of satisfied clients, identify clients at risk of leaving and incentivize them to stay loyal, identify patterns of risk or fraudulent behavior and take action to minimize it as early as possible, and detect patterns of behavior in operational systems and transportation that lead to failures, delays, and maintenance and take early action to minimize risks and costs.

IBM® Operational Decision Manager is a decision management platform that provides capabilities that support both event-driven insight patterns, and business-rule-driven scenarios. It also can easily be used in combination with other IBM Analytics solutions, as the detailed examples will show. IBM Operational Decision Manager Advanced, along with complementary IBM software offerings that also provide capability for systems of insight, provides a way to deliver the greatest value to your customers and your business. IBM Operational Decision Manager Advanced brings together data from different sources to recognize meaningful trends and patterns. It empowers business users to define, manage, and automate repeatable operational decisions. As a result, organizations can create and shape customer-centric business moments.

This IBM Redbooks® publication explains the key concepts of systems of insight and how to implement a system of insight solution with examples. It is intended for IT architects and professionals who are responsible for implementing a systems of insights solution requiring event-based context pattern detection and deterministic decision services to enhance other analytics solution components with IBM Operational Decision Manager Advanced.

Authors

This book was produced by a team of specialists from around the world working at the International Technical Support Organization, San Jose Center.

Whei-Jen Chen is a Project Leader at the International Technical Support Organization, San Jose Center. She has extensive experience in application development, database design and modeling, and IBM DB2® system administration. Whei-Jen is an IBM Certified Solutions Expert in database administration and application development, and an IBM Certified IT Specialist.



Rajeev Kamath is a CTP and Executive Program Manager in the area of Business Analytics. He has been with IBM for over 25 years and has held many leadership positions that have included worldwide responsibilities. He has been a Project Executive and a Program Manager, and has managed several large, complex customer projects including deployments of SAP, eBusiness, eCommerce, and integrated manufacturing solutions. He has also played a leadership role in worldwide sales and setting up of ecosystems for Linux and Linux on IBM z™ Systems, SAP on z Systems™, and the SOA Infrastructure solution. He has closely worked with IBM customers in the Financial, Public, Communication and Industrial sectors.

He is an IBM certified IT Specialist and Executive Project Manager. He holds a BS (Mechanical Engineering), MS (Manufacturing Engineering) and Master's degree in Industrial Engineering with a minor in Operations Research from NC State University in Raleigh, USA.



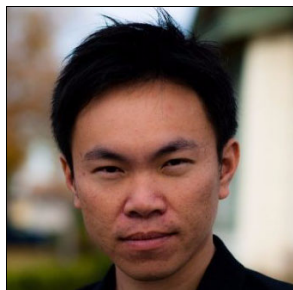
Alexander Kelly is a Performance Analyst at the IBM Lab in Hursley, UK. He has worked on IBM Operational Decision Manager Decision Server Insights™ since the product was first developed. He specializes in the clustered configuration of the product, and the database persistence of contextual data.



Hector H. Diaz Lopez is a Client Technical Professional for Systems Middleware. He has 15 years of cross industry experience in software development, analysis, design, and implementation of multiple IT projects. He has spent the last 9 years working with Decision Management technologies as a Business Rules Management System consultant and technical seller assisting customers with their ODM applications. He holds an E-Commerce Masters degree and a BS degree in Computer Systems Engineering from ITESM in Mexico.



Matthew Roberts designs process improvement solutions for IBM clients across the UK and Ireland. As a member of the UKI technical sales team, Matt leads the adoption of IBM Operational Decision Manager Advanced within systems of insight. In this role, Matt builds proof of concept prototypes and leads the technical specification of new Smarter Process solutions. Matt is a certified developer for IBM Operational Decision Manager and IBM Business Process Manager. He holds a Master of Computer Science degree from the University of Oxford.



Yee Pin Yheng is the Product Design Lead for IBM Operational Decision Manager Decision Server Insights. Coming to IBM through the ILOG® acquisition, Yee Pin brings 13 years of experience in Business Rules Management System. He has been the ODM product manager for 6 years, and manages the product direction for business user tooling and enterprise integration solutions.

The authors would like to thank the following people for their extensive support and consultation of this book:

Amy Dickson, Senior IBM Operational Decision Management Product Manager

David Martin, Core Architect, Operational Decision Management

Andy Ritchie, Senior Smarter Process Product Manager

Daniel Selman, Real Time Actionable Insights Architect, STSM

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

Tony Huo

IBM USA

Kim Clarks

Nigel Crowther

Lewis Evans

Francis Friedlander

Jose De Freitas

Peter Johnson

Tim Poultney

David Rushall

IBM United Kingdom

Now you can become a published author, too!

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

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

ibm.com/redbooks/residencies.html

Comments welcome

Your comments are important to us!

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

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

ibm.com/redbooks

- Send your comments in an email to:

redbooks@us.ibm.com

- Mail your comments to:

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

Stay connected to IBM Redbooks

- Find us on Facebook:

<http://www.facebook.com/IBMRedbooks>

- Follow us on Twitter:

<http://twitter.com/ibmredbooks>

- Look for us on LinkedIn:

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

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

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

- Stay current on recent Redbooks publications with RSS Feeds:

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



Introduction

This chapter describes the concept of a system of insight and its close relationship with business decision making. It also introduces the concepts, solution, and examples that are described in the rest of this book.

The goal is to provide a complete set of sophisticated and advanced decision-making solutions for enterprises in multiple industries. IBM offers a comprehensive and broad portfolio of business intelligence, predictive analytics, big data analytics, cloud based analytics, real-time analytics, optimization, and other solutions. IBM also provides thought leadership in the analytics space by developing business value publications for various industries.

This chapter covers the following topics:

- ▶ Decision making
- ▶ Motivation and business challenges
- ▶ Decision automation
- ▶ Business use cases
- ▶ Example of system of insight provided in this book

1.1 Decision making

There is no denying that we are in an era of rapid digital transformation. Digital technology is touching every aspect of our lives. We are rapidly transforming with the expansion of mobile technology, social media, cloud computing, advances in analytics, and so on. The rate of innovation and adoption has accelerated over the past few decades and is only expected to change at an even faster pace. This transformation is happening in virtually all industries and in all parts of the globe. In this digital transformation era, you must have technologies and solutions to support this rapid and accelerating change. This book introduces “systems of insight” as a solution that uses the recent advances in analytics and shows how enterprise decision making plays a critical role in these systems.

Decision making is a critical function in any enterprise. The quality of decisions has a significant impact on the success or failure of the business. Decisions based on knowledge, experience, and sound logic have a much higher probability of delivering success. Businesses are automating many of their decision-making processes to improve speed and increase consistency. Businesses require even better and more productive tools for people to perform business tasks by analyzing data and making the best decisions at the correct time, based on the most current information available. Analytics is all about improving the quality of decisions by gathering, filtering, transforming, analyzing, and mining data to gain sound knowledge and insight into trends, patterns, and anomalies based on experience. You then apply this insight and logic to the decision-making process. The decision-making process that is enhanced by analytics can be described as consuming and collecting data, detecting relationships and patterns, and applying sophisticated analysis techniques, reporting, and automation of the follow on action. The IT systems that support decision making are composed of the traditional “systems of record”, “systems of engagement”, and the “systems of insight”. See Figure 1-1.

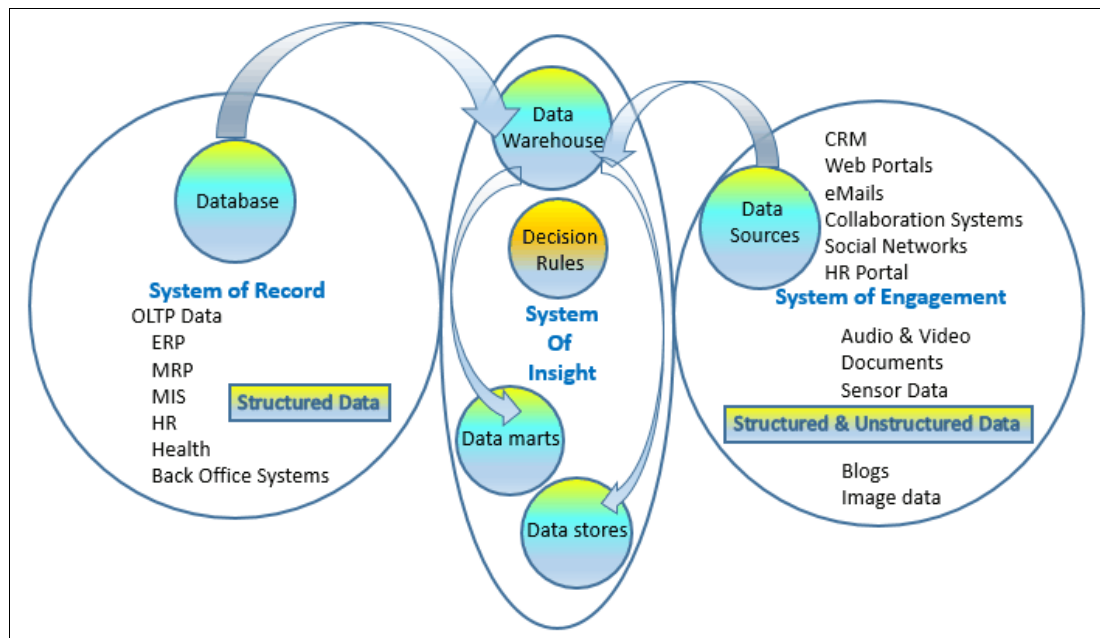


Figure 1-1 Decision making with systems of record, systems of engagement, and systems of insight

At present, most enterprises decisions are based on the information provided by the systems of record and systems of engagement. Systems of insight bring the data in systems of record and systems of engagement together, analyze it, apply business policies and rules to the

combined data, derive insight, and make recommendations to improve the quality of decisions.

In many cases such as in a call center, customer interaction, or a web-based application, a system of insight can be engaged to improve the quality of decisions by accessing new insight not previously available. The net effect can be a more satisfied client, additional sales, or early detection of fraudulent transactions.

Systems of record are similar in how they can be enhanced by systems of insight. Both back-end processes and applications that support decision making can benefit from improved insight. In claims processing, which has eligibility, fraud detection, and payment decisions, the fraud detection activity might benefit from additional and broader insight and make proactive decisions possible.

A phased implementation of systems of insight allows existing decision processes based on systems of engagement and systems of record to be incrementally improved and extended without requiring a major transformation.

An example of where systems of insight enhance systems of engagement is in cross-sell - upsell decisions. A broader insight into the clients' background and behavior can be achieved to determine what the client might find highly valuable or to add a good upsell to what they already have.

An example of where systems of insight enhance a systems of record based decision-making process is in a counter-fraud solution where a broader data set is analyzed to find anomalies and specific patterns of behavior to proactively identify fraudulent activities.

1.1.1 Systems of record

Originally, when IT became widespread in support of business, it consisted mainly of *systems of record*¹. These were the electronic data processing systems that managed census data, the older enterprise resource planning (ERP) systems such as financials, payroll, CRM, HR, order entry, inventory management, and product management, and so on. The systems of record include a data storage and access system and are the authoritative source of enterprise data. Ideally, for data integrity, governance, and security, have a single source of the data in a system of record for any specific data. This configuration offers what is known as a “single version of the truth”, a highly sought after capability that ensures that the input to the decision making is as accurate as possible. For historical, static data, systems of record offer a traceable path to original data. For data that continues to change such as inventory levels and bank account balances, systems of record provide the latest information.

The advent of systems of record accelerated the initial expansion of business worldwide and made today's globalization possible. Systems of record are a basic necessity for running any business.

1.1.2 Systems of engagement

Globalization has put new demands on businesses. Consumers are now geographically dispersed worldwide. Thanks mainly to the internet and social media, these consumers are equipped with current knowledge, are a lot more discerning, and have higher expectations of product and service quality and value. Also, competition is coming from worldwide businesses with unique local advantages such as lower labor costs. Additionally, the center of business growth has shifted to the developing world where most of the world's population resides. Businesses are adjusting to these changes and analytics is playing a huge role in this transformation. One of the major areas of focus for businesses now is “customer centricity” or

catering to every customer as an individual as opposed to dealing uniformly with an entire “segment” of customers. A true example of lack of customer centricity is when a major global airline sends promotions and discounted fares through email where none of the flights offered originate from the recipients’ home airport.

Enterprise IT systems must adapt to this transformation for the businesses to survive and thrive. There is also tremendous pressure to do more with less. IT is also constantly under pressure to cut costs. That means all resource utilization including IT must be constantly improved and optimized. IT must also evolve, advance, and provide more functionality.

IT is now being used to support the superior consumer experiences that the discerning consumer expects. To make that happen, enterprise IT must efficiently support collaboration between various components and participants of businesses that includes partners, suppliers, and consumers. This is being accomplished by “systems of engagement.” These systems use modern information technologies that efficiently communicate across various organizations and networks within the enterprise to deliver the collaborative experience necessary for the business to succeed in the globally competitive environment.

Systems of engagement refer to customer-centric systems that actively encourage peer interactions and collaboration. A few examples of such systems are the social applications that facilitate collaboration in research and development, project management, and other business processes; web portals for customer interactions; and mobile device access for customers and employees. The customer access might also be through multiple channels such as a mobile device, email, web chat, IM, SMS, social channels through call centers, telemarketers, and help desks. One key requirement of system of engagement is integration between applications such as supply chain management, ERP, HR systems, and product management.

An example of a system of engagement is a hotel where the customers interact with the hotel through their web portal. Customers use the hotel website to make reservations, select room location, specify room size, select wanted amenities, manage their loyalty program, and set personal preferences. Data that is collected by systems of engagement includes data gathered about the individual users and their interaction data from social networks.

Generally, depending on their functions, systems of engagement are classified in two categories: Customer-centric and device-centric. Device-centric systems include systems that support devices, Internet of Things (IoT), and vehicles.

Good and effective decision making in such a complex environment can be a major challenge. The decision maker must take into account the impact of several variables on the outcome, which is beyond the capabilities of human intellect. The key to assessing the outcomes lies in the systems of record and systems of engagement. Most of the information required for effective decision making is available in these systems. This information must be selectively cleansed, filtered, transformed, mined, and analyzed to be useful for successful decision making. This task of bringing systems of record and engagement together to produce actionable insight and facilitate decision making is accomplished by “systems of insight.”

1.1.3 Systems of insight

Systems of insight integrate data in the systems of engagement and systems of records to find new relationships and patterns by analyzing historical data, assessing the current situation, applying business rules, predicting outcomes, and proposing the next best action. Systems of insight are analysis systems that facilitate gathering, mining, organizing, transforming, consuming, and analyzing diverse sets of data with statistical modeling tools to

detect patterns, report on what has happened, predict outcomes with a high degree of confidence, apply business rules and policies, and provide actionable insight.

Systems of insight require a cross-functional model for converging and analyzing the data from systems of record and systems of engagement. Systems of insight allow businesses to employ new and sophisticated analytic technologies to the data far beyond the old methods where data from systems of record and systems of engagement were examined separately for retroactive decision making.

Systems of insight begin where data is collected and stored or where a transaction creates a record or a stream of records. These data and records are usually retained and provided by the traditional systems of records and systems of engagement. It supports the entire decision-making process and ends with input to the Action phase. These inputs range from simple alerts to execution commands automatically sent to a system that can run an action. Figure 1-2 shows the building blocks of a system of insight. These are described in detail in Chapter 3.

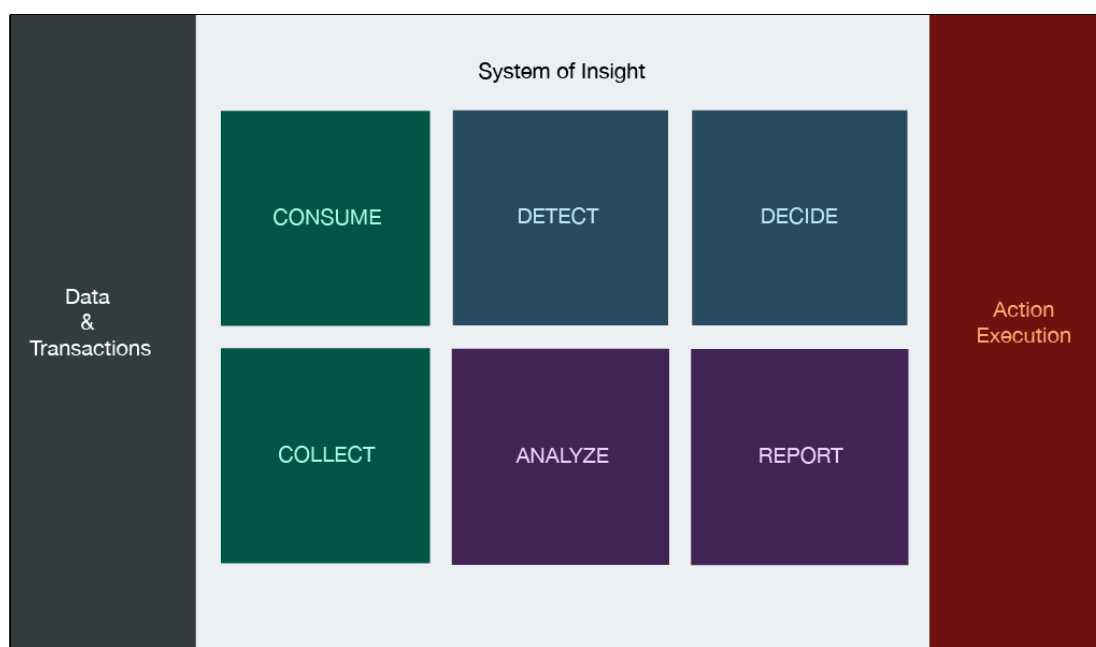


Figure 1-2 System of insight building blocks

This building block model represents many of the analytics situations supported by the systems of insight. The best way to illustrate the applicability of this model is through various examples. Here are three different examples of the model's applicability:

► **Example 1: Real-time transaction analytics**

In real-time analytics, the transaction is analyzed when it arrives in the system of engagement while it is still being processed. The process of analysis, usually known as “scoring,” is performed as part of the transaction processing workflow. This is necessary because of the low latency required for a real-time response. The score is calculated by making a call to the system of insight to run a pre-built statistical model. The “score” is a mathematical representation and an indicator of the nature of the specific transaction. The score indicates the probability of the transaction being a legitimate one or a fraudulent one. The processing or follow-on treatment of the transaction depends on its score.

An example of this is a credit card or ATM transaction analyzed during processing while the customer is still waiting or is online. The score in this case would indicate the propensity or the probability of the transaction being fraudulent. A transaction flagged as

fraudulent would then be suspended and sent for further investigation. The processing or treatment of the transaction depends on its score.

In this case, the following sequence of building blocks from the model are used:

- a. Consume: the contextual information about the transaction or stream
- b. Detect: whether the transactions are legitimate or not
- c. Decide: what action to take

► Example 2: Batch analytics

In this case, the data is collected, transformed, mined, and insight is gained and applied to the decision. One example of the use of batch analytics is the creation of a marketing campaign or promotion based on customer demographics and behavioral analytics. Another example is analytics applied to population and demographic health data and health insurance claims data for disease control and health management programs. Here, historical data is gathered, filtered, mined, and analyzed to make decisions and generate reports.

In this case the following sequence of building blocks from the model are used:

- a. Collect: the relevant data required for analysis
- b. Analyze: that data to derive insight and decision recommendations
- c. Decide: the best action to take
- d. Report: to review the impact of the decisions, in-depth analysis, and perform root cause analysis

► Example 3: Broad based analytics

Analysis that combines elements of batch and real time to deliver comprehensive enterprise solution. One example is operational analytics that provide real-time visibility to enterprise operations or a deeper 360-degree view of customers. Operational analytics might involve analyzing vast amounts of data from multiple sources and in multiple formats, and data at rest and in motion including streams to provide the necessary insight rapidly. This type of analytics requires complex analysis, correlations across different data sets, and might involve detection in real time, reporting, and visualization.

In this situation the following sequence of building blocks from the model are used:

- a. Consume: the contextual information about the transaction or streams
- b. Collect: the relevant data from multiple sources and formats for analysis
- c. Analyze: that data to derive insight and decision recommendations
- d. Detect: the specific nature of the transaction to determine the next best action
- e. Decide: the best action to take
- f. Report: to review the impact of the decisions, in-depth analysis, and perform root cause analysis

Recent trends in systems of insight include big data analysis systems that seeks to gain actionable insights from structured and the vast amount of data available in unstructured data from various sources such as social media and the web. Insight gained from such systems can help enterprises improve in many areas such as discovery of new markets, customer retention, improved customer service, employee retention and enhanced customer loyalty.

When properly applied, systems of insight can help corporations in various ways such as opening new markets and finding new ways to improve customer service and enhance loyalty.

Depending on the requirements, several of the analytics solutions from the IBM portfolio such as SPSS®, SPSS Modeler with Real Time Scoring capability, Cognos®, InfoSphere® BigInsights™, or Operational Decision Manager Advanced with Decision Server Insights can be components of systems of insight. Examples of these solutions are described in Chapter 2.

Integration of systems of insight into business processes can be achieved in various ways depending on the state of the current decision-making process within the enterprise. At a high level, there are three major ways of integrating system of insight into the existing systems:

- ▶ Extend the current decision-making process that might include systems of record and system of engagement by incorporating systems of insight to enhance operational decisions
- ▶ Use the insight provided by systems of insight to adjust business logic to enhance current decision-making processes and improve performance.
- ▶ Extend system of insight and system of engagement application processes to more decision areas and use operational decisions supported by scores that are provided by systems of insights.

1.1.4 Types of decisions

In an enterprise, decisions are made at different levels. Different decisions require different inputs, have different time frames or frequencies, are made in different parts of the organization, and have different intensity of effects. Depending on these characteristics, decisions are classified as strategic, tactical, or operational (at the transaction level).

Strategic decisions

As the name implies, strategic decisions have significant impact on the enterprise, are made at the highest level of the organization, and require input from multiple internal and external sources. Frequency of strategic decisions usually is low.

Strategic decision makers usually use systems of insight for reporting (for example, quarterly or monthly) to make decisions such as major investments, divestments, mergers, acquisitions, and so on. One of the powerful uses of systems of insight in strategic decision making is the “What if” capability, particularly when multiple alternatives must be evaluated using inputs such as historical data.

Tactical decisions

Tactical decisions are made more frequently than strategic ones, possibly weekly or even more frequently. Tactical decisions are those that have medium grade impact and might be made at the branch or regional levels. These are usually made at mid-management level and might also require inputs from multiple people.

For tactical decisions, the systems of insight can provide the key performance indicators (KPIs) for monitoring performance on a weekly or daily basis. Based on these KPIs, the business process parameters can be adjusted for improvements as needed. These adjustments can involve changing the decision thresholds or decision logic. Systems of engagement can also perform data mining to discover patterns or trends that might not be otherwise apparent. The data mining exercise can help detect opportunities, highlight risks, and identify any operational trends.

Operational decisions

Operational decisions are granular in that they are related to individual transactions. These decisions relate to how an individual transaction is treated. Individually, these decisions have

a much smaller impact on the enterprise. In the aggregate though, these decisions can have substantial impact on the success of the enterprise. Obviously, the frequency of the operational decisions must match that of the incoming transactions. In many industries such as in retail, large amounts of transactions and consequently decisions must be made. With recent business trends such as mass customization (treating every transaction uniquely), the importance of sound decision making at the operational level has increased substantially.

Depending on the types of decision, the frequency and amount of time to process vary. Ideally, the system of insight must be capable of supporting all types of decisions. Getting the correct decisions is critical. All available options must be evaluated and the one most likely to succeed must be chosen. To evaluate the options, the decision makers must have solutions that estimate the effects with a high degree of confidence. Systems of insight provide such a solution.

For operational decisions, systems of insight can provide a “score” for individual transactions that can be used for automated operational decision making. Systems of insight event-based solutions can detect real-time patterns and take action in near real time.

1.2 Motivation and business challenges

Getting correct decisions or as close to correct as possible is critical for business success. To improve the quality of decisions it is essential to learn from past experiences, analyze the current situation in that context, apply company policies and business rules, and take in to account any relevant regulations. When the actions are based on such a logical foundation, the results are substantially better than decisions based on human intuition and human recollection of past experiences.

In today's global, highly competitive world where consumers are expecting more, businesses must evolve to meet these challenges. Businesses must produce high-quality products and services at a competitive price and deliver them to the consumers, while keeping competitors at bay. At the same time, the business must reduce losses due to errors, fraud, and so on to grow the business and increase profits.

To make all of these things possible, the use of analytics is becoming essential. The recent trend is to use predictive analytics to arrive at decisions based on historical performance, taking into account current marketplace changes, applying company policies, best practices, and accommodating relevant regulations. Availability of relevant data, reporting applications, business rules engines, and sophisticated statistical and modeling tools make it possible to predict outcomes with a high degree of confidence.

Research and surveys of current businesses have shown that companies that are using analytics are increasing revenue, reducing losses, increasing customer satisfaction, retaining employees and in general are a lot more successful in the global marketplace.

1.2.1 Decision-making process in general

As explained earlier, in general, decisions can be classified in three categories: Strategic, tactical, and operational. Although the strategic and tactical decisions have bigger impacts per decision, the operational ones have a substantial impact in aggregate. The operational decisions occur at a high frequency and require a quick response. Therefore, the operational decisions are well suited for automation with a fast response time.

Decision making has evolved with the changes in the business environment. First, from the days of human intuition and experience, the basis for decision making moved to reports or

descriptive analytics. With the availability of sophisticated modeling and statistical tools, and access to vast amounts of historical data, decision making is vastly enhanced with *predictive analytics*. Predictive analytics can facilitate preemptive and proactive decision making.

Reporting

Descriptive analytics or *reports* are the basic inputs for informed decision making. Reports answer the question “What Happened?” This is usually the basis for the next decision. Reports are usually enhanced with graphics for easier understanding of trends and patterns. In many instances, multiple graphical reports are displayed on a single window, making it a *dashboard*. In this case, by updating each report periodically, the most current information is communicated to the decision makers. In many instances, the individual reports provide the ability to deep dive into specific parameters.

Batch analytics

As the name implies, batch analytics pertains to analytics performed after the fact on accumulated historical data. Traditionally, due to the limitations of available technology and solutions, analytics were usually performed on batch data. Now there is a real effort to deliver analytics when and where needed. In some instances, that would mean analytics in real time and in others the traditional batch analytics might be adequate.

Real-time analytics

In many operational situations, making decisions in real time can improve the results substantially. For example, in a situation where a payment has to be made against a claim and the claim turns out to be fraudulent, recovering that money paid after the fact is almost impossible. This *pay and chase* model just does not work. Instead, if the fraudulent transactions are flagged before the payment is made, the loss can be prevented. This would be possible only with real-time analytics.

In real-time analytics, every transaction must be analyzed and evaluated against historical patterns. Possible bad transaction must be flagged, business policy-based rules applied, and the transaction dealt with appropriately. All of this must happen in a short time while the requester is waiting for the response. The analytics system must also minimize or eliminate “false positives” in identifying possible fraud.

Although each one of these cases are examples of systems of insight and useful in their own right, real-time analytics can prevent the adverse impact of processing a transaction before it occurs. Chapter 9 presents a maturity model that depicts the progression of an enterprise through the analytics maturity journey.

1.3 Decision automation

Automation makes the process fast, consistent, and free from potential human errors. Automation of decision making is easily justified when the decisions are repeatable and their frequency is high. Most real-time operational decisions certainly qualify as good candidates for automation. Identifying and sending a potentially fraudulent transaction to a case management solution for disposition while completing all legitimate transactions rapidly is possible only with automation.

The decision-making process involves analysis of historical data, data mining, data transformation, reporting, applying statistical models, predicting outcomes with high confidence levels, applying business rules, and enabling or taking actions. Further enhancement to the decision-making process is through situation detection and event processing. The issue that needs to be resolved is identified and defined by a situation, and

solutions are sought through analytics to resolve the situation. An event is a change of state that is used to determine whether an action is needed to be taken and carried out. For more information, see Chapter 3.

With the availability of highly reliable and flexible computing systems, sophisticated statistical modeling and analysis tools, and rules engines, it is now possible to automate the entire decision-making process.

1.3.1 Decision automation versus decision optimization

An important distinction must be made between decision automation, based on business rules, optionally combined with prescriptive models, and decision optimization. Decision optimization models a business problem as a set of mathematical equations and constraints to be solved using a specialized solver, such as IBM CPLEX®. Decision optimization is used in many industries to build optimal, or high quality, plans and schedules. For example, in transportation, logistics, and energy distribution, financial services decision optimization can lead to extreme returns on investment. In fact, in some industries, the ability to build high-quality plans and schedules is a prerequisite for offering a competitive offering.

For example, an airport can use decision optimization to compute optimal gate assignments for aircraft. The optimization engine balances competing constraints to optimize gate turnaround time, maximize utilization of the gates, minimize passenger congestion within the terminal, and many other factors related to operational efficiency.

This book does not attempt to cover decision optimization in detail. For more examples of applying decision optimization, see the “IT Best Kept Secret is Optimization” blog and information about IBM CPLEX. For the “IT Best Kept Secret is Optimization” blog, see the following website:

<https://www.ibm.com/developerworks/community/blogs/jfp?lang=en>

1.4 Business use cases

The following examples are business use cases:

- ▶ Banking
 - Fraud detection in real time: Systems of insight can calculate a score and apply business rules to the transaction to determine the probability of the transaction being fraudulent. Further investigation or action can then be taken based on business requirements. Where necessary the identification can be done in real time to prevent the fraud.
 - Customer behavioral intelligence: Systems of insight can analyze business data and social feeds such as social networks, customer service interactions, and web click stream data to predict potential customer behavior and preferences, and determine the next best action.
 - Anti-money laundering analytics: Systems of insight can be designed to help reduce exposure to money laundering and terrorism activities. The systems of insight solution can effectively monitor customer transactions daily and use the customers’ historical information and account profile to provide an enterprise-wide picture of money laundering and other adverse activities.

► Insurance

- Insurance claims fraud detection: During claims intake, systems of insight solutions designed to collect and analyze streaming data, such as social media posts or geospatial data, can help insurers discover fraud. They can, for example, determine whether policyholders are being honest about accident details or if the services rendered are legitimate. Also, the predictive analytics component of systems of insight can help categorize risk and deliver fraud propensity scores to claim intake specialists in real time so they can adjust their queries and route suspicious claims to investigators. Systems of insight can also perform analysis of fraudulent claims and their impact on the business through reports and visualizations of data patterns.
- Customer behavioral intelligence: Systems of insight can analyze business data and social feeds such as social networks, customer service interactions, and web click stream data to predict potential customer behavior and preferences, and determine the next best action that can include upselling, cross selling, or highly targeted promotions.

► Energy and Utilities

- Asset management and preventive maintenance for transmission and distribution equipment: Systems of insight, through analytics and visualization can improve situational awareness through which utilities can better understand factors affecting asset performance and more accurately project the remaining useful life of the asset or what other points in the grid are susceptible to outages. When integrated with asset management solutions such as IBM Maximo®, systems of insight can use historical equipment performance, repair, and lifespan data to predict potential component failures to initiate preventive maintenance actions.

► Retail

- Customer behavioral intelligence: Systems of insight can analyze business data and social feeds such as social networks, customer service interactions, and web click stream data. This data can be used to predict potential customer behavior and preferences, and determine the next best action that can include upselling, cross selling, or highly targeted promotions.
- Social media and sentiment analytics: Systems of insight can analyze public sentiment the results of which retail business can use to customize incentives, promotions, and offerings. IBM has advanced analytics solutions that can track the spread of trends geographically, chronologically, and culturally. On such example is the IBM *Birth of a Trend* project that is a unique effort dedicated to understanding the science behind predicting online trends that might revolutionize an industry. By studying how online trends spread globally, IBM provides deep insights into whether what the trending on social networks is, or is likely to become, commercially viable.

► Manufacturing

- Demand forecasting: Systems of insight can help forecast by using data from a multitude of internal and external sources. Systems of insight can use algorithmic models and predictive analytics to create a *demand forecast* that attempts to predict the number of goods or services that customers and businesses will demand in the future. Demand forecasting is also used by other industries such as retailers, energy and utilities, and financial.
- Supply chain management and optimization: Today's supply chains are global, interconnected, fast, and lean. That also means that the supply chains are vulnerable to shocks and disruptions from various sources around the world. Systems of insight with advanced analytics and modeling capabilities can help decision makers evaluate alternatives against the incredibly complex and dynamic set of risks and constraints and make the best decisions possible.

- Asset management and preventive maintenance for manufacturing equipment: Systems of insight can compliment asset management solutions such as IBM Maximo to reduce unplanned downtime, and improve asset performance and efficiency. Using advanced analytics, systems of insight can analyze structured and unstructured data including equipment history, sensor data, maintenance records, and external data such as weather and geographical data to predict potential failure of components or systems. They can then feed the prediction into the asset management solution to schedule preventive maintenance. The needed spare parts and skilled labor can be planned and managed based on the predictions.
- Inventory management: Systems of insight can use advanced analytics to optimize inventory levels at each stage in the entire supply chain from raw material to work in process to finished goods. The inventory is optimized based on demand forecasts and product mix predictions that are also developed by systems of insight.
- Mass customization of products and services: Systems of insight can provide customer behavior intelligence and buying patterns that can be used to customize products and services by treating every customer as an individual.
- Post sales product maintenance and warranty management: Systems of insight can enhance maintenance and warranty services with predictive maintenance and by providing insights that can be used to improve warranty management.
- Government
 - Crime prediction and prevention: Systems of insight can integrate numerous data sources with structured and unstructured data to provide alerts in real time based on newly discovered patterns, variables, and event correlations for improved situational awareness. The discovered patterns are used to anticipate risks and identify potential repeat offenders. Based on the identified risks, the government department can proactively deploy resources for crime prevention.
 - Fraud detection and prevention: Many federal agencies are faced with fraud in different forms. Systems of insight can calculate a “score” and apply business rules to the transaction and determine the probability of the transaction being fraudulent. Further investigation or action can then be taken based on business requirements.

1.5 Example of system of insight provided in this book

This book illustrates one implementation of a system of insight with an example. This example is from the airline industry and highlights how an airline can use a system of insight to maintain customer satisfaction and retention, and reduce the costs associated with delays.

In this example, a passenger has arrived at the airport to board a flight to his destination that includes a stop and a connecting flight. Because of the specific circumstances at the airport and the passenger’s arrival time, there are potential issues that might disrupt the passenger’s journey, leaving him or her dissatisfied. This is an unusually busy time at the airport and the customer has arrived relatively late and is likely to miss the flight due to the additional time required to reach the gate. By predicting the arrival time at the gate, circumstances of the flight, and so on, the airline can take proactive action and prevent possible customer dissatisfaction. Actions such as a quicker alternative path to the gate, a lounge pass for a day, and rebooking on an alternate connecting flight can be taken by the airline.

This example shows how using patterns in available historical data and monitoring and analyzing current data within the system of insight enable the airline to automatically detect a situation and proactively manage the customer’s situation and ensure that he or she remains a loyal customer.



Systems of insight solutions

This chapter provides examples of real solutions that meet the definition of a system of insight. The solutions presented here are expressed in terms of a typical high-level architecture and a mapping to the systems of insight model described in Chapter 3.

These solutions are classified into *retroactive*, *real-time*, and *proactive* categories. These titles apply depending on when a situation takes place relative to when the insight is realized.

- ▶ For real-time solutions, the situation is occurring *now* and action can be taken immediately.
- ▶ For retroactive solutions, the situation has occurred in the past. This experience is then used to take future actions.
- ▶ For proactive solutions, the situation is expected to happen in the future. Predictive analytics is used to improve decision outcomes.

This chapter covers the following topics:

- ▶ Real-time solutions
- ▶ Retroactive solutions
- ▶ Proactive solutions

2.1 Real-time solutions

In addition to the traditional intelligence available to analysts and managers through retrospective descriptive analysis, systems of insight can enable real-time knowledge, understanding, and observations to be made about how a business is operating.

This section describes system of insight solutions that provide analytic or deterministic actions in response to what is currently happening.

For clarity, in this section “real-time” is used to describe actions that are taken within a few seconds after the detection of a situation. This is sometimes referred to as human-real-time, where the reaction time is imperceptible to a human or is dominated by the time it takes a person to interact.

2.1.1 Event-driven situation detection

This pattern is concerned with the detection of situations that have never been identified before. In the past, it has been difficult and not cost effective to automate detection so that action can be taken in real time. Real-time detection allows you to improve business outcomes through the timely detection of threats and risks, and opportunities. Early warnings or quicker detections can help businesses improve customer service, reduce issues, or manage their resources more efficiently.

The situation detection works by identifying a pattern that corresponds to the situation. For example, the situation is defined by the occurrence (or non-occurrence) of a set of events within a time period. These event patterns are expressed using a set of business rules that describe the detection logic.

For example, in credit-card fraud, detecting that cash withdrawals occurred in London at 10:00, and in San Francisco two hours later is highly suspect! To detect this situation, configure the situation detection platform to do the following tasks:

- ▶ Listen to credit card transactions (the *events*).
- ▶ Correlate and store events by credit card.
- ▶ Look for a pattern in the events in real time as each transaction occurs.

A business policy (expressed as a rule) describes this pattern and the threshold for the distance between locations above which an alert is triggered.

This category of technology solutions is sometimes referred to as complex event processing (CEP), or business event processing (BEP). However, solutions such as the Decision Server Insights component of IBM Operational Decision Manager Advanced (ODM Advanced), add advanced capabilities such as aggregation, global analytics, predictive analytics, and geospatial operators.

Figure 2-1 shows the basic situation detection architecture that follows a **Consume** → **Detect** → **Decide** flow.

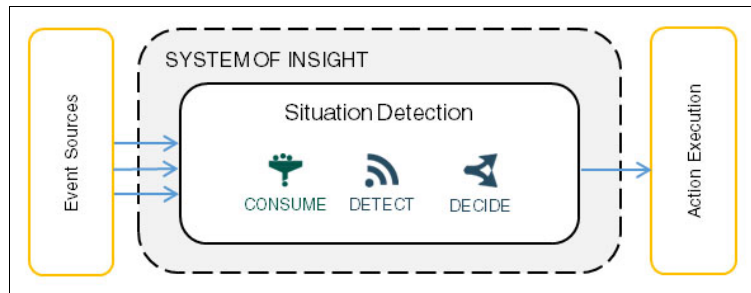


Figure 2-1 Situation detection as a system of insight

Consume refers to the capturing and filtering of events and building the event context. *Detect* refers to rule-based pattern matching and correlation and *decide* corresponds to the action statements in rules that change states and emit events.

This architecture is detailed in Chapter 6.

Event driven situation detection can be driven by rules alone, or can also include predictive analytics, stream processing, and decision automation. Each of these topics is discussed further in this chapter.

Data used in these types of solutions mostly come from “data in motion” sources in the form of real-time events and transactions, although it is possible to process batch transactions if they produce temporal records. If necessary, data is augmented from systems of record and analytical scores through an enrichment mechanism.

Actions are generated by events that are triggered when a predetermined situation pattern is matched. These actions can be either automated or deferred for further human investigation, for example in response to a notification. Notice that in the system of insight model (unlike some other models), action execution is not a property of the system of insight. Rather, it occurs in the target system. By contrast, consumption of the inbound events is internal to the system because you need to filter, transform, and store events so that you can build context in support of the situation detection. Event sources are also external to the system.

Situation detection and decision automation are closely linked. However, the logic that is required to detect a situation is usually distinct from the decision automation logic. Consider a next best offer scenario where a store wants to offer a promotion to a customer depending on their profile and real-time interactions with the store. Deciding when to offer the promotion and what promotion to offer are distinct sets of logic. The current situation (for example, the customer is making a purchase) can affect the choice of promotion to be made. Chapter 6 describes situation-driven decisions that cover both the detect and decide capabilities.

The IBM developerWorks® article by Andy Ritchie and Dan Selman on *Event Driven Architecture and Decision Management* describes the relationship between situation detection and decision making in more detail:

<https://developer.ibm.com/odm/docs/odm-capabilities/odm-advanced-decision-server-insights/event-driven-architecture-and-decision-management/>

This article also proposes the following more detailed capabilities list for event driven architectures. For clarity, the corresponding steps in the system of insight model presented in this book are provided in parentheses.

- ▶ Event Sourcing (*Data and Transactions*)
- ▶ Event Distribution and Filtering (*Consume*)
- ▶ Event Collection and Analysis (*Collect, Analyze*)
- ▶ Detection of Situations and Decision Making (*Detect and Decide*)
- ▶ Taking Action (*Action Execution*)
- ▶ Reporting and Monitoring (*Analyze*)

Solutions that provide event-driven situation detection capabilities include IBM Operational Decision Manager Advanced and IBM Interact.

2.1.2 Decision automation

Providing an automatic way of running a deterministic or analytical decision improves the speed of business.

Decision automation can include deterministic and analytical decisions across multiple use cases such as pricing, eligibility, validation, underwriting, classification, lending, promotion, prioritization, and routing across all industries.

Typically, decision automation solutions are designed to allow request response integration with an application. More mature implementations use a specialist decision management solution to encode the decision logic in a form that is independent of any application. This allows for reuse, business visibility, decision governance, and rapid change of the logic.

A decision service (the encapsulated form of a decision as an invocable operation) alone is not a system of insight. However, it regularly plays a part in situation detection, and increasingly is closely associated with analytical techniques. For example, you can use analytical techniques to determine what the decision logic needs to be, then implement it through a decision service. This combination of analytics and decision automation meets the definition of a system of insight. The case is strengthened further where the analytical tools are used regularly (or even continuously) to update and tune the business rules captured within the decision service.

Figure 2-2 shows the relationship between the analyze component and the decision component. Here the dotted arrow between *Analytical Modelling* and the *Decision Service* represents a link that is not necessarily automated, that is, the analysis phase can simply inform the design of the decision service.

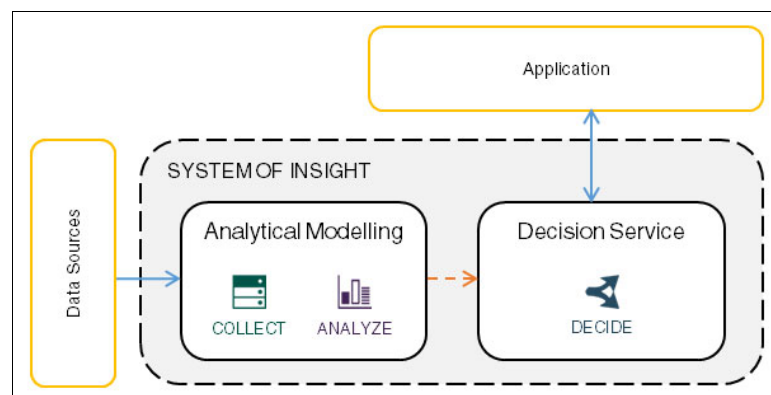


Figure 2-2 Analytics driven decisions as a system of insight

The action in this context is a synchronous response to a request from an application calling a decision service.

This pattern uses the Collect, Analyze, and Decide components. In this example, the application provides both *data* and the *action execution* components from the model.

For more information about automated actions using business rules in the Decision Server Rules component of IBM Operational Decision Manager, see 6.2, “Request-driven decisions in ODM Decision Server Rules” on page 73.

Solutions that provide decision automation capabilities include IBM Operational Decision Manager, IBM Analytical Decision Management, and IBM Interact.

2.1.3 Business activity monitoring

Business activity monitoring is the discipline of monitoring and visualizing the operation of a business. Typically, it is realized through the correlation of events from a system (often multiple systems) to determine a real-time status of a business process. This status is shown in charts and dashboards that are role-specific.

For example, in a retail-banking scenario, a team lead, regional manager, and company director want to know how their mortgage application process is performing. Each individual has access to different information:

- ▶ The team lead's dashboard is likely to show KPIs related to the turnover rate of tasks performed by her team and is alerted to tasks that require special attention such as those that are likely to become overdue.
- ▶ The regional manager wants to know how various branches compare in his region and where bottlenecks occur.
- ▶ The director wants to know about specific KPIs related to the mortgage approval rate and sales performance.

In each example, the data and KPIs available to each person are specific to their role, and are designed to help them to be able to answer pertinent questions. Specifically, the value for the individual is in being able to make a real-time decision and take associated action:

- ▶ The team lead can reallocate tasks or reprioritize.
- ▶ The regional manager can make changes to the process.
- ▶ The director is alerted when the business' exposure exceeds a safe level.

From this perspective, dashboards and real-time reports are a system of insight. The solution described above is likely to rely on data from multiple systems spanning a complex IT landscape, and requires a specialist business activity monitoring solution (Figure 2-3). However, many solutions provide insular visibility into business operations for this purpose. Examples of solutions that contain elements of business monitoring include business process solutions, industry-specific applications, and technical system monitoring solutions.

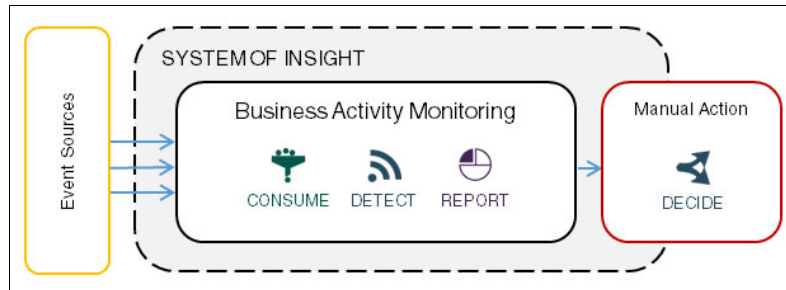


Figure 2-3 Business activity monitoring as a system of insight

From an IT perspective, this pattern follows Consume → Detect → Report. The *decide* and *analyze* functions are performed by the user.

Solutions that provide business activity monitoring capabilities include IBM Business Monitor and IBM Cognos Insight, IBM Cognos Business Intelligence, and IBM Business Process Manager.

2.1.4 Stream processing

Stream computing continuously aggregates and analyzes data in motion to deliver real-time analytics. This capability enables organizations to detect risks and opportunities in high velocity data that can only be detected and acted on at a moment's notice. High velocity data from real-time sources such as market data, Internet of Things, mobile, sensors, click stream, and even transactions are common targets for stream processing.

Figure 2-4 illustrates a stream computing architecture that facilitates real-time decisions.

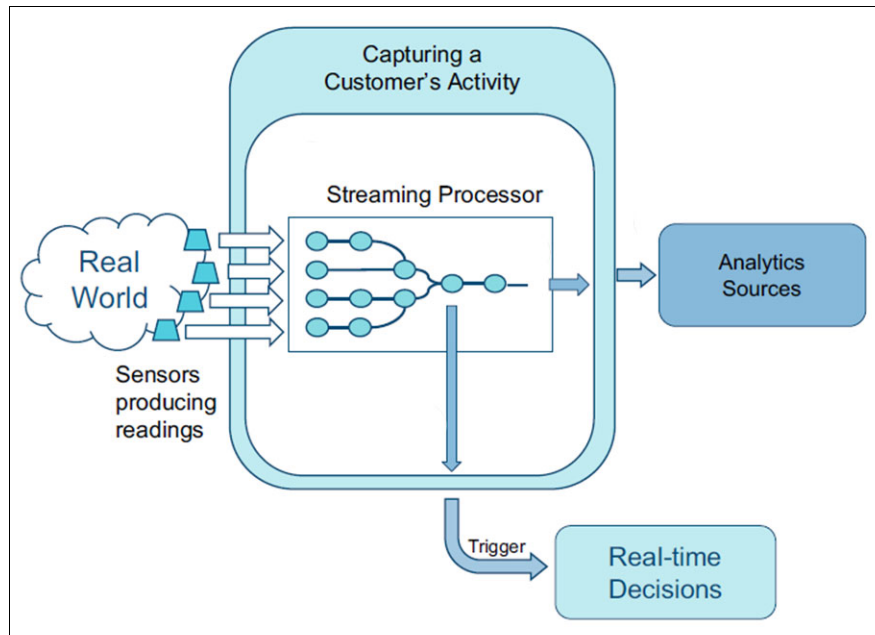


Figure 2-4 Stream processing for real-time decisions

In contrast with event-based situation detection, streams processing typically has a small time window (a few seconds) and so has limited historical context. Furthermore, stream processing can handle both structured and unstructured data. These properties make stream processing especially useful in scenarios where you need to filter data streams and generate higher-level business events that might be of use to a situation detection solution.

Stream processing use cases are found in medical device monitoring, financial market analytics, and facial recognition for security applications.

Stream processing follows a **Consume** → **Analyze** → **Detect** flow from the system of insights component model.

This flow often also includes a *decide* capability that can be an architecturally distinct component. It is also common for stream processing to act as a source for a further system of insight and in the advanced situation detection architecture. Figure 2-5 illustrates the pattern of stream processing as a system of insight.

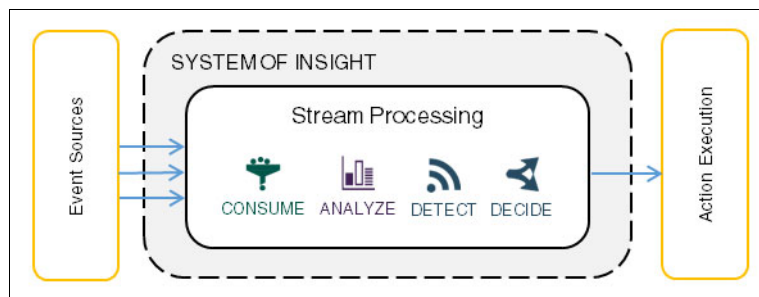


Figure 2-5 Stream processing as a system of insight

For more information about this pattern, see the following documents:

- ▶ *IBM Patterns for Operational Decision Manager in Big Data streams*, found at <http://www.ibm.com/support/docview.wss?uid=swg21686645>
- ▶ *Smarter Analytics: Driving Customer Interactions with the IBM Next Best Action Solution*, REDP-4888

Solutions that provide stream-based processing include IBM InfoSphere Streams.

2.1.5 Next best action

Next best action (or next best offer) solutions strive to answer the question *what is the best course of action given the current knowledge about a customer?* This is often applied in call-center operations where a recommended action can be provided to a call-center agent for offering to improve customer satisfaction. The background about the customer comes from their profile, interaction history, and might even include speech and tone analytics. This technology is also in use to make highly customized product recommendations.

Next best action is also found in business operations scenarios, such as in exception management. In industries with high-levels of IT automation, such as financial services, system-generated exceptions are often deferred to teams of analysts for manual investigation and resolution. Increasingly, situation detection and business rules technologies are used to reduce the cost of manual resolution by determining the next best action automatically.

For example, business rules are used to spot patterns of exceptions and perform grouping of related exceptions into cases, after which aggregate analytics help to determine the correct team for resolution depending on case complexity, priority, and speciality.

Solutions that provide next best action capabilities include IBM Interact and IBM Operational Decision Manager.

Figure 2-6 provides a mapping of the systems of insight capability model to runtime components of the IBM next best action solution.

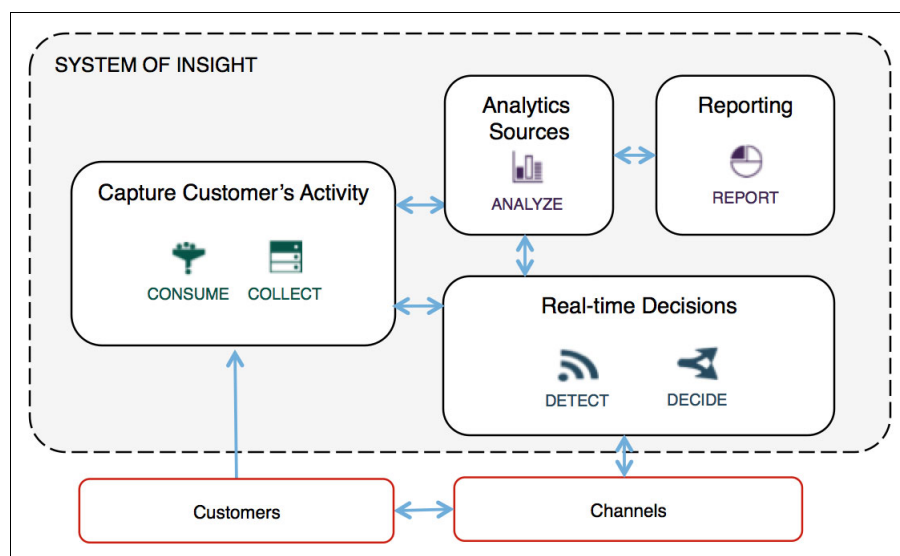


Figure 2-6 IBM next best act solution with systems of insight

For more information about IBM next best action solution, see *Smarter Analytics: Driving Customer Interactions with the IBM Next Best Action Solution*, REDP-4888.

2.1.6 Advanced patterns

Figure 2-7 illustrates an advanced pattern where situation detection is extended with other capabilities to provide continuous insight. This is the pattern that is used throughout this book to describe the capabilities of a system of insight in more detail.

The diagram represents an event-driven architecture with events coming from many heterogeneous event sources and where message transformation is handled by the integration layer. Some event streams are pre-processed through the stream processing component before it passes events on to the situation detection component. The scoring and decision service components are used during situation detection to perform more complex decision logic. Finally, the business activity monitoring component is configured to listen to all events of interest that pass through the system of insight and provides reports and dashboards.

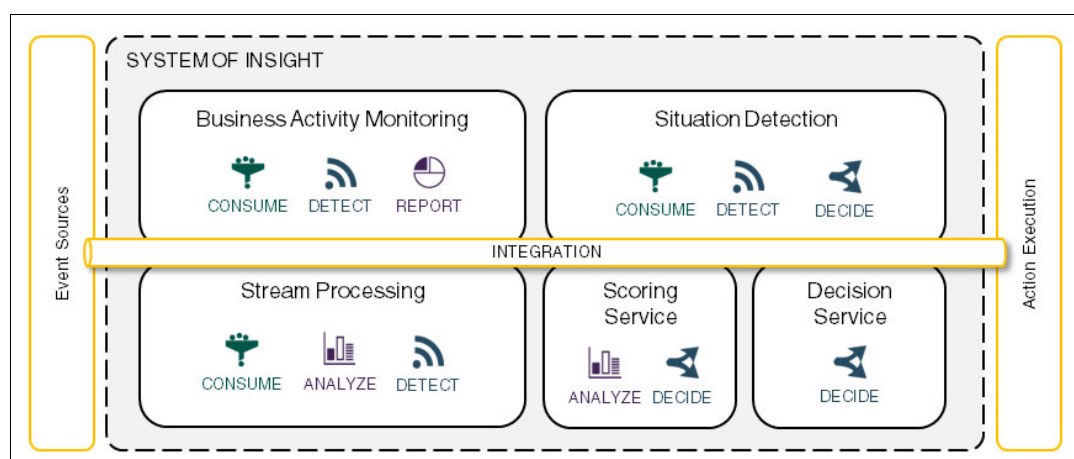


Figure 2-7 Continuous insight through an advanced situation detection solution

2.2 Retroactive solutions

Although real-time decisions are often preferred, many scenarios have computational constraints that require different use of human and analytic insight to respond to a situation from the past that cannot be detected any earlier.

2.2.1 Manual analytic investigations

This class of action relates to the application of analytics in a user-driven context, where an analyst uses numerous analytics tools to perform investigative work to test a hypothesis or respond to uncertainty.

Analytic investigations are typically found in fraud and security domains that use capabilities such as link analysis, classification, segmentation, and risk assessment. Their use typically spans cleansed data in an analytics data store in addition to unstructured data that can be weeks, months, or years old.

The insight garnered from these techniques is used with existing processes and applications (systems of engagement and systems of record) to query analytic scores before making final decisions about what to sell, building a case for prosecution, or deciding how to define pricing for risk-based assessments, for example.

Actions also include implementing new predictive models (see 2.3.1, “Predictive analytics” on page 23) or updating business rules (see 2.1.2, “Decision automation” on page 16).

Figure 2-8 shows the generic data flow that contains Collect → Analyze → Decide components from the systems of insight model.

The following types of tools are used in an analytic investigation solution:

- ▶ IBM i2® Intelligence Analysis Platform
- ▶ IBM Counter Fraud Management
- ▶ IBM SPSS Modeler with Scoring Adapter
- ▶ IBM Case Manager
- ▶ IBM Business Process Manager

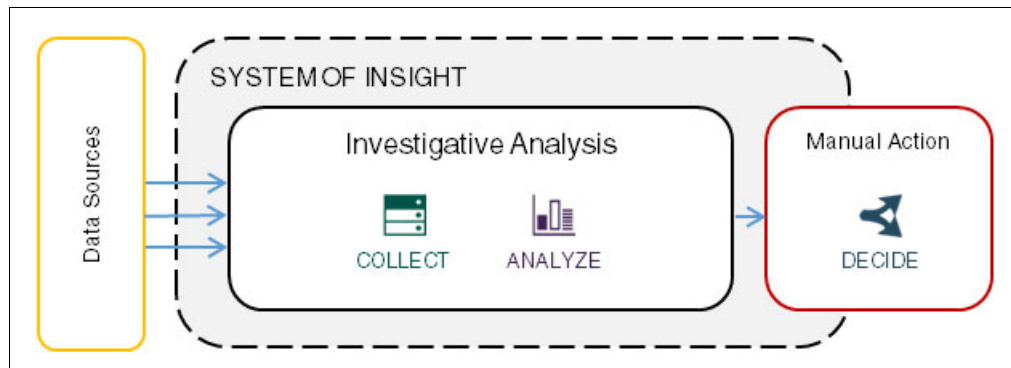


Figure 2-8 Investigative analytics as a system of insight

2.2.2 Big Data analytics

Big Data analytics platforms such as Apache Hadoop and Apache Spark allow for efficient analytics of large structured and unstructured data sets. This is achieved through high levels of parallel processing across many distributed systems. Figure 2-9 shows the basic layout of a Big Data analytics system.

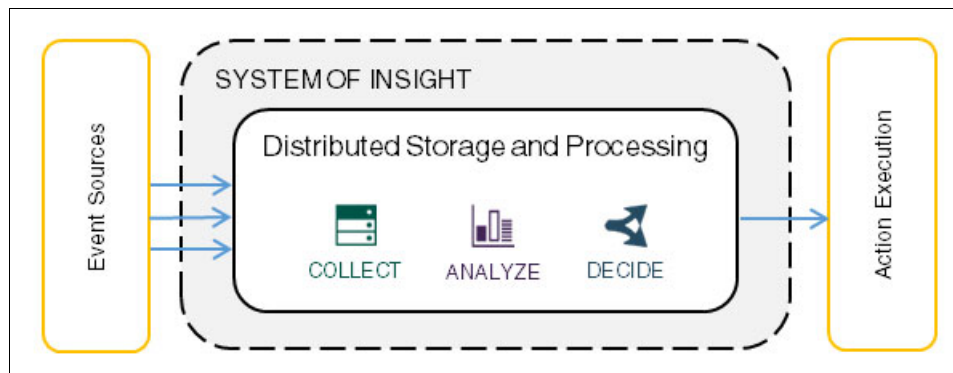


Figure 2-9 Big Data analytics as a system of insight

Although usually restricted to analysis of static data sets, big data analytics can be combined with other real-time analytic techniques such as stream-based processing to extend the computation to data in motion as well.

Actionable insight from big data analytics comes in the form of reports or by event emission that can be used to trigger actions (Figure 2-9).

Big data analytics can take many forms, but in general follows a Collect → Analyze → Decide data flow.

IBM provides big data analytics through the IBM InfoSphere BigInsights solution.

Big data Analytics is explored further in 4.2, “Collecting data at rest” on page 47.

2.3 Proactive solutions

This section describes classes of systems of insight where the insight is anticipatory of a future occurrence.

2.3.1 Predictive analytics

When using predictive analytics, a commonly automated action is to expose a predictive model through a scoring service. The scoring service might be started through a system of engagement (such as a business process and portal), batch process, or an event-driven solution.

Typical scoring services compute a segmentation score, churn score, risk score, or propensity score. For example, to answer questions of the form “what is the likelihood that...?” or “with what certainty will...?”.

More detail about predictive analytics is included in Chapter 5.

When a final score is reported (and often combined with results from deterministic decisions like business rules), there are some use cases where actions are fully automated, such as next best action or next best offer.

The pattern in Figure 2-10 follows a Collect → Analyze → Decide flow, where the application uses the scoring outcome as part of a decision.

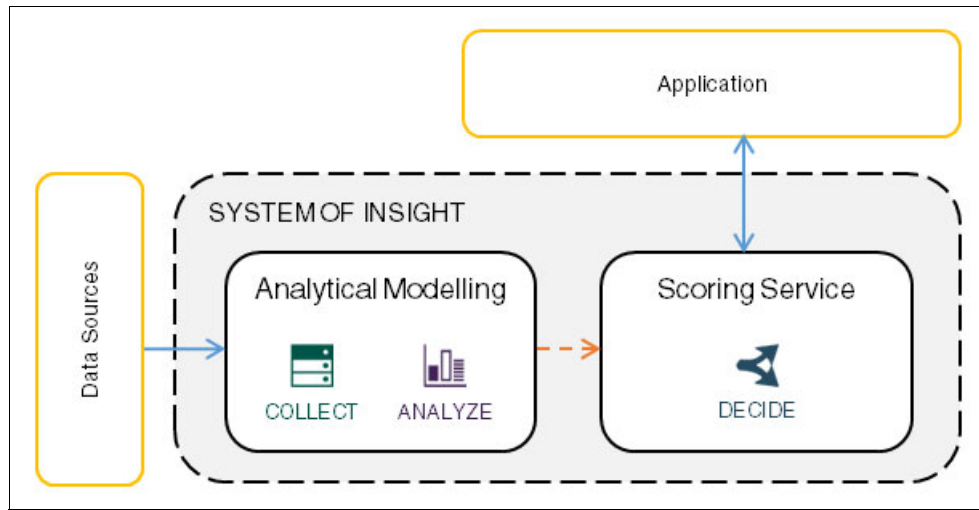


Figure 2-10 Predictive Analytics as a system of insight

IBM SPSS Modeler, SPSS Collaboration and Deployment Services provide analytical modeling and scoring services respectively. Predictive analytics capabilities are also provided by IBM Watson™ Analytics, IBM Predictive Insights, IBM Kenexa® Predictive Retention, and IBM Predictive Maintenance and Quality.

2.3.2 Prescriptive analytics

Prescriptive analytics extend predictive analytics to also provide actions that can take advantage of predictions. Two common implementations of prescriptive analytics are mathematical optimization and machine learning.

With optimization, a solution aims to find an optimal set of outcomes for a situation. From a systems of insight perspective, the optimization becomes actionable when it is used to inform business decisions or as part of system design.

Machine learning is a field of artificial intelligence and has applications in multiple solutions such as language translation, automated question and answer, computer vision, game play, and search. Generally, machine learning is made actionable through the implementation of a statistical model that, when trained, can provide answers on demand for external applications (Figure 2-11). Specifically, with machine learning, the models are expected to improve over time as they are provided with feedback.

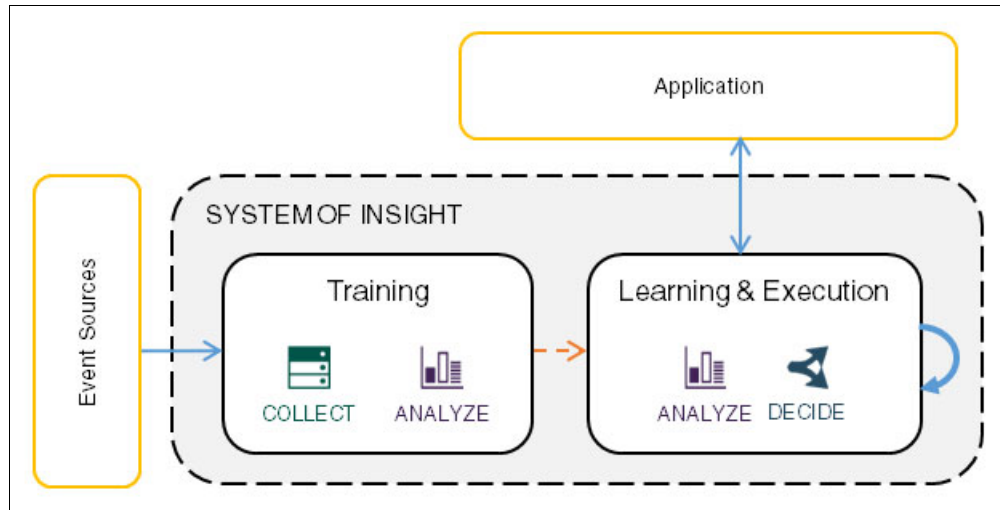


Figure 2-11 Machine learning as a system of insight

Prescriptive analytics covers the Collect, Consume, Analyze, Detect, and Decide capabilities in the systems of insight model. Examples of solutions used for prescriptive analytics are IBM Decision Optimization Center, CPLEX Optimization Studio, and numerous tools that are associated with IBM Watson.



“A model for systems of insight

This chapter introduces the capability model for a system of insight. It focuses on the abstract capabilities or building blocks that your organization needs to consider to gain more insight from the data it has.

This chapter also introduces capability categorization for a system of insight, and then describes the capability of each category in detail.

It also describes the importance of making these insights actionable and the integration strategies relevant to a system of insight.

This chapter covers the following topics:

- ▶ Systems of insight capabilities
- ▶ Consume and collect
- ▶ Analyze and report
- ▶ Detect and decide
- ▶ Integration

3.1 Systems of insight capabilities

Chapter 1 introduced systems of insight and how you can address business challenges by bringing systems of record and systems of engagement together, analyzing historical data, and applying predictive analytics to the current situation. This chapter introduces a model that helps you to define more precisely what a system of insight is. This chapter uses this model to explain the key capabilities or building blocks that a system must offer to be considered a system of insight.

The system of insight capability model includes the following components:

Consume and collect	Consumption of structured, unstructured, and transactional data, such as events.
	Collection of data for business analytics to gain new insight and drive better business decisions.
Analyze and report	Analyzing data or decisions answers questions such as why it happened, what is the trend, and what will happen next. It generates insights that can improve the quality of business decisions.
	Reporting tells you what happened, how many times it happened, and so on. It includes querying, batch, and real-time reporting.
Detect and decide	The ability to detect a situation using correlation with things that have or are happening to determine when to make a decision and take action.
	Decisioning is where business policies are implemented to dictate how you want to deal with the situation. The business policies can be derived from the results of analysis, company best practices, or the need to adhere to industry regulations.

Figure 3-1 shows these capabilities as part of systems of insight and the associations with external data, transactions, and action execution systems. Related capabilities are grouped for ease of illustration in the subsequent chapters of this book.

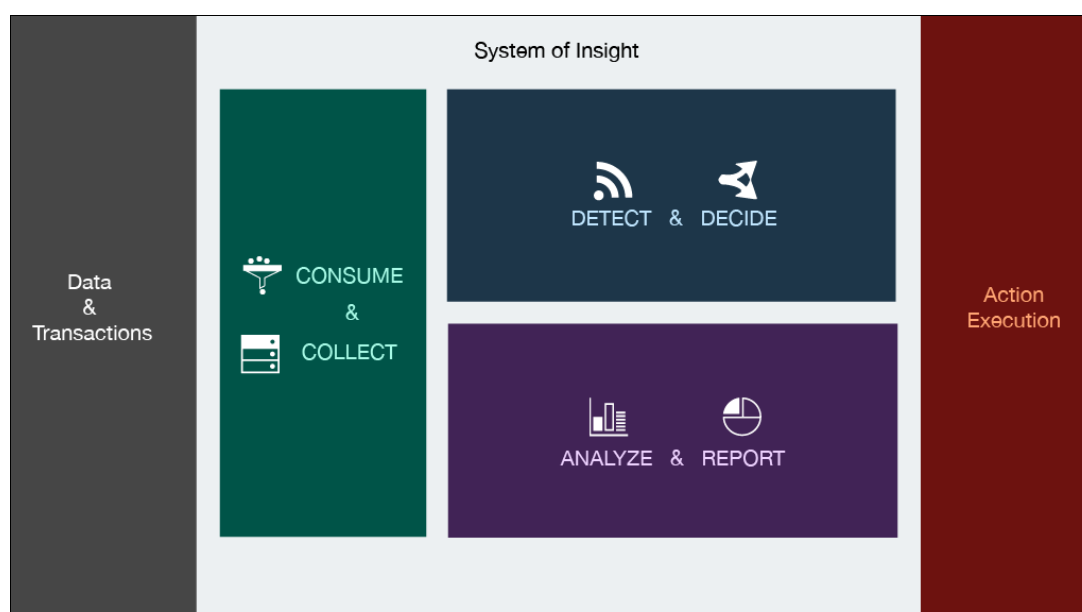


Figure 3-1 Capabilities of systems of insight

There is no specific order implied in Figure 3-1. In fact, depending on the solution context or the maturity of the solution, different combinations and sequences of the building blocks can be chosen to implement such a system. Some blocks might be repeated.

At a minimum, systems of insight exhibit capabilities from *consume and collect* and at least one of *analyze and report* and *detect and decide*.

Finally, this model should be considered a superset of some other models that exist today that have been applied to individual solutions and offerings. Although each model is valid in the context of its associated solution, the systems of insight capability model presents an abstract view that can be applied to multiple solutions. Common solution-specific models include the following types of models:

- ▶ IBM Decision Server Insights: Sense, Build, Decide, Act
- ▶ Enterprise Analytics: Transact, Transform, Integrate, Report, Analyze, Act
- ▶ IBM Counter Fraud Management: Detect, Respond, Investigate, Discover
- ▶ Enterprise Marketing Management: Awareness, Decisioning, Execution

Many of these product offerings span multiple capabilities and can be considered systems of insight in their own right (for example, Decision Server Insights). However, others clearly fit only in one category and multiple components are required to build a system of insight. Chapter 2 illustrates how many of the IBM product offerings map to the model presented in this chapter.

3.2 Consume and collect

Gathering data is one of the most important steps as it is the source of actionable insights. This section discusses the capabilities of consuming and collecting data as part of the system of insight.

Chapter 4 details examples of technologies that are used to collect and consume data and transactions.

3.2.1 Consume

This step is where you determine what data is important when making a business decision. *Consume* refers to extracting pieces of information from the transactional systems that will be used as part of the decision-making process. For example, you might consume credit card transaction data from different channels such as ATMs, online shopping, and brick and mortar retailers, so that you can find suspicious patterns in these transactions and detect possible fraud up front.

Data grows in volume constantly because it is gathered, increasingly cheaply, by mobile devices, sensors, cameras, and even social media. It is crucial for organizations to be able to harvest this data and derive meaningful outcomes that can be put into action. Usually, the intended business goals determine which data should be consumed. For example, if the goal is to raise an alert to doctors when the blood sugar level of a type-2 diabetes patient is measured as high with remote sensors, then the patient's medical history and the filtered readings from the remote sensors need to be consumed to determine whether an alert is necessary.

The volume of data of interest varies depending on the application. For example, getting temperature readings from thousands of sensors in a rocket is a vastly different pattern from what is needed for a processing medical insurance claim. It is important to pick the correct technology to consume the data. Typically, for high-velocity high-volume data such as traffic data or unstructured data from sensors (for example, images and videos), it is more efficient to use event streaming technology so that the data can be filtered and summarized into smaller volume higher value data that is more business oriented. For example, video feed of vehicles on highway can be summarized into the vehicle flow per second. The decision whether to reroute the approaching fleet of trucks can then be based on that traffic flow rate.

The format of data to be consumed is usually categorized as an *event*. It carries basic temporal information of when the event occurred and optionally a geospatial element to describe where the event occurred. It also often associated with a business entity such as the customer account or the flight number, so this association must be resolvable such that you have the full context when the event is processed later.

After you decide the specific events to be consumed from the sources, it might require an integration layer to route or transform the events, or require some specific filter, enrichment workflows, or mediations so that they can be effectively and efficiently processed by the system of insight.

Example solutions that grant functionality in this capability are IBM Integration Bus, IBM InfoSphere Streams, and IBM Operational Decision Manager. More detailed implementation scenarios to show how the products can be used together to consume the data are provided in Chapter 8.

3.2.2 Collect

The *collect* capability refers to collection and retention of data. This data is generally collected to perform analytics and aggregates upon it to identify patterns for generation of business logic or predictive modeling.

As an example, you collect the spending data of a customer over time and gain understanding of the user preferences and interests so that a customized promotion can be offered to the customer at the correct time.

When putting the data in a data store for further processing, it is important to cleanse and enrich so that more accurate analysis can be performed.

Example solutions in this capability include IBM DB2 and IBM InfoSphere BigInsights.

3.3 Analyze and report

Analyzing and reporting on the data captured can provide useful insights such as predictive behavior of certain customer profiles or alerting for action when things went wrong. This section introduces some basic elements of these two capabilities.

3.3.1 Analyze

Data that you collect is useless unless you can make some sense of it. This is achieved by performing analysis. The simplest form of analyzing the data is by applying human knowledge that you have about your business domain. For example, when two credit card transactions occur close together on the same account in two remote geographic locations, you know this is impossible, and warrants a fraud investigation. This human knowledge can be used to block any future suspicious transactions that exhibit this nature. This kind of analysis relies heavily on the domain knowledge of business experts.

Predictive analytics are also used here to decide what is the next best action to take. It analyzes the historical data to identify trends and patterns that can then be used for batch or real-time automated decision. For example, depending on account type and the customer's spending profile, you can categorize whether the customer is a highly valued customer or not. You can then use this knowledge to make business decisions that can help in ensuring retention of the customer.

Big data analysis allows insight discovery on high volume, high speed, and high variety of captured data. Big data is able to digest data with lower information density, predict behavior with algorithms, reveal hidden relationships, and show causal effects. The goal is to create actionable data, then deliver it when a situation arises that needs it.

Data can come continuously in the form of streams. For example, the data feed from connected devices, sensors, infrastructure, or applications. The analytics are done dynamically because it is costly to store this data and it will not be responsive if analytics are done later. Stream based analytics allow you to report on what is happening, current trends, and provide comparison with historical values, then act immediately upon the findings so that automated decision can be taken.

Solutions such as IBM SPSS, IBM InfoSphere BigInsights, IBM PureData™, and IBM InfoSphere Streams can be used to analyze the data that you collect and provide input for making better business decisions.

3.3.2 Report

You often draw insights by analyzing reports, either from historical or real-time data. The insights are then used to improve the decision-making process, capturing more opportunities and improving customer experiences. The outcome can be used to create new policy that can effect how the decision is made. For example, a report of total approved loan amount in a financial institute is correlated with the company's stock price so that the risk can be maintained at a low level. When the stock price is too low, it is risky to approve too many loans. By looking at dashboards or reports, business owners can response quickly by revising the corporate loan approval policy.

The report can be generated from data at rest, data in use, or data at motion. Reporting data in motion is much more responsive because it is processed in-flight and the report is generated in real time. Data at rest is typically processed at set intervals, which makes reports from old data. IBM Business Monitor and IBM Cognos Business Intelligence are example IBM solutions for reporting trends, measuring past business performance, and getting real-time alerts when unexpected situations occur.

3.4 Detect and decide

To use the actionable insights that often come from the analysis and reporting phase, infrastructure needs to be in place that can detect situations and then trigger decision-making processes. This section introduces the *detect* and *decide* capabilities with some examples.

3.4.1 Detect

The *detect* capability refers to the detection of a situation that is interesting to your business. Detection generally implies a decision will be made, so you can assume that any system of insight will have a decision as part of it, whether manual or automated.

The qualification of when a decision needs to be taken requires careful consideration of the context: What is the new piece of information the system knows and what has happened in the past. For example, if a fuel purchase transaction occurs using a company fuel card, but based on the real-time truck movement data (telematics), the distance traveled from the last refuel is still low, then this transaction could be a fraud. The driver could be buying the gas for personal use.

Detection is important for uncovering situations that might not be practical to do manually. You can achieve that by correlating events from different sources, as described in 3.2, "Consume and collect" on page 30. For example, events that contain the vital signs can come from 50 medical devices in intensive care unit (ICU) ward. While monitoring the current vital signs and correlating with historical measurements, something unusual happening can trigger a medical alert. It is difficult to scan through that huge amount of information for one patient and draw the conclusion manually. Doing it in real time, for many patients, is impossible without automation.

IBM solutions such as IBM Operational Decision Manager and IBM InfoSphere Streams can be used to detect situations that require business actions to be taken.

3.4.2 Decide

The *decide* capability refers to the core logic of how a business decision is made, taking the input of the detected situation, the context that was collected and the predictive behavior of the subject of interest.

A decision can be as simple as responding to a single piece of known information about a customer's status, such as notifying the customer when the account exceeds the weekly spending limit. It can also be a more sophisticated decision where it authorizes or declines a credit card transaction based on customer past spending history and the likelihood of customer making the purchase.

IBM Operational Decision Manager can be used to automate the decision-making process. It is a highly scalable platform that uses rule-based decisioning logic to respond to a situation detected in real time. The decision can be based on the knowledge held in in-memory business entity data, historical events, and the result of running predictive scoring models.

3.5 Integration

As illustrated in Chapter 2, the usefulness of a system of insight is determined by how well it is connected to source systems that provide data for analysis or processing, and how well it allows action to be taken in response to a decision or insight. This section covers the relationship between a system of insight and its supporting systems of record and systems of engagement.

3.5.1 Data and transactions

Data entering a system of insight comes in two forms:

- ▶ Data-in-motion
- ▶ Data-at-rest

In the former, data is presented as emitted events, a data stream, or through transactions. Each datum is filtered and transformed before entering the system of insight. This can be accomplished through various connectivity solutions depending on the scenario. Data-in-motion corresponds exclusively to *Consume* in the system of insight capability model.

With data-at-rest, the system of insight processes a static data source to elicit insight. Data stores might take advantage of an extract, transform, and load (ETL) process to import data from one or more existing data stores to convert the data into a format that is more appropriate for the chosen analysis. This corresponds to the *Collect* capability.

To complicate matters, many examples exist where you want to convert one form of inbound data into another. For example, to collect data from an event source and store it for retroactive processing, or to take a data store and emit events during updates to a database. In these cases, the boundary between systems that provide data and the *collect and consume* capabilities of the system of insight is blurred.

Pragmatically, consider components that expressly or uniquely provide collect or consume capabilities for a system of insight to be internal, and all others to be external.

Given the apparent overlap between *Data and Transactions* and *Collect and Consume*, this section presents solutions that provide integration with source systems in context of the collect and consume discussion in Chapter 4.

3.5.2 Integration within a system of insight

When you look inside a system of insight, you expect to see components responsible for the internal connectivity between different capabilities. In some systems, this is transparent because the integration is provided as part of a packaged application. However, in others, such as the advanced continuous insight example in Figure 2-7 on page 21, the integration component is explicit and necessary to provide a data flow between the component parts. The integration component does not naturally fit into any of the capability area for the system of insight model. Nevertheless, it is an integral part of the solution. In some circumstances, this additional *Integration* capability is necessary. It is convenient to represent the integration capability as an overlay that spans sources systems, each component of the system of insight and any systems where action takes place, as shown in Figure 3-2.

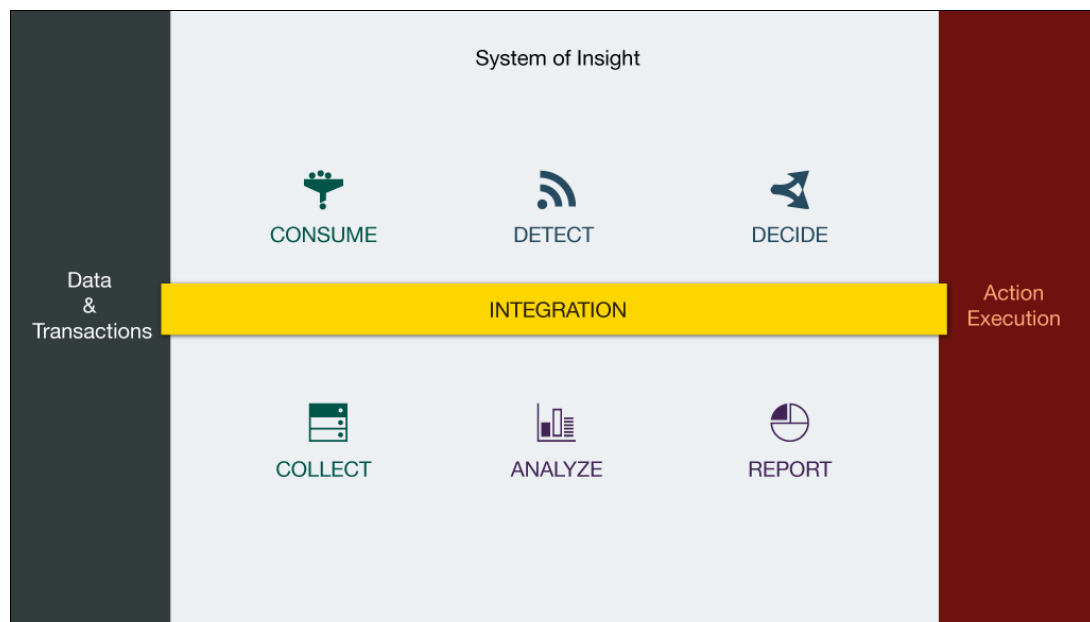


Figure 3-2 Integration layer within a system of insight

3.5.3 Action execution

When a business decision is made within a system of insight, it requires interaction with actionable systems to run the response to this decision. This action is the key step in extracting value from a system of insight, even if it is a manual step. For example, a subject matter expert using a dashboard to decide to offer promotions for a certain under-utilized service.

Actions can be as simple as sending a mobile notification to a traveler whose flight is delayed or as complex as opening a fraud case within a business process management solution and assigning a task to the fraud specialist.

Although action is enabled by insight gained within the system of insight, action execution does not need to take place within the system. In many solutions, such as Decision Server Insights, it is a logically distinct component. In others, particularly those that are considered applications, the solution contains features that permit action execution within the application. For example, the IBM Counter Fraud Management solution contains a case management component.

Furthermore, in many scenarios, particularly where insight is requested from a source application, the action execution also takes place in the same source application. That is, in the model in Figure 3-1 on page 29, the systems represented by *Data & Transactions* and *Action Execution* are the same.

To aid understanding, throughout this book, actions are described abstractly as the output of a system of insight, but the concepts apply regardless of composition of the solution.

Chapter 7 categorizes action execution into two types:

- ▶ Asynchronous interaction
- ▶ Synchronous interaction

Asynchronous interactions neatly fit the abstract perspective of an emitted action as an outbound event from a system of insight. These actions include the following examples:

- ▶ Notifications
- ▶ Initiation of a process
- ▶ Communication with devices

Commonly, all asynchronous actions appear to be non-deterministic in relation to source data. For example, you do not usually expect a one-to-one mapping between transactions and emitted action events.

Conversely, synchronous interactions are related to scenarios where the source and destination systems are the same. For example, a user or process waits for a response to a question from the system of insight before continuing.

Chapter 7 also lists approaches to outbound integration from a system of insight as they relate to action execution.

Consume and collect

This chapter provides an overview of how data is consumed or collected in preparation to submit it to a system of insight. Data is available in different formats and comes from different systems. Figure 4-1 shows the example systems of insight capabilities classification. This chapter focuses on *consume* and *collect*.

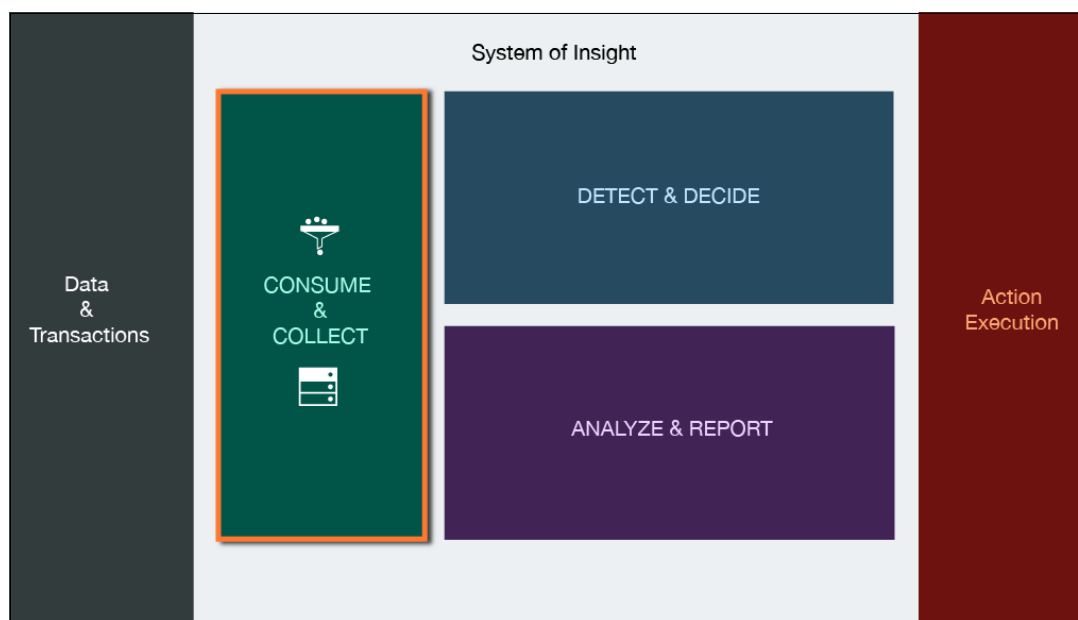


Figure 4-1 Systems of insight capabilities highlighting consume and collect

This chapter presents some patterns used to consume and process data as part of the systems of insight capabilities. In particular, it focuses on how data can be made available for Decision Server Insights for further processing.

This chapter covers the following topics:

- ▶ Consuming data in motion
- ▶ Collecting data at rest

4.1 Consuming data in motion

Consuming and collecting data is about gathering different types of data and obtaining business value from that data. Not all data needs to be treated the same, and not all data can provide value immediately. Sometimes, data must be collected and stored over time before it can provide knowledge or insight. This section describes data collection from applications, enterprise systems, mobile devices, and Internet of Things (IoT) for analysis, decision making, and action execution.

Figure 4-2 contrasts consumed data volume with data value, providing the evolution of the data value. In this case, data consumed refers to any type of incoming events. Figure 4-2 presents some of the tasks that are performed when data is received and what type of storage is required for the data.

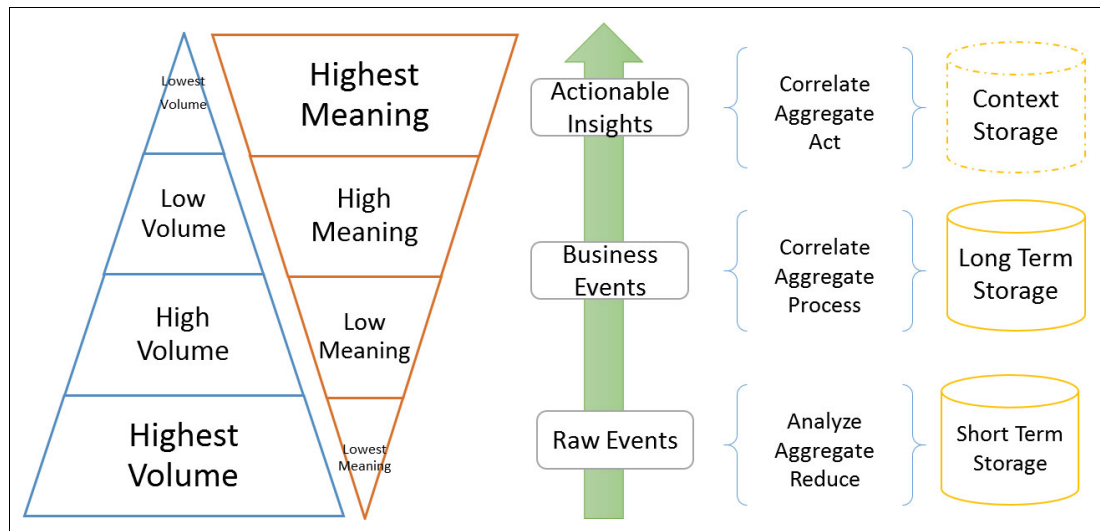


Figure 4-2 Volume versus value during event processing

The business value from high volume data is usually low because data is similar and repetitive, such as the temperature in a room. There is no business value when temperature is the same. The value comes from when changes in temperature are identified. Through data analysis, you can identify and discard unnecessary, irrelevant sets of events. The data value is increased with classification of data and at the same time, the data volume decreases. This smaller set of data is sent to other applications.

Preserving the data that has most value to the business is a challenging task. IBM offers products that support different connectivity protocols for screening and converting data to meaningful events.

For data in motion, high volume raw data usually has a shorter storage time. After the data is analyzed and processed, those high business value data will be stored longer for detecting business situations in which an enterprise is interested.

A simple example is collecting speed data from the sensors in a truck. The data volume is high because the data received can be a metric every second or even more frequently. This event data does not provide any significant value until there is a speed change exceeding a certain threshold. If a car slows down, it might be a traffic-related event or perhaps a malfunction. A transportation company might need to take action based on these new events. The action can be sending a tow truck, a mechanic, or a new truck to move the load and make it available on time for delivery. In the truck example, different types of events and different

types of processing are required before you can provide meaningful insights. Some processing might be related to decisions about what to do based on the truck being stopped, whereas another processing scenario might be related to the decision of whether or not to trigger a call to the truck driver.

4.1.1 Event sources

Events are everywhere from devices or systems to human interactions. This book covers events that are generated from devices, applications, or enterprise systems. Figure 4-3 shows a high-level overview of different capabilities that are typically performed by software products or hardware devices at the Integration layer needed to enable the consumption of data for a system of insight.

Figure 4-3 is divided into three vertical tiers starting from left with events consumed from connected devices (mobile phones, car sensors, and so on), enterprise applications like CRMs, Business Process applications, or legacy systems among others. To communicate these events, these devices rely on different protocols, such as MQTT, JMS, or HTTP. It is also important to mention that consumption can take place in an asynchronous or synchronous mode depending on the system needs.

The middle tier in Figure 4-3 shows capabilities performed by the integration layer to prepare data for a system of insight. These capabilities are performed by appliances (hardware devices), software products, or both. Depending on system needs, data might be transformed to add or remove values, transform fields, or permanently create a data store where insights can be obtained from.

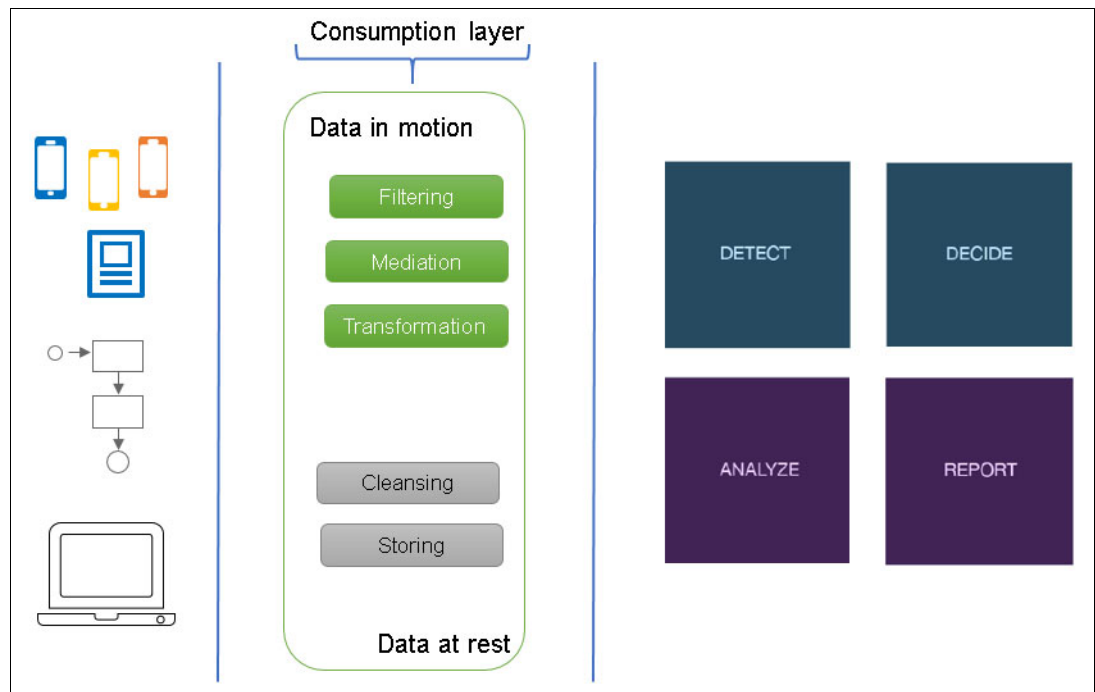


Figure 4-3 Events consumption layer

The rightmost tier in Figure 4-3 represents the systems of insights model to highlight the set of tasks that will take place after consumption has been performed.

The following sections provide some patterns to describe how the consume and collect process takes place with some IBM products to highlight capabilities that are presented in Figure 4-3 on page 39.

Protocols:

- ▶ IBM WebSphere® MQ Telemetry Transport (MQTT): Lightweight protocol that is used to connect devices. Ideal for Internet of Things.
<http://mqtt.org>
- ▶ Hypertext Transfer Protocol (HTTP): Foundation protocol for data communication in the World Wide Web:
<http://www.w3.org/Protocols/>
- ▶ Java Message Service (JMS): Standard for sending and receiving messages asynchronously:
<https://jcp.org/aboutJava/communityprocess/final/jsr914/index.html>

Internet of Things

The Internet of Things (IoT) is increasingly one of the major data sources for systems of insight. IoT refers to any electronic device connected to the Internet that is capable of sending any type of data it is responsible for capturing. Electronic devices in IoT are considered smart devices.

These types of devices include the following examples:

- ▶ Electronic sensors in a car to provide information about speed, outside temperature, objects around the car, and so on (connected vehicles).
- ▶ Home appliances connected to the Internet that can order groceries or send maintenance requests to manufacturers.
- ▶ Temperature sensors in every conference room for controlling the room temperature.
- ▶ Wearable devices from people that count the number of steps accomplished per day, or heart rate while sleeping or working.
- ▶ GPS devices in wild animals that provide location data that allows identifying migration patterns, or prevents accidents such as shark attacks in popular beaches.

From these examples, you can obtain an idea about how these connected devices can become a set of sources for events that enterprises are interested in.

With all that data (events) produced by IoT devices, the challenge is how to capture these events for analysis and decision making.

Internet of Things Foundation

The IBM Internet of Things Foundation is a solution that provides a cloud-hosted service to manage devices. IoT Foundation provides tools for device registration, connectivity handling, a security infrastructure, and storage.

Figure 4-4 illustrates these capabilities with the IoT Foundation in the middle, making sure that the devices can connect to applications through different APIs.

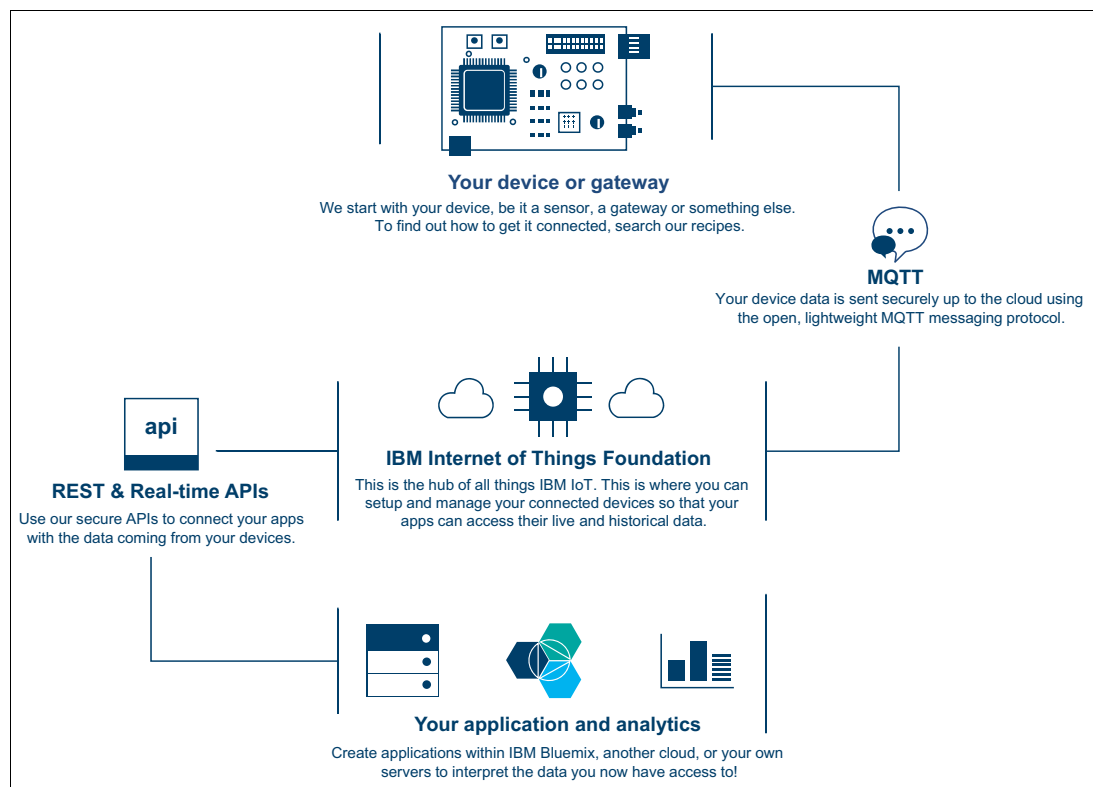


Figure 4-4 IoT Foundation understanding

IoT Foundation uses various technologies to connect to various systems, some of which are appliances that handle the large volume of data that these devices provide. Some of these devices are described in the next section.

More information about IBM IoT Foundation is available at the following web address:

<https://internetofthings.ibmcloud.com>

Gateway or messaging devices

The increasing number of devices available every day requires high throughput systems capable of consuming large chunks of data and make it available to be used by internal applications in the enterprise. One of the main concerns when consuming or collecting data is security. How can you guarantee that you are receiving data from the correct source? How can you guarantee that data is not compromised? And most importantly, how do you do this at the high throughput levels that are needed by applications in this new era?

This section describes some appliances with capabilities to address integration capabilities that are needed during the consumption of data.

IBM MessageSight

IBM MessageSight is an appliance that is designed for the IoT and mobile environments. It provides a secure, DMZ-ready channel for lightweight, rapid, bidirectional messaging. IBM MessageSight delivers the performance, value, and simplicity that you need to accommodate the growing number of mobile devices and sensors.

MessageSight provides these features:

- ▶ **Security:** Is De-militarized zone (DMZ) ready with no user level operating system.
- ▶ **Reliability:** Assures critical messages are delivered.
- ▶ **Developer-friendly APIs and libraries:** Enables native application development, including Android and iOS.
- ▶ **Integration:** Extends and connects to JMS, IBM WebSphere Application Server, and IBM WebSphere MQ.
- ▶ **Scalability and high performance:** Provides high throughput for persistent and non-persistent messages.

MessageSight uses MQTT, JMS, or message queues for connectivity and is considered the appliance of choice for IoT. In the event flow pattern (Figure 4-5), MessageSight is used to transport and route data from different IoT devices as the main integration transport layer. IBM MQ is also responsible for integration, queuing, and routing data to the system of insight. In this scenario, the system of insight also requires some dedicated event filtering and analysis to be done before the data is processed by the system of insight. This is highlighted in the green box. It can use the capabilities in InfoSphere Streams before the data can continue its flow to DSI capability inside the system of insight.

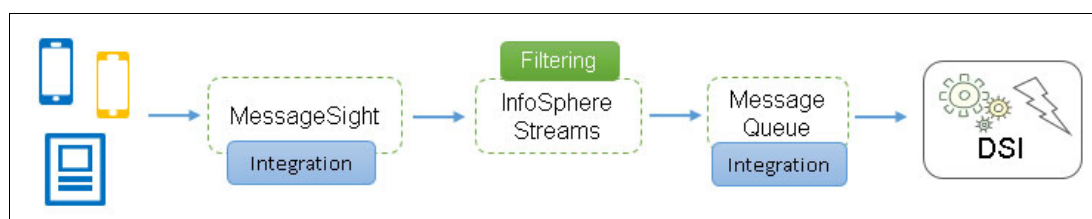


Figure 4-5 Events consumption with MessageSight, InfoSphere Streams, and IBM MQ

IBM Integration Bus integrates with the IBM MessageSight appliance to enable and manage high volume client access to enterprise applications, systems, and data. Figure 4-6 shows a consumption pattern. In this scenario, the system of insight also requires extra key filtering and mediation capabilities through IBM Integration Bus to ensure that the appropriate events are routed to the system of insight, which maximizes its efficiency and reduces latency.

For details about Enterprise Service Bus integration, see “Enterprise service bus” on page 44.

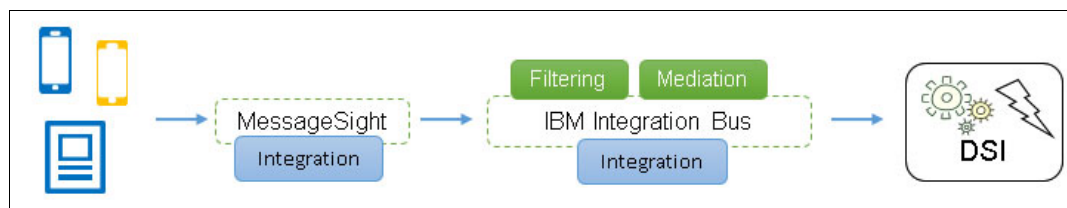


Figure 4-6 Events consumption with MessageSight and IBM Integration Bus

For more information about IBM MessageSight, see the product page at:

<http://www.ibm.com/software/products/en/messagesight>

For more information about MQTT and MessageSight, see these books:

- ▶ *Building Real-time Mobile Solutions with MQTT and IBM MessageSight*, SG24-8228
- ▶ *Responsive Mobile User Experience Using MQTT and IBM MessageSight*, SG24-8183

IBM DataPower Gateway

IBM DataPower® Gateway is a purpose-built security and integration platform for mobile, cloud, API, web, service-oriented architecture (SOA), and B2B workloads.

Among other things, IBM DataPower Gateway offers a single multi-channel gateway platform to secure, integrate, control, and optimize delivery of workloads across channels that include mobile, API, web, SOA, B2B, and cloud as well as integration with IBM MobileFirst™ and WebSphere platforms.

DataPower Gateway provides secured and optimized delivery of applications and services through transport-layer and message-layer encryption, rich authentication, granular authorization, auditing, and token translation. Other capabilities of DataPower Gateway include application optimization and integration including XML off load, caching, message validation and filtering, and transport protocol. These values make DataPower Gateway an ideal appliance option for data consumption in a secured environment.

For the flow pattern shown in Figure 4-7, DataPower is used with the MessageSight to provide a secure and robust high performance messaging system to consume and make the data available for an organization. Both DataPower Gateway and IBM MessageSight are part of the Integration layer and can be responsible for routing tasks.

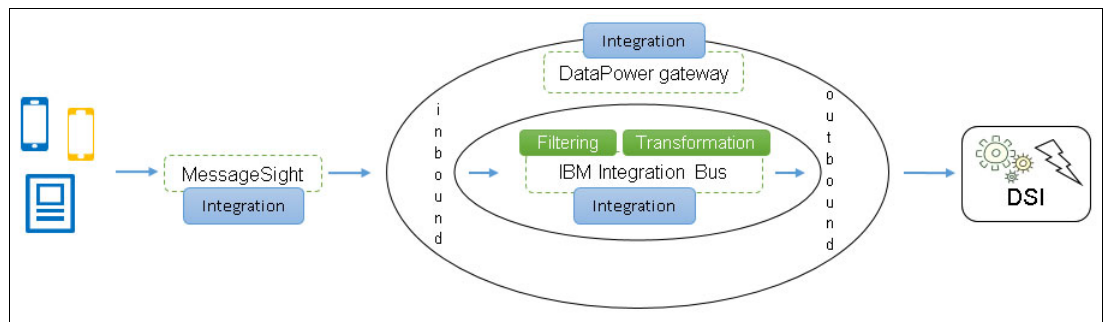


Figure 4-7 Events consumption with MessageSight, DataPower, and IBM Integration Bus

For more information about DataPower, see the product page at:

<http://www.ibm.com/software/products/en/datapower-gateway>

IBM MQ Appliance

IBM MQ Appliance M2000 provides the application connectivity performance of IBM MQ software in a physical messaging appliance. It offers rapid deployment of enterprise messaging with easier administration. Performance and message throughput are optimized for the appliance's capability and configuration.

IBM MQ Appliance M2000 provides these features:

- Increases flexibility with single or multiple queue managers that participate in IBM MQ clusters and exchange messages with other queue managers or IBM MQ clients
- Improves message security with IBM MQ Advanced Message Security
- Delivers high availability through connected pairs of appliances
- Lowers cost of ownership through rapid deployment, and simplified administration and maintenance

This section presented different messaging devices and appliances and described some of their basic capabilities. Figure 4-7 on page 43 shows an enterprise usage example in which

MessageSight handles high volumes of data possibly from IoT devices, and is mainly responsible for the transport of messages.

Received data must pass a security gateway that is provided by DataPower Gateway, where some tasks can include routing, encryption or decryption tasks before it can be used inside the organization. DataPower Gateway acts as the inbound and outbound parts of this process.

IBM Integration Bus becomes the mediator inside the security zone of the enterprise to perform tasks like filtering, enrichment, transformation, and routing. IBM Integration Bus can access systems of record or systems of insight. Systems of insight solutions can define additional message workflows that filter, enrich, or transform the events to run on IBM Integration Bus efficiently and free these tasks from the system of insight.

4.1.2 Event distribution and filtering

Events distribution or consumption is key to all event processing applications. The event triggering applications must make sure that the events from source applications are available for their analysis and processing. Events can come from systems of engagement or be triggered by a system of record.

In *Event Driven Architecture and Decision Management*¹, Dan Selman and Andy Ritchie identify the following categories of approaches for event distribution and filtering:

- ▶ Publish and Subscribe messaging: This is the oldest and most traditional approach to consume data or events from applications. In this approach, filtering capabilities are built in as part of the enterprise bus capabilities. This approach is covered in “Enterprise service bus” on page 44.
- ▶ Point to point: This type of approach is used when connecting to applications directly through different protocols is required.
- ▶ Stream analytics: Ability to consume data in motion (streams of data) that must be processed and analyzed. For more information about how to use InfoSphere Streams to consume, analyze, and filter massive amounts of data continuously from different sources, see “Stream based processing” on page 46.

Enterprise service bus

An enterprise service bus provides universal integration between different systems and applications. In general, an enterprise service bus is a key component in any service-oriented architecture. A service bus is responsible for transforming standard and non-standard data to the correct format required by applications. A service bus simplifies complex operations related to collection, aggregation, and sequencing of data.

IBM Integration Bus is an enterprise bus offering that provides functions to support high transaction volumes and quality of service. IBM Integration Bus is useful for integrating on-premises applications, existing systems, and custom data formats.

IBM Integration Bus has two key patterns of usage with IBM ODM Decision Server Insights:

- ▶ Transactions that are currently being processed by IBM Integration Bus can be instrumented to post a copy of the transaction to an IBM MQ topic. ODM DSI unobtrusively consumes events from that topic with no specific workflow or mediations required and minimal impact to the transaction that are taking place.

¹ By Dan Selman and Andy Ritchie at developerWorks at:
<https://developer.ibm.com/odm/docs/odm-capabilities/odm-advanced-decision-server-insights/event-driven-architecture-and-decision-management/>

- Where events are being routed from devices and transactions where the only consumer of them is ODM DSI. In this use case, a system of insight IBM Integration Bus message flow is created to transform and enrich the event or transaction into the format required by DSI and route it to the JMS or HTTP endpoint that DSI is using. This pattern is shown in Figure 4-8, where IBM Integration Bus is also responsible for the integration layer to transport messages from provider to DSI.

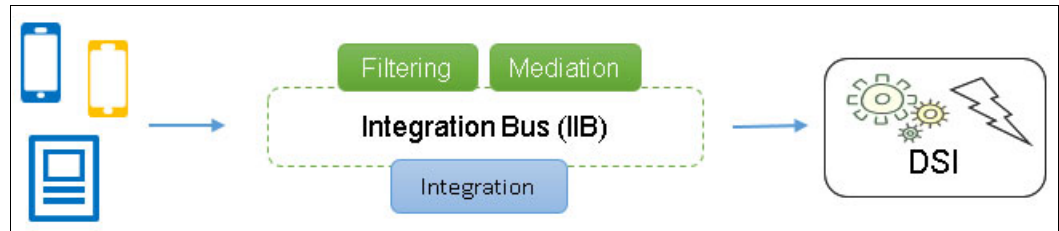


Figure 4-8 Events consumption with IBM Integration Bus

There are some usage patterns of IBM Integration Bus and IBM ODM Decision Server Rules (DSR). Figure 4-9 shows a consumption pattern where the flow of events passes through IBM Integration Bus. If the system of insight requires some smarter rule-based filtering and mediation task to allow rule based data enrichment based on some business decisions provided by DSR before passing these events to DSI for event processing, this is a useful pattern to choose. The system of insight architect defines the filtering or enrichment policies that the ODM rule-based decisions will implement as shown in Figure 4-9 in the DSR box.

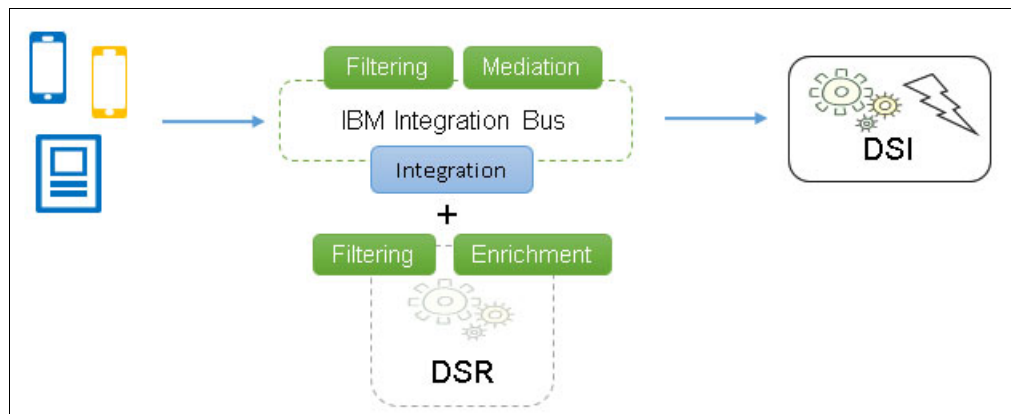


Figure 4-9 Events consumption with IBM Integration Bus and Decision Server Rules

The following are some of the utilization patterns of IBM Integration Bus and DSR of value to a system of insight:

- Message validation: Perform checks to events to make sure that they contain information that is needed by the system of insight. This can greatly reduce the work that must be done by the system of insight (SOI).
- Message augmentation: Ability to add more data to the received events that can be used later for events correlation or patterns identification in the system of insight. The more accurate and complete the event data, the better decisions and insight can be.

The combination of IBM Integration Bus with DSR and Decision Server Insights (DSI) is a perfect example of a system of insight capable of building context, taking decisions based on this context, and defining actions.

Stream based processing

The data in motion comes as “a stream” and the volume tends to be high. The traditional sequential processing paradigms cannot keep up with these high volume requirements, which can create a bottleneck in the system. To address this issue, a new processing paradigm, stream processing, was introduced. Stream processing provides parallelism in the processing that is used for the high volume of data or *streams* where streams of data or streams of events are continuously processed and analyzed for decision-making applications.

When working with streams of data, you face the following challenges:

- ▶ Volume of data to process (such as temperature sensors and geo-position data)
- ▶ Variety of data to process (such as structured or unstructured data like video, social media, and so on)
- ▶ Velocity of processing (such as one hour or one second)

Tackling these challenges is what real-time analytic processing (RTAP) software focuses on. This is where IBM InfoSphere Streams software comes into action.

IBM InfoSphere Streams is an advanced analytic platform that allows applications to quickly input, analyze, and correlate information as it arrives from thousands of real-time sources. The solution can handle high data throughput rates, up to millions of events or messages per second.

InfoSphere Streams has different capabilities to work on the analysis of data in motion:

- ▶ Geospatial toolkit: Useful to process geospatial data
- ▶ Text analytics toolkit: Useful for processing unstructured data
- ▶ Time series toolkit: Abilities to detect patterns and predict future values
- ▶ Social media accelerators: Useful for sentiment analysis
- ▶ Data mining toolkit: Use scoring services for real-time scoring

InfoSphere Streams provides the following tools for internet and database connectivity:

- ▶ Internet toolkit: Retrieve data through HTTP, FTP, and RSS
- ▶ Database toolkit: DB2 ODBC, and unixODBC, among others
- ▶ BigData toolkit: Integrate with Data Explorer or BigInsights

With collect, decide, and analyze capabilities, InfoSphere Streams can be used as part of a system of insight by itself. It can be used, for example, in the Medical field, to prevent early diseases based on real-time analytics and correlation on physiological data streams such as blood pressure and oxygen level. Another use of InfoSphere Streams is to provide trade suggestions in the finance industry in real time and fraud validation in suspicious trading activities.

InfoSphere Streams plays a key role in the filtering high volume of data and converting the data to be more meaningful with business value as illustrated in Figure 4-2 on page 38.

Figure 4-10 provides a consumption pattern with InfoSphere Streams that is responsible for the integration layer tasks of receiving and processing events. A system of insight can also take advantage of InfoSphere Streams to handle and provide filtering tasks to reduce the number of events that the SOI processes. It can also analyze incoming data and generate higher-level business events for the SOI to process.

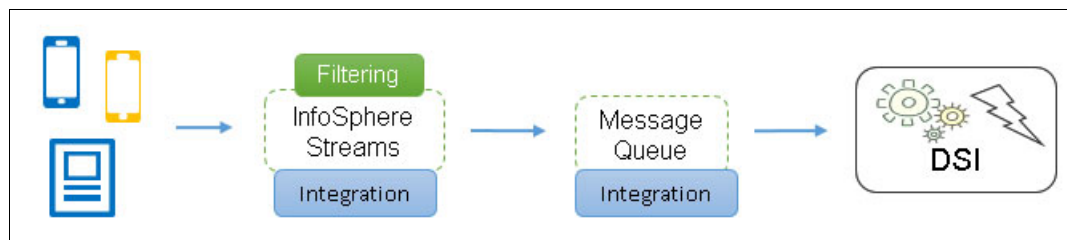


Figure 4-10 Events consumption with InfoSphere Streams and IBM MQ

In the scenario for Filtering, InfoSphere Streams supports the system of insight, whereas in the Analyzing scenario, it is part of the system of insight and generates events to other parts of the system of insight.

IBM MQ, as part of the integration layer, makes the events available for DSI in a JMS format to be consumed. This pattern is used in the implementation of the book sample scenario shown in Chapter 8.

InfoSphere Streams can also use DSR to handle complex decisions during the stream processing. It can also allow users to author rules in a simple English-like natural language that is managed and governed by a set of tools provided by DSR.

InfoSphere Streams is also described as part of the real-time solutions in 2.1.4, “Stream processing” on page 18.

For more information about InfoSphere Streams, see the product page at:

<http://www.ibm.com/software/products/en/infosphere-streams/>

4.2 Collecting data at rest

Every day, quintillion bytes of data is created. Most of the data in the world today was created in the last two years. This data comes from sensors that are used to gather climate information, posts to social media sites, digital pictures and videos, purchase records, and cell phone GPS signals to name a few. This data is big data.

It is often difficult or impossible to process big data at the speed that business needs with traditional storage techniques. However, data can now be stored in novel ways expressly for big data analysis.

Analytics using data at rest is covered in Chapter 5, which describes some techniques to combine decision management and data at rest.

4.2.1 Batch processing

The traditional way of transforming static data into an event source that is useful for situation detection is through batch processing. That is, an application iterates through a set of records (for example, files on disk, lines in a file, and rows in a database table) and applies processing

logic to each element. The processing logic might do some filtering or situation detection, but often, it starts a service through which further processing is performed.

For decision management in a request-driven context (that is, using Decision Server Rules for example), the service can be embedded within a Java application that does the batch processing (called through a J2SE rule session), or sent to a remote decision service through a service call.

In a situation-driven context (that is, using Decision Server Insights), the batch processing application sends events to be consumed by a system of insight.

Batch processing is also common in mainframe environments. The high reliability, availability, serviceability, and performance characteristics of such systems make them highly suitable for the storage and processing of high-value data.

For more information about the configuration of IBM Operational Decision Manager for batch processing with the zRule Execution Server on IBM z/OS®, see *Using IBM Operational Decision Manager: IMS COBOL BMP, COBOL DLIBATCH, and COBOL MPP*, REDP-4997.

For more information about multiple runtime configurations available for zRule Execution Server on z/OS, see *Flexible Decision Automation for Your zEnterprise with Business Rules and Events*, SG24-8014.

4.2.2 Database triggers

Alternatively, where data is located in a database, database triggers provide a convenient way of responding to data changes.

Database scripting languages (such as SQL PL or PL/SQL) allow the definition of triggers to call the stored routines when an insert, update, or delete occurs. When using triggers, the database must record the fact that data has changed in an event store, which is typically a database table. The event store is not the same as the application data.

Figure 4-11 shows the interaction between the database, event store, and the event processor. The data flow is as follows:

1. A database application changes a database table.
2. The database management system (DBMS) records the change in the event store.
3. The integration node polls the event store after a specified polling interval.
4. The event processor (in this example, IBM Integration Bus) retrieves the new or changed data, and updates the event store so that the data is processed only one time.
5. The event processor processes the data and ultimately presents it to the target application (for example DSI). The data can be presented in a different logical and physical format, if required.

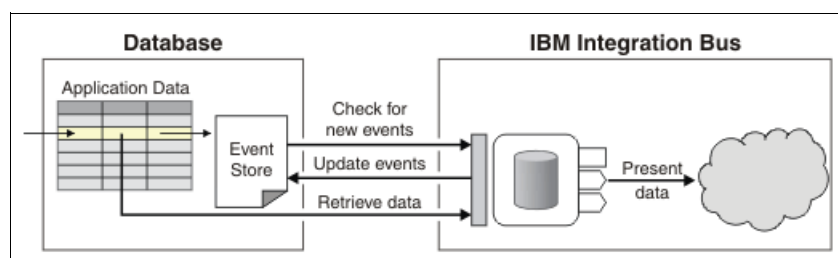


Figure 4-11 Using an event store to handle database triggers

For the steps to configure a message flow to respond to events from a database, see *Event-based database integration* in IBM Integration Bus Knowledge Center at:

https://www.ibm.com/support/knowledgecenter/SSMKHH_10.0.0/com.ibm.etools.mft.doc/bc34040_.htm

4.2.3 Big data and business rules

With data at rest, the goal is not to collect big data, but to act upon it.

By convention, big data processing uses scripting or programming languages to implement processing logic. For example, on a Hadoop platform, Java code applied in an IBM Platform Symphony® MapReduce framework allows highly distributed processing. Alternatively, query languages such as Big SQL in the IBM InfoSphere BigInsights platform make these large data stores more accessible to IT professionals with existing SQL skills. In both of these examples, the result is an answer that can be used to make a decision.

By combining Big Data with Business Rules, data scientists and subject-matter experts can now write their business logic for big data analytics using natural language rules and decision tables. This makes the logic much easier to understand and the separation from the source language of the big data platform allows for more rapid change of the processing logic.

This pattern is detailed in the following IBM developerWorks articles:

- *Integrating Hadoop and IBM Operational Decision Manager*

This article provides a generic pattern for using business rules within the context of a Hadoop MapReduce framework. This is illustrated by using a trade validation example.

http://www.ibm.com/developerworks/bpm/bpmjournal/1209_crowther/1209_crowther.html

- *Think big! Scale your business rules solutions up to the world of big data*

This article applies the integration pattern that is described in the first article between Hadoop and ODM using IBM Bluemix™. IBM Bluemix is an open-standard, cloud-based platform for building, managing, and running applications of all types, such as web, mobile, big data, and smart devices. This article can be found at:

http://www.ibm.com/developerworks/bpm/library/techarticles/1411_crowther-bluemix/1411_crowther.html

For information about various IBM big data solutions, see IBM big data website at:

<http://www.ibm.com/software/products/en/category/bigdata>



Analyze and report

Analyze is one of the most important steps in the overall process of decision making. Before this phase, relevant historical data must be integrated, gathered, cleansed, aggregated, transformed, and formatted to a form that can be used by the analytics solution.

Models are developed and selected with appropriate algorithms and statistical techniques for mining the historical data. The historical data is used to “train” the models. The incoming data can also be “consumed” if an immediate action is required. If it is to be analyzed later and in aggregate, then the data is “collected” and stored for analysis later. The accuracy of the results of analysis is further improved through an iterative process to enhance or ‘train’ the models. The models are then applied to current data to produce actionable insight. Business rules and policies are applied to this insight to further enhance the usefulness of the results of the model execution. The resulting actionable insight is the basis of decision making.

The results of the analyze phase, the actionable insight, include “descriptive analytics” as in reports and dashboards or “predictive analytics” such as forecasts or estimated outcome of a decision. Further analysis, learning from experience and optimization can yield “prescriptive analytics” that reveal the best possible decision. “Cognitive analytics” is the next stage of maturity in the decision-making process with the exploitation of multiple technologies to come close to human analysis.

This chapter describes the *analyze* and *report* phases from the system of insights model defined in Chapter 2 with many of the solutions in the IBM Analytics portfolio as shown in Figure 5-1.

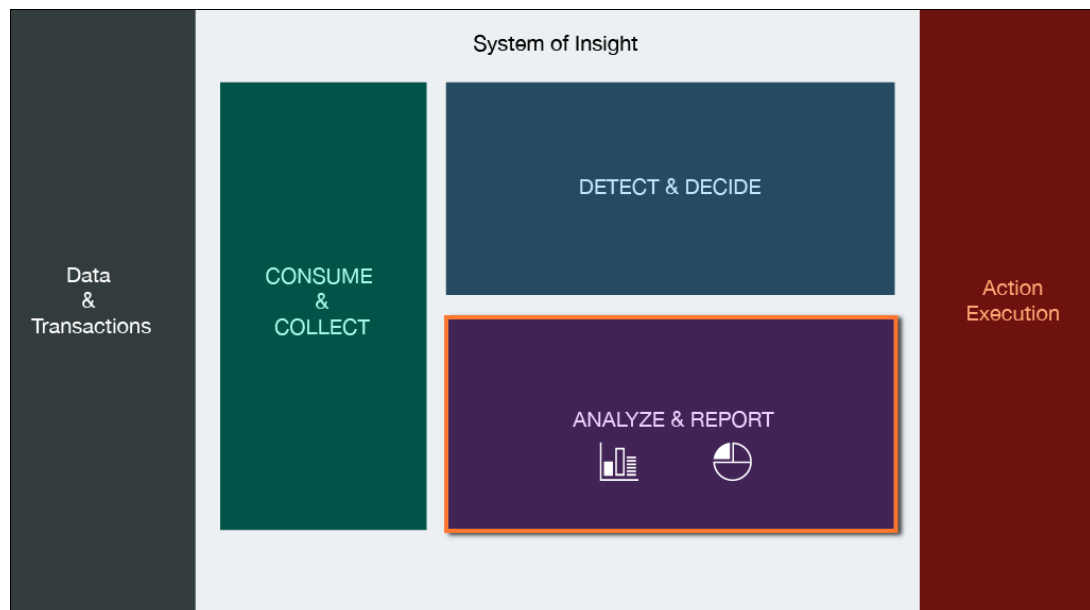


Figure 5-1 System of insight: Analyze and report

In particular, this chapter describes predictive analytics with IBM SPSS Modeler and how it can be used with IBM Operational Decision Manager Advanced (ODM Advanced).

This chapter covers the following topics:

- ▶ Descriptive analytics
- ▶ Predictive analytics
- ▶ Real-time analytics
- ▶ Prescriptive analytics
- ▶ Cognitive analytics
- ▶ Reporting and monitoring

5.1 Descriptive analytics

The results of the analyze phase can include descriptive results such as in reports and dashboards. These results use visualization and graphical techniques to format and display past information. Reports describe *what happened* and dashboards depict the more current *what is happening* alongside relevant past data. The information conveyed by the reports and dashboards is used by decision makers to take the next steps, usually after the fact (retroactive decisions). Businesses have been using reports for years. Reports provide decision makers with good information to make better decisions compared to the ones based on human instincts.

5.2 Predictive analytics

The more powerful and useful results of the analyze phase are the predictions such as a forecast or an indication of what is likely to happen. These predictions are a result of data mining, machine learning, and the application of sophisticated statistical models to the current situation. Through this process, the outcome of a decision is predicted with a high degree of confidence. Such high confidence predictions allow the decision makers to take proactive actions that can help prevent adverse results before they happen. This is a much better situation than trying to fix problem situations after the fact. The ability to reduce or prevent adverse outcomes, and identify and take advantage of positive outcomes make predictive analytics vital for the future of businesses.

5.2.1 The predictive analytics process: The CRISP-DM data mining process

As data analytics capabilities become more prevalent, data scientists need a methodology capable of providing a guiding strategy, regardless of the technologies, data volumes, or approaches involved. This is the Cross-Industry Standard Process for Data Mining (CRISP-DM) methodology model (Figure 5-2).

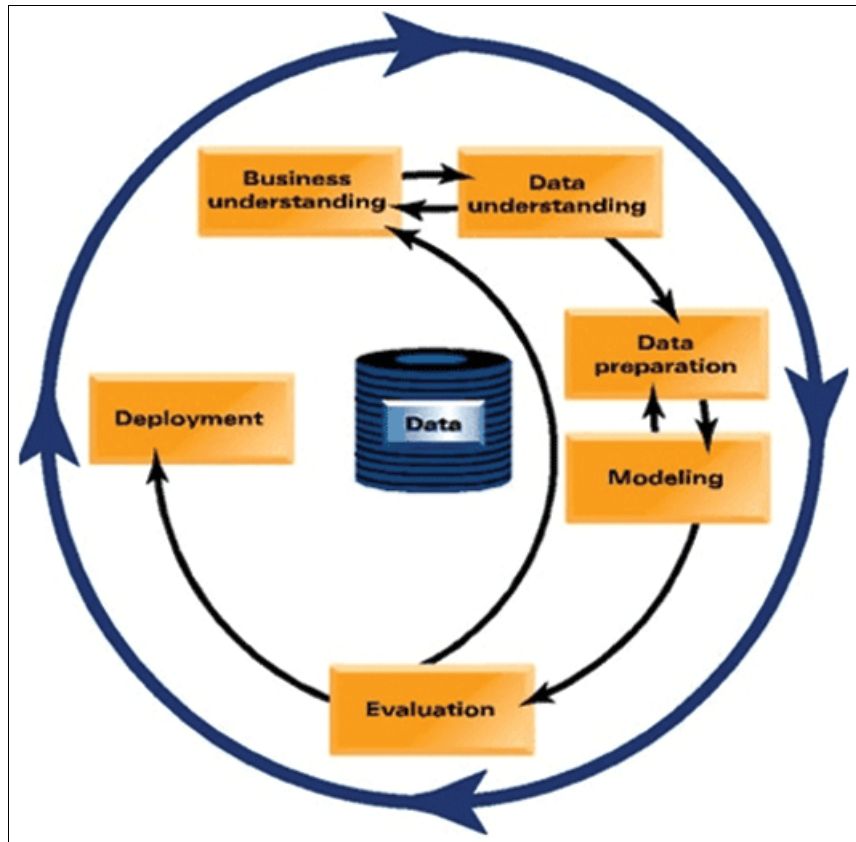


Figure 5-2 Cross Industry Standard Process for Data Mining (CRISP-DM) methodology model

The general CRISP-DM process model includes six phases that address different areas of the data mining process. It is a cyclical process that is designed to incorporate data mining as part of the business processes.

The CRISP-DM process includes the following phases:

- ▶ *Business understanding*: A good understanding of business that includes business objectives, assessment of situation, problems to be solved, the goals of data mining, and so on are essential for data mining.
- ▶ *Data understanding*: Data includes hidden patterns and relationships that need to be discovered through the process of data mining. A good understanding of the available data, data sources, and their characteristics is needed to discover the relationships. Collection of initial data, data description and exploration, and verification of data quality are part of the understanding phase as well.
- ▶ *Data preparation*: Data must be prepared and formatted for mining. The preparation includes selection, cleansing, construction, integration, and formatting.
- ▶ *Modeling*: In the modeling phase, sophisticated analytical techniques and methods are used to extract valuable insight from data. This phase includes selection of the correct modeling techniques, generating trial designs, and building and updating models.

- *Evaluation:* The data mining results are evaluated for their effectiveness in achieving the business objectives. The main steps include review of results, evaluating their effectiveness, reviewing the model and the data mining process, and determining the next steps.
- *Deployment:* In the deployment phase, the focus is on integrating the knowledge and models into the business processes to solve the original business problem. Deployment includes deployment plan, integration, monitoring and maintenance, and project review.

Although the normal process is flow through these steps, feedback loops influence the previous steps. Data is usually prepared before modeling. However, review of the modeling phase and the results might suggest changes in the data preparation. Modeling and data preparation might require a few feedback cycles to obtain the correct results.

Evaluation is another phase that can require reevaluation of the understanding of business and the target answer. The feedback might require a redefinition of the problem being solved.

The overall data mining process is highly iterative. As more knowledge is gained from one cycle of data mining, it usually leads to new questions, new issues, and new opportunities. These can be investigated by mining the data again. This iterative process leads to a continuously improves data mining, enhances the accuracy of the modeling process, and consequently improves the quality of decisions.

With the current computing capabilities and availability of sophisticated SPSS Modeler and statistics, and the rules engine, ODM Advanced with Decision Server Insight (DSI), the decision-making process can be automated and applied to every transaction in real time, when necessary.

The premier platform from IBM for predictive analytics is the powerful and sophisticated IBM SPSS suite, which has extensive modeling and statistical capabilities.

5.2.2 Predictive analytics with SPSS

IBM SPSS suite of solutions includes Data Collection, Modeler, Statistics, and Social Media Analytics. Data Collection facilitates creation of effective surveys and integration of feedback into the decision-making process. SPSS Statistics applies sophisticated statistical techniques and analysis to understand data, spot trends, predict outcomes, and develop forecasts with a high degree of confidence. The Social Media Analytics can analyze massive volumes of unstructured data from social media sources as part of “big data” analytics.

SPSS Modeler is a set of powerful data mining and modeling tools. These tools facilitate quick development of predictive models and deploying them into business processes for effective decision making. The Modeler adheres to the industry standard (CRISP-DM).

When it comes to data mining, SPSS Modeler facilitates the discovery of useful patterns and relationships within large data sets. Compared to older statistical systems tools, there is no need for preconceived ideas of these relationships. The Modeler helps explore the data by fitting various available models to investigate different relationships until the most useful information is found.

The modeler has a graphical interface and allows discovery of patterns, trends, and relationships in both structured and unstructured data. It includes advanced algorithms and is supported by analytic capabilities that include text and entity analytics for optimization of decision management process. SPSS Modeler offers several different methods that are derived from the fields of statistics, machine learning, and artificial intelligence. The modeler allows the selection of a method that is best suited for the specific problem being solved.

SPSS Modeler allows deployment of predictive models directly into the business processes and helps optimize decisions every time the business process is run. These predictive models can be integrated with rules, scoring, and optimization solutions. For example, the modeler integrates with IBM ODM to apply predictive analytics to automated operational decisions and situation detection systems. Examples of using SPSS with IBM ODM are provided in 5.3, “Real-time analytics” on page 62 and in Chapter 8. IBM Decision Server Rules (DSR) and IBM DSI are components of ODM Advanced that facilitate detection of situation, events, and apply business rules to decision making.

5.2.3 SPSS Modeler capabilities

SPSS Modeler is an easy to use, versatile, and sophisticated predictive analytics solution from IBM. It is available in Professional and Premium versions with varying capabilities.

SPSS Modeler Professional

SPSS Modeler Professional provides all the tools that you need to work with most types of structured data, such as behaviors and interactions tracked in CRM systems, demographics, purchasing behavior, and sales data.

SPSS Modeler Premium

SPSS Modeler Premium is a separately licensed product that extends SPSS Modeler Professional to work with specialized data such as that used for entity analytics or social networking, and with unstructured text data. SPSS Modeler Premium comprises the following components.

IBM SPSS scoring capability

The IBM SPSS Collaboration and Deployment Services Deployment Portal allows you to generate scores from models deployed in IBM SPSS Collaboration and Deployment Services Repository. The SPSS Modeler streams, scenario files, and Predictive Model Markup Language (PMML) files can be scored.

This component facilitates scoring of individual, incoming transactions in real time. The scoring is performed directly within the online transaction processing (OLTP) application itself. This real-time capability and the ability to incorporate the scoring into the decision-making process makes this a powerful solution for real-time analytic applications such as fraud detection. With this solution, the fraudulent activity can be prevented before the damage occurs.

By avoiding export and reformatting data for processing through a model on another server and importing it back into the original OLTP system, the performance is dramatically improved and need for resources and therefore costs are substantially reduced.

IBM SPSS Modeler Entity Analytics

This component provides further enhanced capability to the IBM Modeler. When data to be analyzed contains natural variability, errors, and possible deliberate falsehoods entity analytics allows resolution of these inconsistencies. This capability focuses on improving the coherence and consistency of current data by resolving identity conflicts within the records themselves. The modeler entity analytics improves the quality of data even when the entities do not share any key values. An identity can be that of an individual, an organization, an object, or any other relevant entity for which ambiguity might exist. Identity resolution can be vital in a number of fields, including customer relationship management, fraud detection, anti-money laundering, and national and international security.

IBM SPSS Modeler Social Network Analysis

This component analyzes social media data, text files, and entities, and assesses their impact on the social behaviors of individuals and groups. Interaction between individuals on social media is analyzed to provide insight into the behavior of the individuals. Using these relationships, the Modeler Social Network Analysis can identify social leaders who influence the behavior of others in the network. It can also determine who are most influenced by other network participants. By combining these results with other measures, comprehensive profiles of individuals can be created. The predictive models can be based on these profiles, which can be a lot more effective than the ones do not use social network analysis. This analysis is critical in situations such customer retention and churn. Predictive models can be built to assess individuals in the network at increased risk of churning and preemptive actions can then be planned to retain those customers.

IBM SPSS Modeler Text Analytics

This component uses advanced linguistic technologies and natural language processing (NLP) to rapidly process a large variety of unstructured text data, extract and organize the key concepts, and group these concepts into categories. Extracted concepts and categories can be combined with existing structured data, such as demographics, and applied to modeling using the full suite of Modeler data mining tools to yield better and more focused decisions. Unstructured data analyzed by the Modeler can include items such as text messages, emails, social media data, content from blogs, and customer feedback to develop customer sentiments, trends, and themes can be captured and used make better decisions.

Geospatial analytics

This component takes in to account geographic location, time, and space in predictive models. This analysis spots pattern and trends in geographic, location-based context so that response can be prepared for the local conditions. The predictive model can predict and forecast events at a specific location at some time in the future. Disease surveillance and crime prevention are some of the areas where geospatial analytics is commonly used.

Automated Model Selection

This component allows use of more than one modeling methods simultaneously in an individual run and depending on the results allows selection of the best model. These models do not have to be run separately for comparison. Auto Classifier, Auto Numeric, and Auto Cluster are the three choices in automated model selection.

5.2.4 The predictive modeling process with SPSS

The predictive modeling process begins with a good understanding of the problem that needs to be solved, the underlying business process, and the various relevant variables that are associated with the business process. A good understanding of the problem and business process is used to identify the target predictors of the outcome.

Historical data is mined and analyzed using various techniques to discover patterns and relationships. The output from data mining can be used in areas such as decision support, prediction, forecasts, and estimation. The data mining process generates models from historical data that can be used with current data for prediction of outcomes.

SPSS Modeler includes many facilities and techniques that can be used to apply expertise to data, such as:

- ▶ Data manipulation. Constructs new data items derived from existing ones and breaks down the data into meaningful subsets. Data from various sources can be merged and filtered.

- Browsing and visualization. Displays aspects of the data using the Data Audit node to perform an initial audit including graphs and statistics. Advanced visualization includes interactive graphics, which can be exported for inclusion in project reports.
- Statistics. Confirms suspected relationships between variables in the data. Statistics from IBM SPSS can also be used within the Modeler.
- Hypothesis testing. Constructs models of how the data behaves and verifies these models.

Usually, these facilities and techniques are used to identify potentially useful attributes within the data. These attributes are then used with modeling techniques to identify underlying relationships and rules.

The results from data mining are then used to build predictive models using the best suited methods from the various methods that are offered in the Modeler. These models are then applied to current data to arrive at a prediction with a high degree of confidence. Predictive output from an executed predictive model can be exported to different devices such as the BI solution, other visualization solutions, mobile devices or an automated device for action, and so on.

The predictive modeling begins with the “stream canvas” space on Modeler window where models are built and data streams are manipulated (Figure 5-3). The streams run from left to right on the canvas. Streams are created by drawing a workflow of operations that are required for the model on the canvas. The operations are represented by icons called nodes that are linked in a logical order.

Below the stream canvas in the Modeler window is a “palette”, which is a series of tabs that provide the data and nodes that perform specific operations in the stream. These nodes are dragged onto the canvas and connected in a logical flow to create the streams. Double-clicking the individual nodes in the canvas opens pop-up input boxes that allow you to enter information such as pointing towards files and tables.

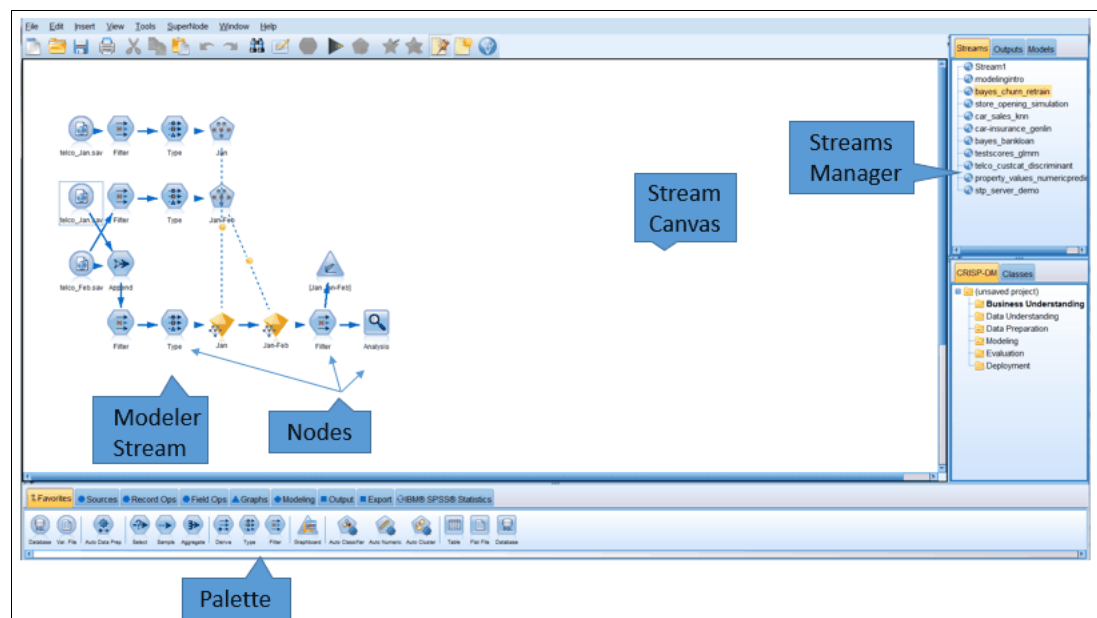


Figure 5-3 SPSS Modeler stream canvas

Each palette tab contains a collection of related nodes used for different phases of stream operations:

- ▶ **Sources:** Nodes bring data into the Modeler. Includes multiple data sources.
- ▶ **Record Ops:** Nodes perform operations on data records, such as selecting, merging, and appending.
- ▶ **Field Ops:** Nodes perform operations on data fields, such as filtering, deriving new fields, and determining the measurement level for given fields.
- ▶ **Graphs:** Nodes graphically display data before and after modeling. Graphs include plots, histograms, web nodes, and evaluation charts.
- ▶ **Modeling:** Nodes use the modeling algorithms available in the Modeler, such as neural nets, decision trees, clustering algorithms data sequencing, integration with R, and the open source statistical and graphing environment.
- ▶ **Output:** Nodes produce a variety of output of data and charts. Model results can be viewed in the Modeler. The Data Audit tool in the Output node palette of the Modeler is a good tool for understanding data.
- ▶ **Export:** Nodes produce a variety of outputs that can be viewed in external applications such as the IBM SPSS Data Collection and Microsoft Excel.
- ▶ **IBM SPSS Statistics:** Nodes import data from, or export data to, IBM SPSS Statistics, in addition to running IBM SPSS Statistics procedures.

Multiple streams can be created and worked on at the same time and a “streams manager” in the upper right portion of the modeler window. The palette can be customized according to your preferences.

Below the Nodes palette is the report pane that shows the progress of the operations in a stream. A status pane indicates what the application is doing currently and whether user feedback is required.

To build a model based on the data requirements, drag data sources nodes from the first tab onto the stream canvas. The Modeler can simultaneously import data from many diverse sources. These sources can include databases, spreadsheets, flat files, CSV files, and Point of Sale terminal data. After the data source is selected, these source nodes can be pointed to a specific data source such as a file. The modeler then allows a preview of the data source by displaying a subset of up to 10 rows of data from the source. This preview helps ensure that the data in the source is in the correct format for use by the model. The data management and exploration capabilities of the Modeler allow actions such as creation of a new variable from the input data as part of data preparation. Operations nodes from Record Ops and Field Ops can be added as needed.

The Graph node, as the name implies, allows data exploration and visualization by displaying different types of graphs.

The Modeling tab offers nodes for different modeling and machine learning techniques. Currently, there are nearly 35 distinct techniques available. The machine-learning and modeling technologies that are offered by the Modeler can be roughly grouped according to the types of problems they are intended to solve:

- ▶ **Predictive modeling methods** include decision trees, neural networks, and statistical models.
- ▶ **Clustering models** focus on identifying groups of similar records and labeling those records accordingly. Clustering methods include Kohonen, k-means, and Two Step.

- ▶ Association rules associate a particular conclusion (such as the purchase of a particular product) with a set of conditions (the purchase of several other products).
- ▶ Screening models can be used to screen data to locate fields and records that are most likely to be of interest in modeling and identify outliers that might not fit known patterns. Available methods include feature selection and anomaly detection.

The output tab contains various files such as graphs and tables. These nodes are used to display and visualize the results from the execution of the model.

The export tab facilitates export of results from the models to the decision-making solutions such as Business Intelligence solutions, statistical applications, and spreadsheets.

SPSS Modeler and Decision Server Rules (DSR)

Analytics can provide insights during the decision management process by using the knowledge that is gathered in a manual data analysis. Figure 5-4 shows a system of insight architecture with two components, SPSS Modeler and Decision Server Rules.

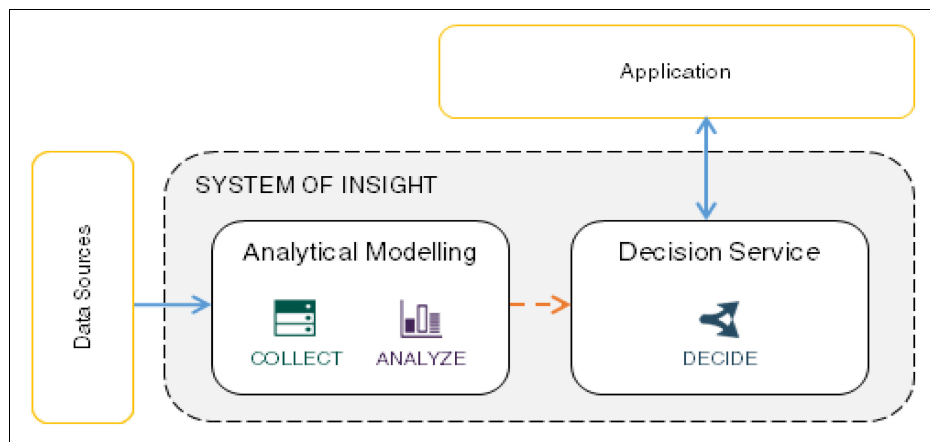


Figure 5-4 System of Insights components with SPSS and DSR

The building process of this system of insight takes place in two main steps that can be run multiple times for further refinement.

1. Data scientist manually analyzes collected data through a complex model to identify unknown data patterns with business value.

In this step, data scientists identify the most important attributes and the impact of these attributes in a pattern. That is, they identify segmentation of data based on the identified attributes and identify the behavior of data based on the segmentation identified.

The example scenario uses SPSS Modeler to create a model to perform the analysis.

2. Operationalizing the decisions using attributes and behaviors that are defined in the analytics phase.

In this step, subject matter experts create business rules using the knowledge gathered by data scientists in step 1. DSR provides different artifacts to implement decision logic, which can be in the format of if-then statements, decision tables, and decision trees.

The following examples show where data from analysis can be used for decisions:

- ▶ Retail: Online application for personalized catalog of products using factors such as gender and age derived from some analytical models as key factors in the customer buying behavior.
- ▶ Insurance: Probability of a claim to be a fraud during the claim processing phase.

- Entertainment: Application to provide a list of recommended movies or shows to viewer based on their calculated likelihood probability for their household income or day of the week.

DSR provides agility through business rules that can be changed during business operations. Subject matter experts can make their business decisions based on both the facts gathered from data and their experiences.

DSR can also be integrated with Analytical Decision Management products. For more information, see the *Patterns for Combining Analytics with Operational Decision Management in Smarter Process* technote at the following link:

<http://www.ibm.com/support/docview.wss?uid=swg21683093>

SPSS Scoring and Decision Server Insights (DSI)

DSI uses action rules to make decisions based on data received from events at the moment of interaction to apply actions. The decision and resulting action can use analytics, in particular, scoring services that are provided by SPSS Collaboration and Deployment Services.

Figure 5-5 shows a system of insight that includes the three components that are needed for real-time analytics.

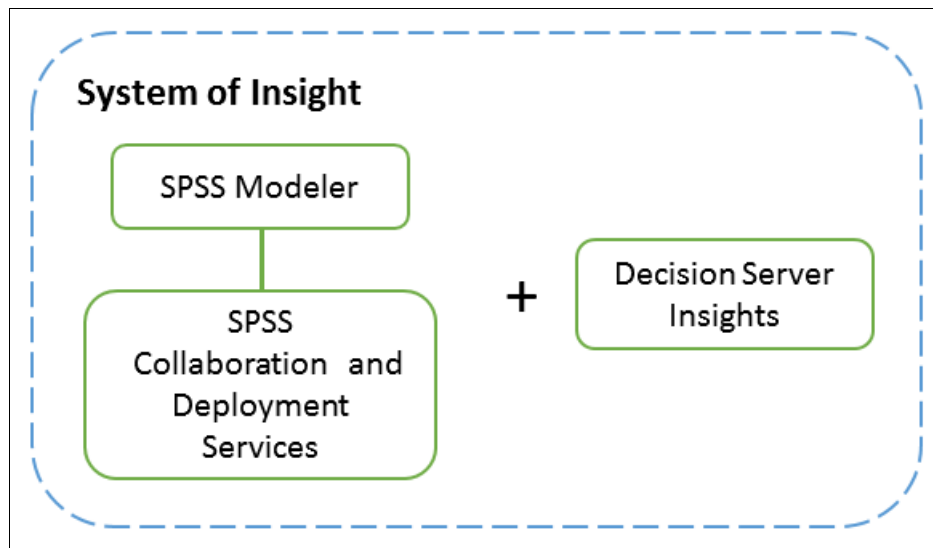


Figure 5-5 System of Insights components with SPSS and DSI

The analytics process is performed in three steps:

1. A data scientist uses SPSS Modeler to manually create a model based on collected data to identify behavior patterns.
2. The application developer deploys the model to the SPSS Collaboration and Deployment Services server where the scoring service is created and externalized.
3. DSI starts the scoring service in real time to retrieve the values that are needed for a decision.

The following examples provide predictive analytics scenarios:

- Hospitality: Room management is one of the major tasks for hotels. A hotel company must be able to predict how many rooms will be available on a specific date depending on factors such as events being hosted at their location, season of the year, and so on. DSI

can be used in an alert system to prevent overbooking and to react to an overbooked case in a way that prevents a bad customer experience.

- ▶ Transportation: A shared bicycle program application where DSI is used to recommend the correct station to drop a bicycle by using the probability of having empty spaces available provided by SPSS and other factors like distance to different stations.

From the implementation perspective, the suggested patterns for integrating DSI with SPSS is to use the following agent priorities:

- ▶ Predictive scoring agent has the priority set to high in the agent descriptor
- ▶ Predictive scoring agent updates attributes from the bound entity with results from the scoring service
- ▶ A rule agent has the priority set to lower value than the predictive scoring agent, which ensures that the entity was updated when the rule agent processes it.

For information about how to set priority in an agent descriptor, see the IBM Knowledge Center at:

http://www.ibm.com/support/knowledgecenter/SSQP76_8.7.1/com.ibm.odm.itoa.ref/topics/ref_itoa_bdl_agents.html

5.3 Real-time analytics

Real-time analytics is a powerful and critical capability. The most significant benefit of real-time analytics is the ability to reduce or prevent the adverse result of an action before the action is run. Common examples are real-time fraud detection in case of an ATM withdrawal or a credit card transaction. Although the business wants to process legitimate transactions quickly, they also want to prevent fraudulent transactions from going through immediately. To make that happen, the solution must be able to identify potentially fraudulent transactions instantly, without consuming too much processing power or latency. Depending on the size, transaction volume, and geographical spread of the business, the real-time solution might have to handle a substantial workload.

The IBM solution for real-time analytics is the SPSS scoring capability, which facilitates analysis of every individual incoming transaction within the business process itself and assigns each transaction a “score” based on specific factors that are indicators of the quality of the transaction.

The IBM SPSS Collaboration and Deployment Services Scoring Server is a separate application under the IBM SPSS Collaboration and Deployment Services Scoring Service. The predictive model that is used for scoring can be defined by using IBM SPSS Modeler streams, scenarios, or PMML generated from other IBM applications. In addition, existing markup (for example, SPSS-ML) from older products can also be used for scoring. For more information, see the IBM SPSS Modeler documentation at:

<http://www.ibm.com/support/knowledgecenter/SS69YH/welcome?lang=en>

The transaction score that is generated by the scoring service can then be used to decide how the transaction should be treated. A score at a certain value has a high probability of being fraudulent. The score instantly flags the suspicious transaction and can send it to a “case management” system for further processing. All of this can happen in a short time so that it does not affect the execution times of legitimate transactions.

The scoring can also be performed by running an SPSS Modeler directly within the business process. The “nugget” from the SPSS Modeler is published to the SPSS Collaboration and

Deployment Services Deployment Manager. The publishing process generates an SQL statement. This SQL statement uses user-defined functions (UDFs) to start the SPSS Modeler stream and generates a predictive score that can be used by any decision management application.

SPSS facilitates predictive analytics and transaction scoring in combination with application of business rules, which together make the “next best action” recommendation possible. This capability is essential for implementing real-time analytics. After the score for a transaction is calculated, business rules can be applied to determine the best disposition of that transaction. Application of business rules can also be automatically applied by IBM ODM Advanced including DSR and DSI. The following implementations are described in Chapter 6:

- ▶ Using predictive models with DSR
- ▶ Using predictive models with DSI
- ▶ Using predictive models with InfoSphere Stream

5.4 Prescriptive analytics

Prescriptive analytics is a set of techniques and technology that can guide the business in not only identifying risks and opportunities, but also selecting among them.

They break down into two broad categories:

- ▶ Combining predictive analytic models and business rules. The predictive models calculate the probability of a set of outcomes and business rules can then be used to choose a next best action that increases the likelihood of a favorable outcome.
- ▶ Using decision optimization and mathematical modeling to compute an optimal next best action based on a mathematical model of the business domain, business constraints, and a cost/reward function to be optimized. This approach does not require predictive capabilities. However, a model of decision variables, constraints, costs, and rewards must be expressed mathematically such that a solver such as IBM CPLEX can be used to compute the optimal next best action. This approach can lead to an extreme return on investment.

5.5 Cognitive analytics

As the name implies, cognitive analytics comes close to the analytics systems “thinking” like humans. This technology uses natural language processing, pattern recognition, statistical techniques, and stochastic optimization. With cognitive analytics, machines learn, adapt, process, and analyze vast amounts of data and interact with people. These systems help improve decisions by managing the complexities of analyzing big data. IBM Watson Analytics is a powerful cognitive analytics solution.

5.6 Reporting and monitoring

Monitoring of activities within an enterprise is collectively known as business activity monitoring (BAM). Generally these activities involve customers, suppliers and business partners, and various departments in the enterprise. The goal of BAM is to provide the decision-makers with information about and status of the various business processes, activities, operations, and transactions in real time. This helps them to make informed

decisions rapidly, and resolve issues before any possible damage occurs. This also allows the enterprise to use any new opportunities and technologies.

Cognos Business Intelligence is the IBM premier solution for monitoring and reporting. It transforms transaction and operational data into historical, current, and predictive views of business, which provides a complete picture of past, current, and possible future insight. These views can be in the form of reports, dashboards, and scorecards that provide valuable insight for effective decision making. Cognos is implemented and integrated with SPSS suite of products including the Modeler.

5.6.1 Reports

The correct reports can help in making more effective decisions. Cognos Business Intelligence provides authoring, modifying, viewing, and interactive visualizing capabilities. The reports are accessible online or offline, on mobile devices, and embedded within Microsoft Office applications or other in-process applications. The reports can be created by professional authors and distributed to the whole enterprise, or the business users can create their own or modify existing reports. Mobile workers can interact with reports while disconnected.

Cognos provides guided analysis of analytics reports. The source of the lineage of specific information can be viewed to make sure that it is from the appropriate source. It is also possible to move from a high-level overview to a more detailed, specific view within the reports. The drill-down capability facilitates investigations such as root cause analysis. Cognos provides a large and growing number of customizable visualizations and reports available for download.

5.6.2 Dashboards

Dashboards are effective tools for monitoring business activity in real time. The input to dashboards is usually from business processes, business activities, business events, and other information sources. Dashboards depict the business process performance graphically and give users visibility into key measurements of these business processes. Solutions that provide dashboards must be flexible and customizable to address the specific needs of every business. Many dashboards have drill-down capability to review individual measurements further. Dashboards facilitate prompt and correct decisions for improving the business processes.

IBM Cognos Business Intelligence is a solution that provides effective business monitoring through dashboards. Dashboards in Cognos Business Intelligence allow viewing and monitoring, deep diving into specific fields, and interacting with historical data side by side with current data including any data in motion, and possible predictive outcomes. These dashboards can be customized for the unique decision-making requirements of the business. With Cognos dashboards, content can be displayed in a personalized way and shared with others.

Figure 5-6 shows a sample Cognos dashboard image.

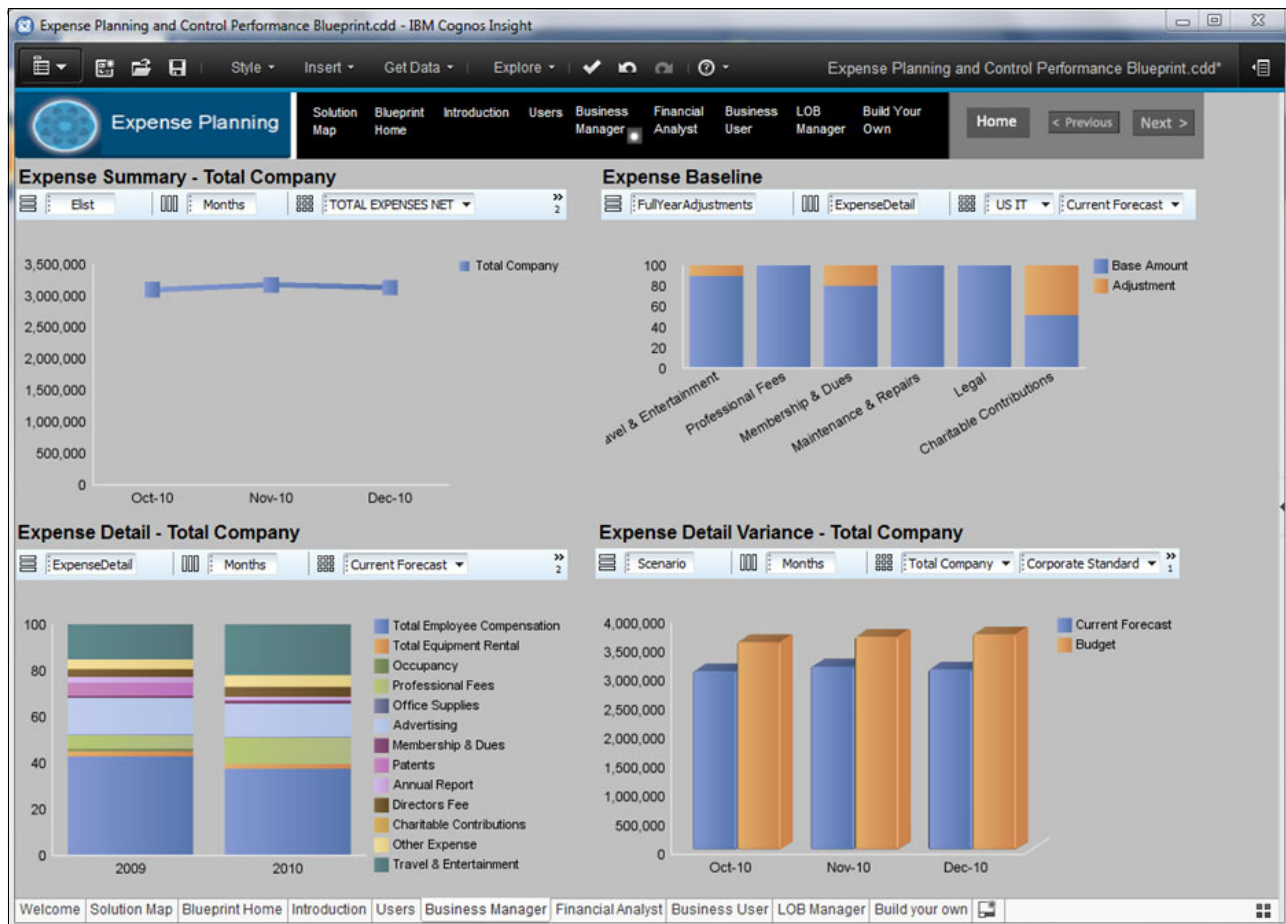


Figure 5-6 Sample Cognos dashboard image

Another IBM solution that has powerful tools for monitoring business activity is the IBM Business Monitor. It gives users a real-time view of business processes including performance and reports on business operations. Dashboards in IBM Business Monitor can display key performance indicators (KPIs) and critical metrics and process alerts. The dashboards can be displayed in a browser on desktops and mobile devices such as smartphones and tablets. The dashboards that are provided by IBM Business Monitor are highly customizable. They can provide notifications and alerts to provide a near real-time view of the business operations, transactions, and processes. IBM Business Monitor can also indicate trends, patterns, and issues by performing statistical analysis on historical and current data.



Detect and decide

This chapter describes the core concepts around the detect and decide capabilities in the context of a system of insight.

It describes in detail the various tools available within IBM Operational Decision Manager Decision Server Rules (ODM DSR) and Decision Server Insights (DSI) for making an automated business decision.

This chapter covers the following topics:

- ▶ Introduction
- ▶ Request-driven decisions in ODM Decision Server Rules
- ▶ Situation-driven decisions in ODM Decision Server Insights

6.1 Introduction

In a system of insight, detect and decide are two capabilities that can add significant business value and competitive differentiation. Figure 6-1 highlights these capabilities in the example model for system of insight categorization defined in Chapter 3.

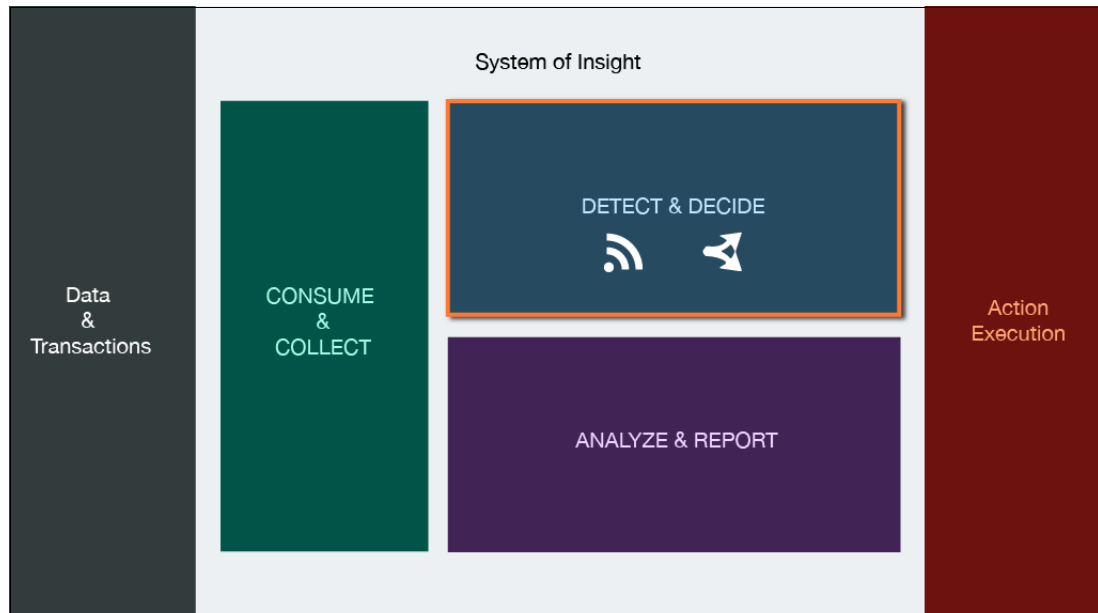


Figure 6-1 System of insight: Detect and decide

6.1.1 Detect

Detect, in the context of systems of insight, refers to the detection of situations. These situations often correspond to the circumstances when you want to take action.

After you determine which situations you are interested in, the question becomes how to detect them. This generally involves the identification of patterns.

If the pattern is simple, and the context changes infrequently and the response can be delayed, then a person monitoring a dashboard might offer this capability.

However, if the pattern is complex, or the context changes frequently, it is often much more useful to have this capability automated. Several products offer this capability. Where they differ is in the types and complexity of the patterns that they can identify. This chapter focuses on IBM Operational Decision Manager Advanced.

6.1.2 Decide

Decide is fairly self-explanatory, but there are some nuances to note. There is a distinction between request-driven (6.2, “Request-driven decisions in ODM Decision Server Rules” on page 73) and situation-driven (6.3, “Situation-driven decisions in ODM Decision Server Insights” on page 88) decisions. It is important to consider how and when your decision is made because this information informs whether your decision is request-driven or situation-driven.

For example, if you need to decide to offer a loan to a loan applicant, use a request-driven decision because the decision only depends on the customer information and loan details. However, if you want to offer car insurance for a customer, conditions of the insurance might be driven by the buying history of the customer, number of vehicles the customer or his family own, and physical area where the customer will be driving. In this case, you need the situation-driven decision system.

6.1.3 Interaction between detect and decide

Any system of insight should be able to decide whether this is an automated process (for example, with a business rules management system), or a manual process (for example, with a subject matter expert making a decision based on information provided by a dashboard). It is worth noting that this decision can be “do nothing”.

Therefore, in this chapter, rather than discussing detect and decide individually, instead they are grouped in the following way:

- ▶ Decision without detection, or request-driven decisions (6.2, “Request-driven decisions in ODM Decision Server Rules” on page 73).
- ▶ Decision with detection, or situation-driven decisions (6.3, “Situation-driven decisions in ODM Decision Server Insights” on page 88).

6.1.4 Decision management

Decisions can be found everywhere in all organizations and are one of the most important assets of a company. Some decisions are documented and visible to every individual in the organization. However, more often, decisions are not visible. This lack of visibility makes these decisions difficult to track and keep up to date with current policy.

There are various different decision sources. When made manually, they are found in requirement documents, operation manuals, or people’s heads. When decisions are implemented, and perhaps automated, they can be found in almost every software application, with various programming languages such as COBOL, Java, and C#. The scattered decision sources create these challenges for organizations:

- ▶ Consistency: When a decision is in different applications, any change to that decision must be replicated in all applications that contain the logic. This assumes that you even know where all these decisions are. Even when all the applications with the decision are identified, the changes might not be able to be implemented simultaneously for all these applications due to various reasons such as schedule and resource availability. This creates inconsistency in the organization.
- ▶ Agility: When decisions are in multiple applications, a policy change requires more time to coordinate the implementation of the changes with application owners who can be spread across departments or even across divisions.
- ▶ Lack of efficiency: When decisions are everywhere, more testing and rework time is required to ensure that new changes are applied consistently in all the applications. This inefficiency increases the cost to the company.
- ▶ Lack of visibility: To meet regulatory compliance requirements, organizations must provide reports with information about how their business operates and how decisions were made. When decisions are embedded in multiple applications, demonstrating compliance becomes a challenging task.

To address these challenges, a decision management system that allows users to achieve the following goals is required:

- ▶ Author decisions in a more business-oriented language.
- ▶ Store decisions in a place where these decisions are visible and easily accessed for maintenance.
- ▶ Run and share decision applications with other applications within their organizations.

A decision management system with these capabilities is considered a business rules management system (BRMS). A BRMS is an essential technology within operational decision management. It allows organizational policies, and the repeatable decisions associated with those policies such as claim approvals, cross-sell offer selection, pricing calculations, and eligibility determinations, to be defined, deployed, monitored, and maintained separately from application code. With a BRMS, organizations can be more agile and effective in their decision management processes.

A BRMS manages decisions represented by collections of rules. These rules provide solutions to complex decisions that can be stateless or stateful. When decisions are stateless, the ability of the system to provide a result depends on two things: Data provided to the system and defined business rules. These business rules are evaluated against provided data and when conditions inside the rules are met, actions are taken. Stateless decisions do not keep any history on previous results, so every time that the system is started, all the business rules are evaluated against provided data. These types of decisions are considered *request-driven decisions* because there is no history of previous invocations. Request-driven decisions are always made in a passive mode, which means you always need a client application to trigger a request for a decision.

When time and history are important in decision making, use *situation-driven decisions* where interest is based on rule logic related to a set of situations identified by monitoring streams of events. The notion of events becomes important to organizations because events can trigger actions that can prevent unwelcome situations from happening. Fraud detection is a good example of situation-driven decision making. A credit company wants to detect atypical spending behavior of their customers at the precise moment that the behavior occurs. It is important to detect unusual behavior quickly so you can decide what preventive action is required.

IBM ODM is the IBM offering to support complex decisions for both request-driven and situation-driven decisions. ODM provides a set of components to fully support needs for authoring, storing, and running decisions. The following main components are illustrated in Figure 6-2:

- ▶ **IBM Decision Center (DC):** This component focuses on the rule management process and provides all tools that are needed for authoring and governing business rules. This component provides two main web-based consoles: One for users to author, validate, and maintain rules, and another oriented towards administrative tasks.
- ▶ **IBM Decision Server:** This component focuses on the execution, design, and monitoring of all executable artifacts. Decision Server comprises two submodules: DSR and DSI. The type of decision application defines what submodule is needed. DSR is responsible for request-driven decision applications. DSI is responsible for situation-driven decision applications.

The following sections describe the differences between these two submodules.

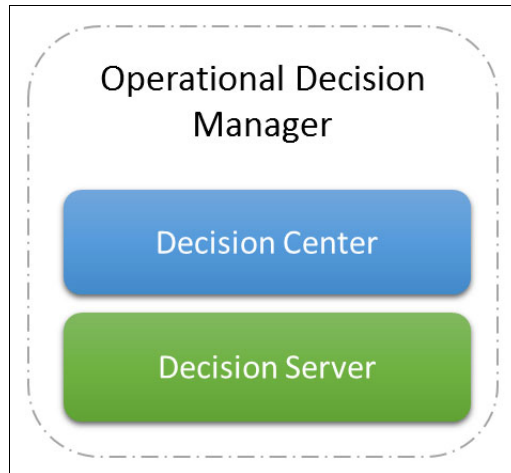


Figure 6-2 High-level view of ODM components

6.1.5 ODM in the context of systems of insight

Table 6-1 shows the three ODM offering packages.

Table 6-1 ODM packages

Offering	Description
Express	Basic offering that includes all business rules management capabilities. ODM Express is targeted for small, entry-level projects.
Standard	Includes all the business rules management capabilities. ODM Standard is targeted for customers with requirements associated to business rules only.
Advanced	Full offering of decision capabilities related to business rules management systems and situation detection.

The examples in this book use ODM Advanced to demonstrate its capabilities in a systems of insight application.

In a system of insight application, ODM provides capabilities in consume, detect, decide, and analyze, as shown in Figure 6-3.

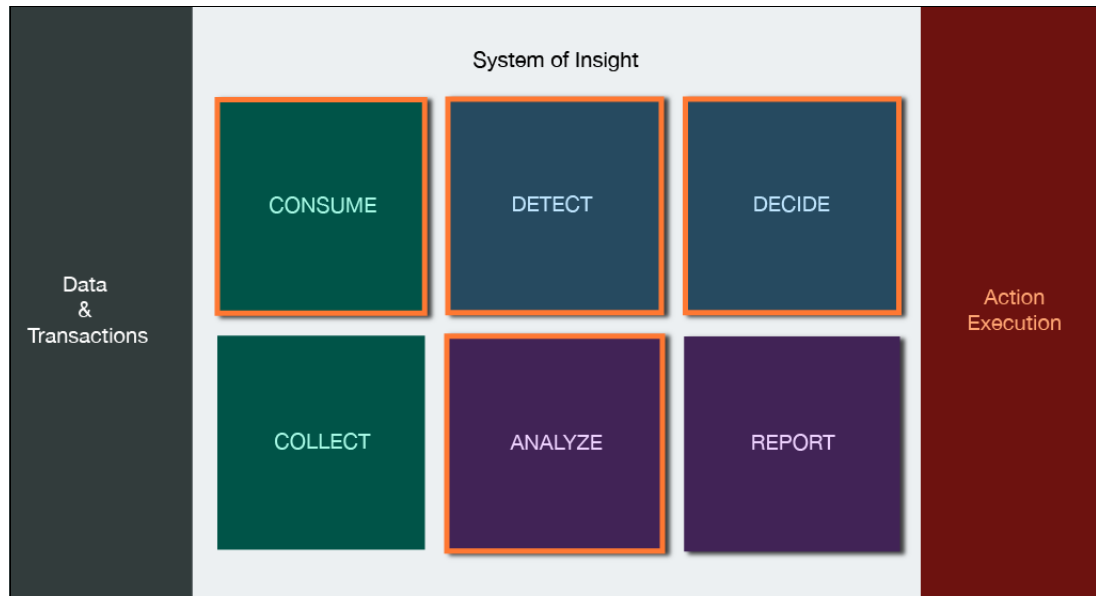


Figure 6-3 ODM fits in a system of insight

- ▶ Consume: ODM provides components capable of consuming events from different sources.
- ▶ Detect: ODM can identify patterns at the moment of action. It can correlate data and build up the rich context needed for this pattern identification.
- ▶ Decide: ODM provides mechanisms to evaluate rules or code to drive decisions. Decisions can be made in a request-driven or situation-driven approach.
- ▶ Analyze: ODM can perform aggregates on data it uses for decisions and integrates with IBM Analytics offerings for advanced analytic capabilities.

Figure 6-4 shows a breakdown of the ODM Advanced components. The DSR and DSI modules in the Decision Server are related to the runtime phase and provide different capabilities. These two modules are described separately.

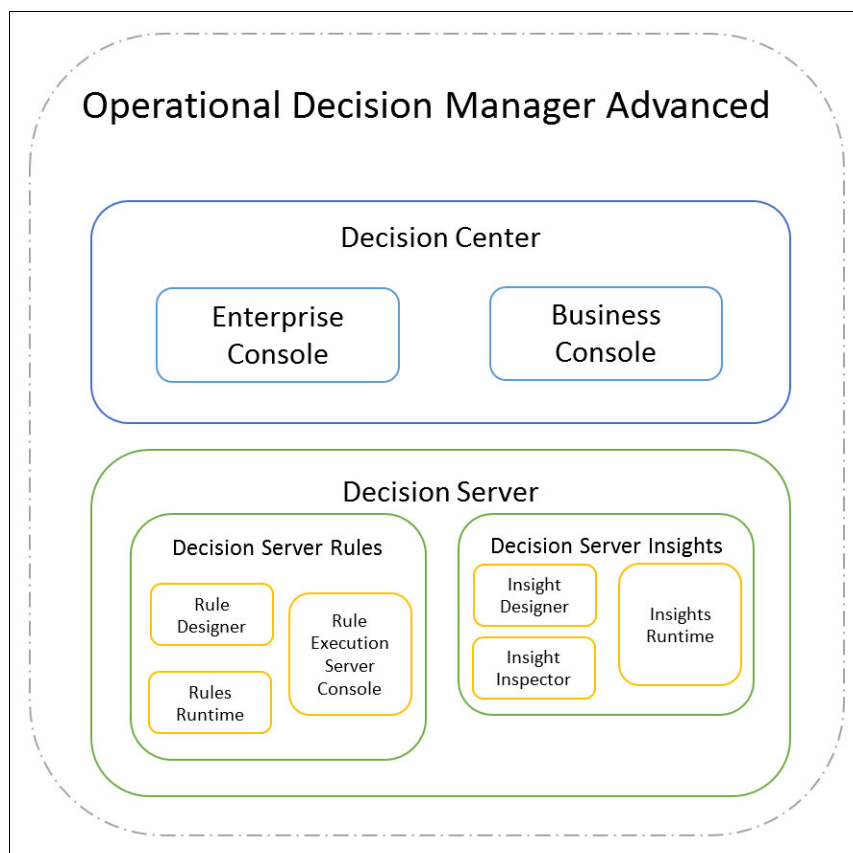


Figure 6-4 ODM Advanced components

6.2 Request-driven decisions in ODM Decision Server Rules

A request-driven decision is the most often used type of decision. It is a result of an evaluation of multiple rules against provided data to produce a final outcome. Business rules for these decisions are authored by subject matter experts who know how the business is operated.

The reason that the decisions are being referred as “request-driven” is because the rule application relies on a client application to start it. The rule application itself remains in a passive state waiting for a request to come in. A request triggers the evaluation process of business rules, which leads to a final decision returned to the client application.

Figure 6-5 shows the ODM components that are required for request-driven decisions. This section details the functions and usage of each of the components in the diagram, including the integration points between the components.

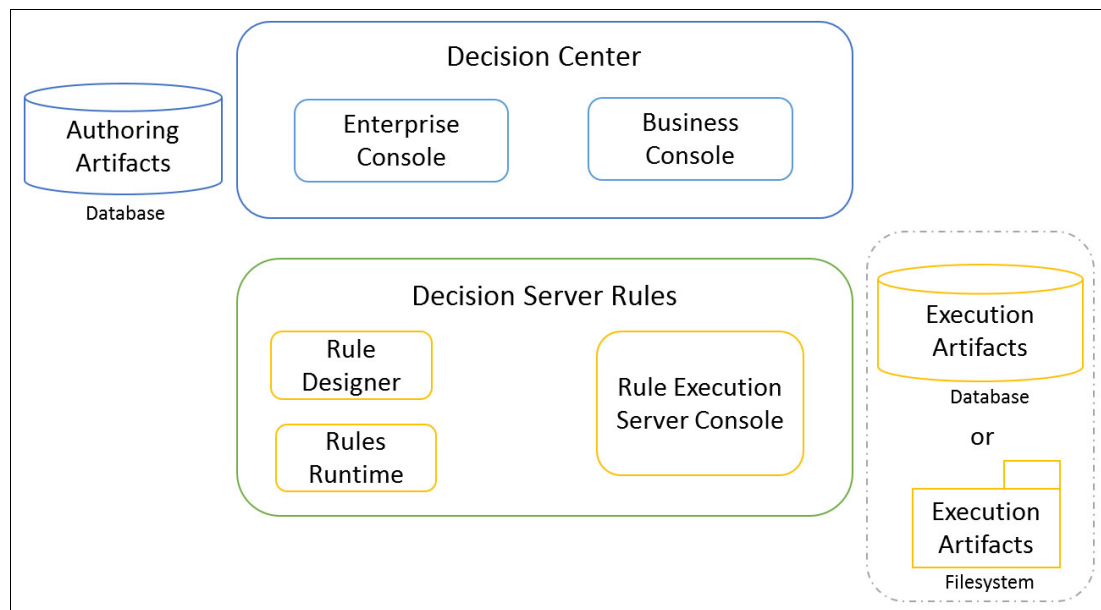


Figure 6-5 ODM components for request-driven decisions

6.2.1 Solution components

The web-based Business Console and Enterprise Console are modules of the Decision Center. Each console has unique capabilities to provide for different users.

Decision Center: Business Console

The Business Console of Decision Center is considered the preferred environment for change management for business users.

Figure 6-6 shows the home page of the Business Console where users can see at a glance all of the activities taking place in the rules repository. Home section provides a list of rules that were recently created and updated. Customers can also obtain a list of rules being followed because they were considered important at some point. The section at the bottom shows some posts, which are part of the collaboration capabilities of the Business Console.

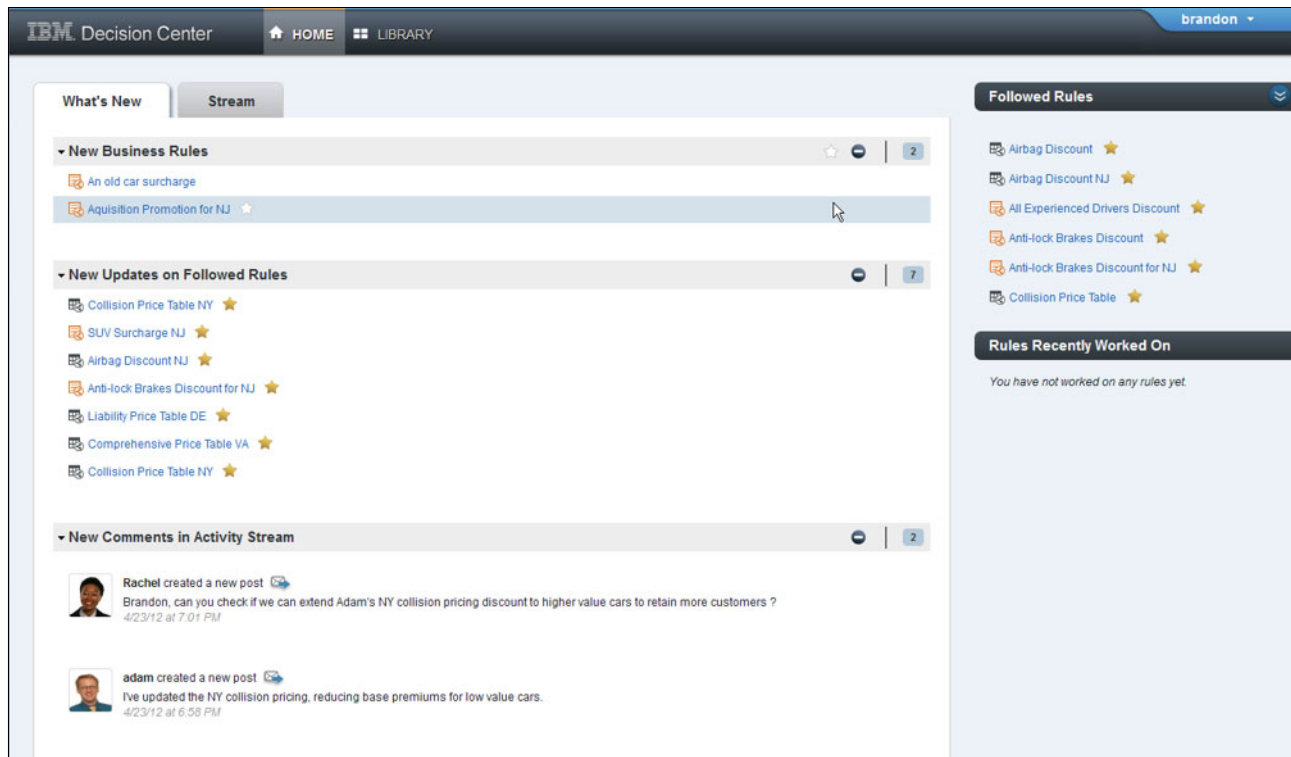


Figure 6-6 Home page of the Decision Center Business Console

The following tasks are available in the Business Console:

- ▶ Rule editing: Web editors available for rule artifacts. Rule artifacts management (create, read, update, and delete) for rule authors.
- ▶ History visibility: History of changes is available for every artifact (action rules, decision tables).
- ▶ Comparison: Action rules and decision tables side by side comparison.
- ▶ Synchronization: Rule synchronization from Rule Designer to Decision Center repository.
- ▶ Deployment: Rules deployment of RuleApps to runtime environment.
- ▶ Accessing decision governance framework to define releases and change or validation activities. Full decision governance framework management is available including associating users roles with releases and tasks.
- ▶ User collaboration through stream of messages where users can post messages.
- ▶ Notifications with a full summary of recently updated or created rule artifacts.
- ▶ Creating, updating, and deleting simulations using metrics and key performance indicators (KPIs).
- ▶ Decision Service rule projects deployment using deployment configurations.
- ▶ Snapshots management for comparing the evolution of rule projects and side by side comparison of the existing snapshots.

Decision Center: Enterprise Console

The Enterprise Console focuses more on administrative tasks related to the Decision Center repository, including setup and security configuration tasks. Enterprise Console also provides all the basic functions for rule artifacts management (create, update, and delete). Figure 6-7 shows the home page of the Enterprise Console. On the top are the configuration tabs for performing queries, analyzing tasks, and configuring the repository.

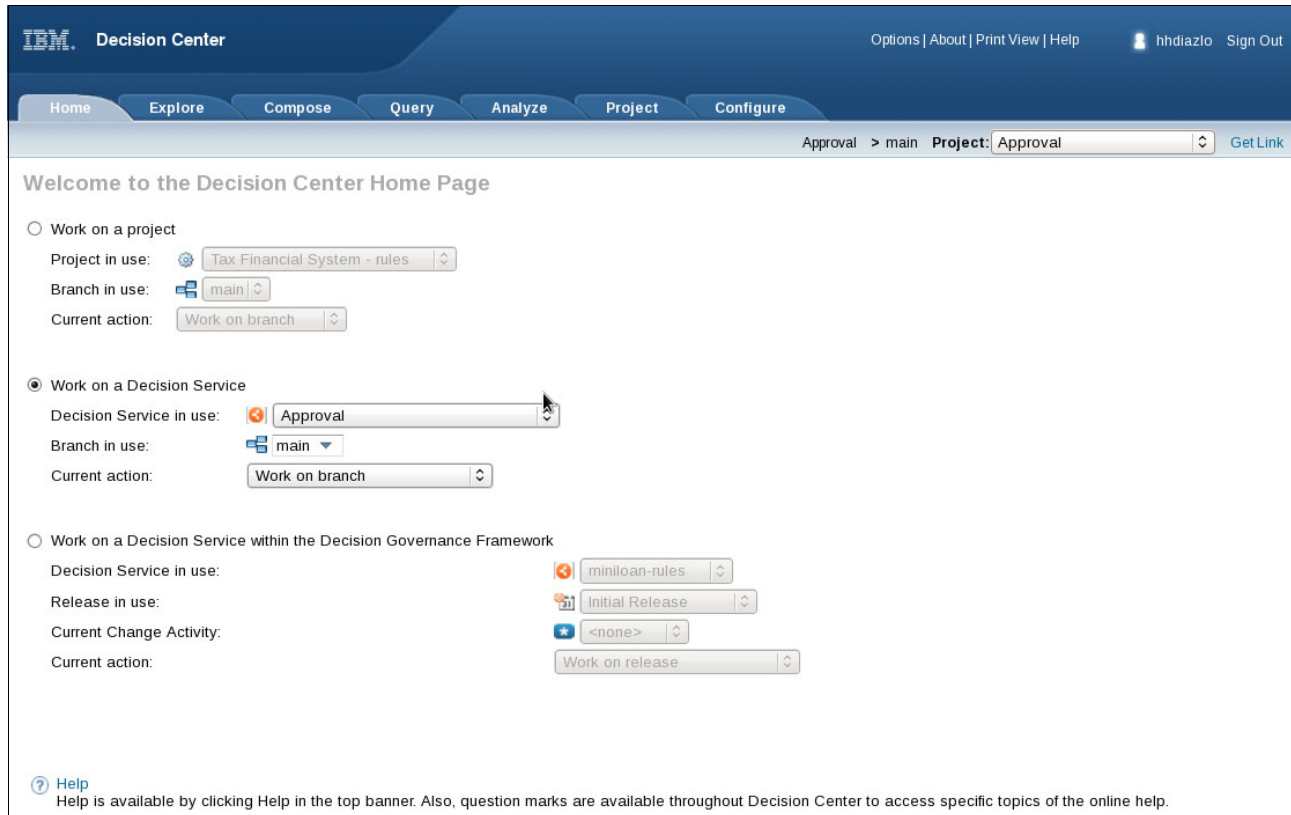


Figure 6-7 Home page of the Decision Center Enterprise Console

The following tasks are available in the Enterprise Console:

- ▶ History visibility: History of changes is available for every artifact (action rules, decision tables).
- ▶ Comparison: Action rules and decision tables side by side comparison.
- ▶ Synchronization: Rule synchronization from Rule Designer to Decision Center repository.
- ▶ Deployment: Rules deployment of RuleApps to runtime environment.
- ▶ Rule artifacts management (create, update, and delete tasks).
- ▶ Step-by-step authoring guided editor for action rules.
- ▶ Author rules of decision trees, technical rules, action rules, decision tables, queries, and smart folders. Templates are available for action rules and decision tables.
- ▶ Analyze rule artifacts with consistency check report.
- ▶ Analyze rule artifacts with completeness report.
- ▶ Generate business rules project report.
- ▶ Configure rule project and branch security.
- ▶ Configure rule permission for rule artifacts.

- ▶ Access installation wizard to define setup configuration parameters.
- ▶ Erase rule projects.

Decision Server Rules: Rule Designer

Rule Designer is the primary tool for IT roles that include developers, architects, and business analysts.

Figure 6-8 shows the tasks available in Rule Designer for full development cycle of a rule application.

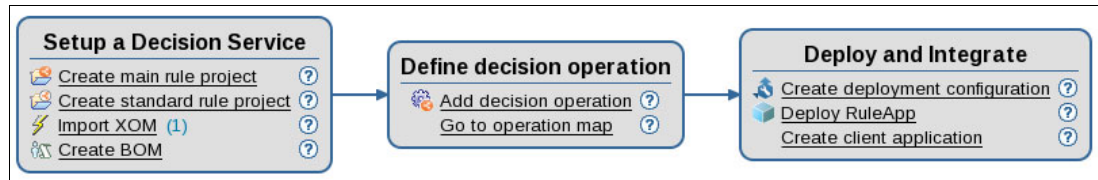


Figure 6-8 Decision Service map from Rule Designer

Rule Designer is the only place where you can create decision service rule projects, import the execution object model (XOM) to use in a business object model (BOM) definition, define the vocabulary for a BOM, and update the BOM entries.

During the authoring process, Rule Designer is used for the following activities:

- ▶ Refactor BOM to update attributes and methods from the XOM.
- ▶ Refactor verbalization when terms or phrases change in the BOM. If there are rule artifacts using these terms, rule designer can automatically update these artifacts with the new term.
- ▶ Manage rule flows.

Rule developers can test and debug rules with Rule Designer. When debugging rules, rule developers can set up breakpoints to run a rule step by step. Rule Designer also provides views to inspect variables, working memory, agenda, and so on.

For the deployment phase, you can use Rule Designer to create decision operations, to deploy configurations on decision service rule projects, and to deploy RuleApps to a Rule Execution Server as shown in Figure 6-8 inside the Deploy and Integrate box.

Decision Server Rules: Rule Execution Server (RES)

Rule Execution Server is not a single component, but an architecture based on a set of independent modules that work together to provide management, performance, security, and logging capabilities for business rule applications.

The RES includes these components:

- ▶ Execution unit: This is the main component for running RuleApps. The execution unit provides scalability, execution tracing, logging, statistics, and more functions for the RuleApp execution.
- ▶ Java EE execution components: Provides a set of rule sessions to start rule applications in stateless or stateful modes in the Java EE environments.
- ▶ Java SE execution components: Provides a set of rule sessions to start rule applications in stateless or stateful modes in the Java SE environments.
- ▶ JMX management and execution model: Provides access to management and execution JMX MBeans.

- ▶ **Management console:** Web interface that provides a single point of entry to manage deployed RuleApps. The management console includes these functions:
 - Creating and deploying new RuleApps
 - Viewing ruleset execution statistics
 - Listing decision services that are linked to rulesets
 - Monitoring ruleset execution and execution units
 - Accessing logging consoles
 - Running trace with Decision Warehouse
- ▶ **Persistence Ant tasks:** Set of management tasks used to deploy RuleApps or write directly to database.
- ▶ **Transparent decision services:** Functions to provide web services from deployed rulesets automatically (Hosted transparent decision services) or integration for Managed transparent decision services.

6.2.2 Decision Service projects

Three types of rule projects can be combined in the development of a decision service:

- ▶ *Standard Rule Project:* This is the standard project type that stores the rule artifacts (6.2.3, “Modeling and authoring artifacts” on page 79) that you create. Standard Rule Projects can be arranged into a hierarchy so that artifacts can be grouped and structured in a logical, clear fashion.
- ▶ *Main Rule Project:* This is the project that sits at the top level of the hierarchy. Standard Rule Projects can be promoted to Main Rule Projects and vice versa.
- ▶ *BOM Project:* This project contains the BOM entries for a decision service. The BOM Project ordinarily does not contain any other rule artifacts so that it can be reused in a similar fashion to a library.

Rule applications can have multiple rule projects and their organization depends on the analysis. Rule project structure is important because it helps to simplify multiple tasks during authoring or deployment, such as a security definition to limit visibility of rule artifacts for specific groups.

Figure 6-9 shows a typical diamond rule project structure that reuses BOM and authoring assets. In this example, you can see how the Main Rule Project references two rules projects for different sets of rules (rules A and rules B) that in turn, reference the same BOM.

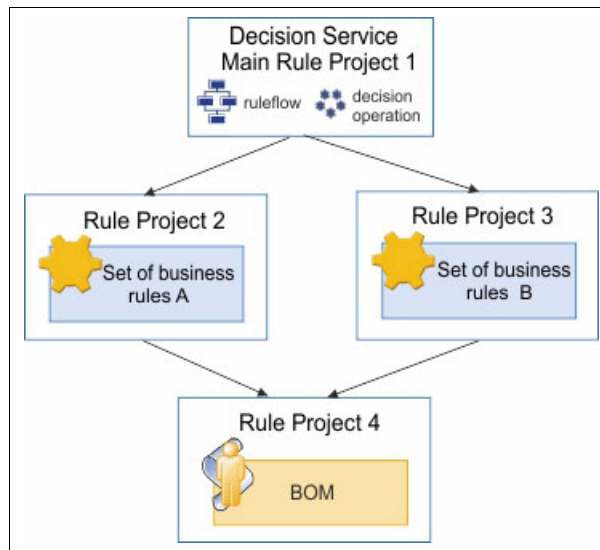


Figure 6-9 Multiple rule projects structure sample

Rule Project 2 and Rule Project 3 reference the same project Rule Project 4, which contains only a BOM. With this reference, BOM does not need to be replicated in each rule project. This configuration simplifies the management process such that every change performed to the BOM is automatically available in Rule Project 2 and Rule Project 3.

For more information about these subjects, see the following IBM Knowledge Center articles:

- Setting up rule projects:

http://www.ibm.com/support/knowledgecenter/SSQP76_8.7.1/com.ibm.odm.dserver.rules.designer.dev/developing_topics/tpc_rulep_setup_ruleproj_intro.html

- Setting up a project hierarchy:

http://www.ibm.com/support/knowledgecenter/SSQP76_8.7.1/com.ibm.odm.dserver.rules.designer.dev/developing_topics/con_ds_dev_project_org.html?lang=en

6.2.3 Modeling and authoring artifacts

This section provides an overview of some artifacts and constructs that are available in Rule Designer for modeling and authoring business rule solutions. For more in-depth information about Rule Designer, see the IBM Knowledge Center at the following web address:

http://www.ibm.com/support/knowledgecenter/SSQP76_8.7.1/com.ibm.odm.dserver.rules.designer.dev/topics/odm_dserver_rules_designer_dev.html

For more information about rule authoring, see the IBM Knowledge Center at:

http://www.ibm.com/support/knowledgecenter/SSQP76_8.7.1/com.ibm.odm.dserver.rules.designer.author/authoring_topics/tpc_authoring_brules_intro.html?lang=en

eXecution object model (XOM)

The eXecution object model (XOM) is the base for a rule application that contains the executable definition to be used during runtime for evaluating and running the rule artifacts. In

ODM, the XOM can be created from different sources. The examples in this book focus on distributed platforms where two data sources are available: Java classes and XSD schema (Figure 6-10).

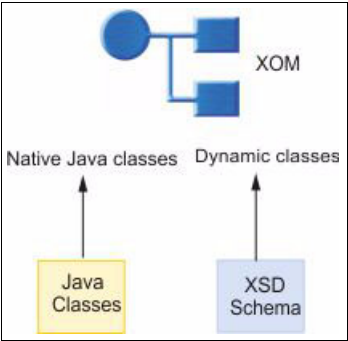


Figure 6-10 Data sources for XOM creation

Note: For ODM on z/OS, the XOM can be generated from COBOL copybooks or PL/I data sources.

Building the XOM is one of the main responsibilities of the rule developer, as the result of the rule analysis performed by a business analyst and subject matter experts.

The XOM created for the sample scenario is shown in “Creating the XOM” on page 165.

Business object model (BOM)

The BOM is the next layer in the modeling phase. It is built from the eXecution model and becomes the entry point for verbalization of terms and phrases. The BOM is constructed in Rule Designer.

Figure 6-11 shows mapping between the XOM and the BOM, which is used by rule authors during the authoring of business rules.

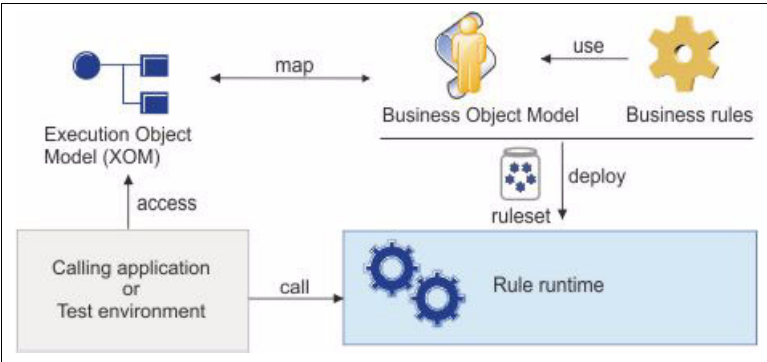


Figure 6-11 XOM and BOM mapping for business rules authoring

There are some situations when the business object model does not provide all attributes that are needed to make decisions. For example, you want to author a rule to use the age of a passenger for validation. However, a Passenger attribute only has a birthday attribute in date data type, not age. The solution is to create a BOM to XOM (B2X) implementation layer that allows you to create some code to extend the BOM with additional attributes or methods. This allows you to write rules using natural-like language phrases that are easy to understand.

Figure 6-12 graphically represents the evolution from XOM to BOM to create the age attribute for rule authoring. Extended attributes are typically referred as *virtual attributes* or *virtual methods*.

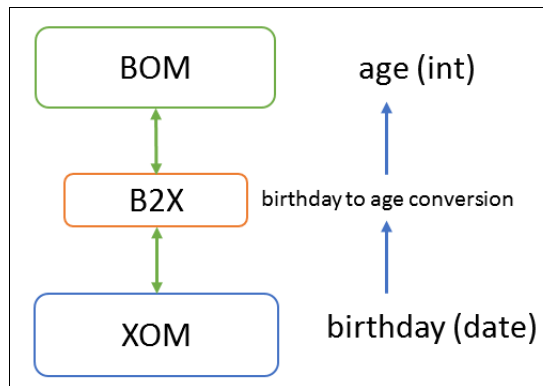


Figure 6-12 B2X example to extend XOM

An overview of BOM and execution object model is available in the IBM Knowledge Center at:

http://www.ibm.com/support/knowledgecenter/SSQP76_8.7.1/com.ibm.odm.dserver.rules.designer.author/designing_bom_topics/con_bom_overview.html

Mapping details for BOM to XOM mapping are available at the following link:

http://www.ibm.com/support/knowledgecenter/SSQP76_8.7.1/com.ibm.odm.dserver.rules.designer.author/designing_bom_topics/tpc_bom_defining_mapping_to_xom.html?lang=en

Vocabulary

ODM provides a rich rule authoring capability that enables authors to use an English-like language to define business rules. The vocabulary is made of terms and phrases that are derived from the BOM.

Figure 6-13 shows the evolution from the XOM technical model to BOM in preparation for the vocabulary. For example:

- ▶ Java class evolves to a vocabulary term: This mechanism is described with a Java class named `Loan` that translates to the vocabulary term `the loan`.
- ▶ Java method evolves to a vocabulary phrase: Java method `reject` evolves to the phrase `reject the loan with reason...`

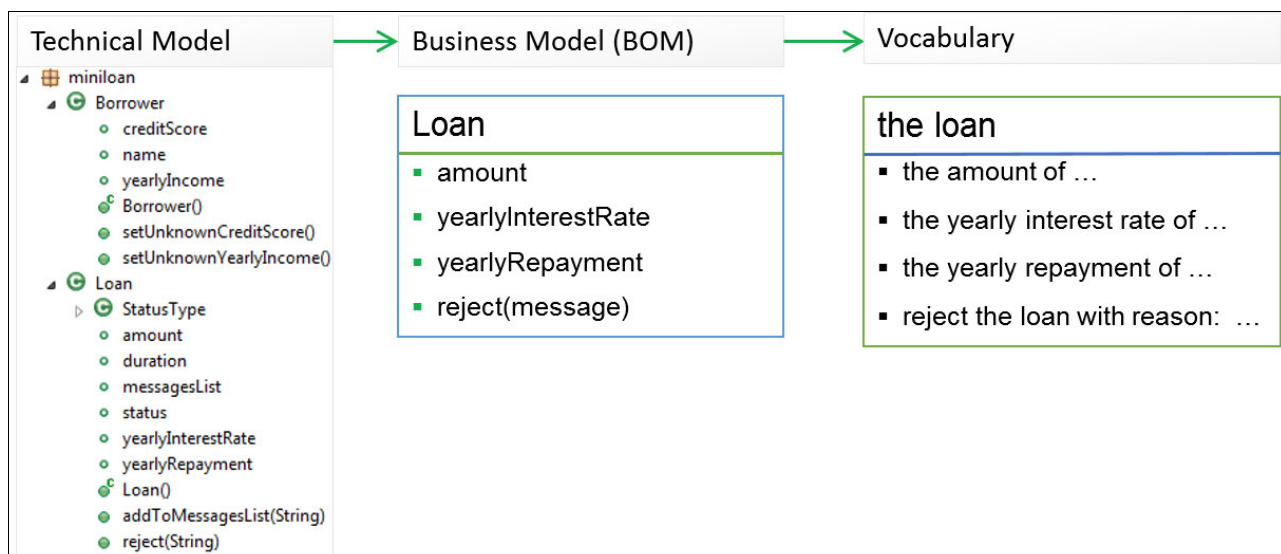


Figure 6-13 Evolution from technical model to vocabulary example

After vocabularies are defined, rule authors can start creating rule artifacts. When an attribute or method is not verbalized in the BOM, it is not available for rules authoring. More information about defining a vocabulary is available in the IBM Knowledge Center at:

http://www.ibm.com/support/knowledgecenter/SSQP76_8.7.1/com.ibm.odm.dserver.rules.designer.author/config_auth_topics/tpc_rd_bom_defining_vocabulary.html?lang=en

Vocabulary can also be defined in different languages, which allows multi-language rule authoring depending on the locale used by the tool. This function allows two teams to work together with the same set of rules, for example one team using an English verbalization while the second team works in French. Information about opening Rule Designer in a specific locale is available in the IBM Knowledge Center at:

http://www.ibm.com/support/knowledgecenter/SSQP76_8.7.1/com.ibm.odm.distrib.install/starting_topics/tsk_start_rule_designer_locale.html?lang=en

BOM entry

A *BOM entry* is the representation of your complex business objects (that is, anything more complex than a primitive type such as string or int). There are two options for building the BOM:

- ▶ **Top-down:** Create the business elements up front without necessarily considering how the execution elements will be implemented. They can be mapped later when the implementation is completed.
- ▶ **Bottom-up:** The BOM is created from the XOM. You can then extend the BOM later if necessary in the same manner as the top-down approach. DSR supports BOM creation from Java, XSD, and mainframe options (for example, COBOL, PL/I).

The only terms (equivalent to classes) available in Business Action Language (BAL) are those defined by BOM entries. You can also define variables based on primitive types.

The BOM entry created for the sample scenario is shown in “Creating the BOM” on page 166.

Rule language: Business Action Language (BAL)

BAL is the rule authoring language that is used in ODM. BAL provides constructs to build rules; operators to perform arithmetic operations, string operations, and negate conditions; BAL literals; and punctuation rules. Here are some examples:

- ▶ BAL constructs: if, then, else, set, all of the following conditions are true, any of the following conditions is true
- ▶ BAL logical operators: and, or
- ▶ BAL date operators: is after, is before, is in month, is in year
- ▶ BAL number operators: is more than, is less than, is at most, is at least

Details about BAL are available in the IBM Knowledge Center at:

http://www.ibm.com/support/knowledgecenter/SSQP76_8.7.1/com.ibm.odm.dserver.rules.ref.designer/lang_bal_ref_topics/tpc_bal_intro.html?lang=en

Action rule

An *action rule* is a simple BAL sentence that describes the business logic. An action rule is composed of four parts:

- ▶ **definitions** (definitions, optional): Allows definition of variables for the rule.
- ▶ **if** (conditions, optional): Allows specification of the required conditions for the actions to be triggered. If the conditions are not specified, the actions will always trigger.
- ▶ **then** (actions, mandatory): What will be done if the conditions are met.
- ▶ **else** (alternative actions, optional): What will be done if the conditions are not met.

Example 6-1 shows these four parts.

Example 6-1 Example Action Rule

```
definitions
  set vegetarian to a passenger
    where the dietary requirement of the passenger is vegetarian
if
  the number of vegetarian meals is more than 0
then
  offer a vegetarian meal
else
  offer a vegan meal
```

ODM provides different editors to create action rules in different tools:

- ▶ Guided Editor: Provides a step-by-step authoring experience with a graphical editor that guides the author through drop-down menus with possible options based on vocabulary or BAL constructs. Guided Editor is available in Rule Designer and Enterprise Console.
- ▶ Intellirule Editor: Provides an open text area for constructing action rules with a syntax check and color highlighting for valid constructs. Intellirule Editor is available in Rule Designer, Business Console, and Enterprise Console.

Decision table

A *decision table* is a more concise representation of a large selection of rules that have the same structure (that is, they are symmetric). This means that the decision tables typically use the same attributes of the model in their condition and action columns. However, these condition columns tend to span a range of values in a specified domain or numeric thresholds, with each row representing a specific set of these values. Figure 6-14 shows an example of a decision table.

Decision table editor is responsible for automatically building the action rule that is associated to each row into it. Action rules per row are visible by hovering the mouse over the row index of the row as shown in Figure 6-14.

	Product name	Order date	Discount %
1	Product A]6/1/2015; 6/30/2015[15
2	Product B]12/1/2009; 12/25/2009[5
			2

if
all of the following conditions are true :
- (the product name is "Product B")
- (the order date is after 12/1/2009 and before 12/25/2009) ,
then
decrease price by 5 % ;

Figure 6-14 Decision table example

Some of the benefits of decision tables are that the editor will automatically validate situations where gaps or overlaps are present. The rule author can then decide whether a gap situation is tolerable or if it needs to be considered an error to be corrected before the rules can be deployed.

A decision table used in the implementation sample is shown in “Creating a decision table” on page 168.

Rule packages or folders

Contents of rule projects are organized in multiple folders or packages. In Rule Designer, rule projects are organized as packages, whereas in Decision Center, rule projects are organized in folders. Folders and packages are just a way to organize assets that helps in accessing rule artifacts quickly, especially when applications have a large number of rules.

Some possible folder organizations are by Country, State, Region, Product Line, Business Unit, and Type of Decision.

Figure 6-15 shows a rule project with two folders (or packages): Eligibility and validation. This example assumes that all rule artifacts related to the eligibility function must be added under the eligibility folder, and similarly the validation function with the validation rule artifacts.

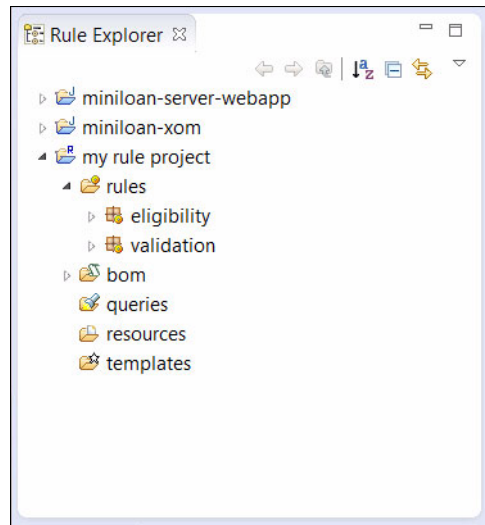


Figure 6-15 Example package organization in a rule project

Ruleflow

A *ruleflow* in DSR orchestrates the invocation of other rule artifacts. Orchestration means controlling the sequence of the decision making between these rule artifacts, such as action rules or decision tables. In a decision operation, you select a ruleflow as the main ruleflow, which dictates the decision service entry point (see “Decision operation” on page 86). Figure 6-16 shows an example ruleflow.

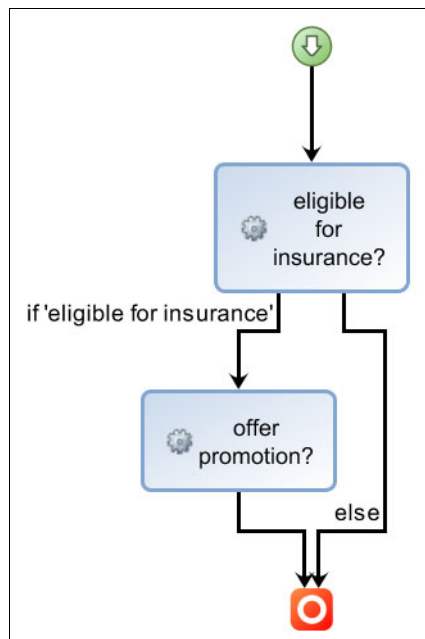


Figure 6-16 Example ruleflow

A ruleflow includes the following most common components:

- ▶ *Start node*: Represented by a circle with a down arrow. This is the entry point to the ruleflow.
- ▶ *End node*: Represented by a box with a circle. This is the exit point from the ruleflow.
- ▶ *Rule tasks*: Represented by a box with a cog. These are where the business logic (for example, action rule or decision table) is run.
- ▶ *Transitions*: Represented by an arrow between other components. These control the flow between the other nodes. They can branch based on a condition.

A preferable practice in orchestrating rule artifacts with ruleflows is to relate rule tasks to folders. This approach allows rule authors to add as many rule artifacts as they need without any ruleflow impact because the rule task will always consider all rules under the defined folders.

For steps to create a ruleflow for the sample scenario, see “Creating a ruleflow” on page 168.

For more information about orchestrating ruleset execution, see the IBM Knowledge Center at:

https://www.ibm.com/support/knowledgecenter/SSQP76_8.7.0/com.ibm.odm.dserver.rules.designer.dev/orchestrating_topics/tpc_orch_intro.html

6.2.4 Rule project artifacts

Rule project artifacts are those that support development and deployment of a ruleset, but do not necessarily contain business logic themselves. For information about how to create a rule project for the sample scenario, see “Creating the Rule Project” on page 166.

Decision operation

A decision operation is the entry point for a ruleset. A decision operation defines a ruleset and its signature in terms of parameters. Ruleset parameters are used by the rule engine to communicate with the outside world. They are used to provide data from the client application to the rule engine. Ruleset parameters have the following defined direction:

- ▶ **Input**: Indicates that the parameter is an input only.
- ▶ **Input - Output**: Indicates that the parameter will provide incoming data that can be updated by some rules in the process and sent back to the client application.
- ▶ **Output**: Indicates that the specified data will be sent back to the client application from the rules.

Decision operations also define the entry point for a ruleset through the selection of a main ruleflow. This ruleflow is used by the rule engine to evaluate the rules for the ruleset.

For the scenarios that do not require all rules from a rule project to be included in a ruleset, you can use a query to select rules that you need. An example is when a rule project has a specific rule lifecycle, then only rules with specific status need to be included in the ruleset when deploying the ruleset.

Decision operations are usually scoped around a business decision (for example, deciding whether to offer a customer a loan).

For directions on how to create a decision operation for the sample scenario, see “Creating the decision operation” on page 169.

Ruleset

A ruleset is an executable container of rule artifacts and some additional elements such as parameters information, BOM vocabulary, and B2X files. When using a dynamic XOM, the schema definition file is also part of a ruleset.

Rulesets contain rule artifacts in ILOG Rule Language (IRL) and are created from a decision operation.

Deployment configuration

Deployment configurations govern deployment of a decision service to a Rule Execution Server. Deployment configurations contain the following items:

- ▶ Configuration type (production or non-production). This can be used to limit deployment to a certain RES.
- ▶ The RuleApp name. This is what the client application uses to call the Decision Service.
- ▶ The Decision Operations required for this Decision Service. These contain the business logic.
- ▶ The target servers.
- ▶ Versioning information (base version number and versioning policy).

A deployment configuration is used to create a RuleApp from a set of decision operations. A deployment configuration can be of a non-production or production type, and contains a list of target servers that the RuleApp can be deployed to.

When defining operations for a deployment configuration, a user can define ruleset properties that can be used to enable tracing functions.

Ruleset version policy is another feature defined in the deployment configuration.

For information about how to create a deployment configuration for the sample scenario, see “Creating the deployment configuration” on page 170.

For more information about deployment configurations, see the IBM Knowledge Center at:

http://www.ibm.com/support/knowledgecenter/SSQP76_8.7.1/com.ibm.odm.dserver.rules.deploying/shared_cmgtopics/con_cmgt_deployconf_overview.html?lang=en-us

RuleApp

A RuleApp is a deployable management unit that contains a list of rulesets. It is generated from a deployment configuration definition in a decision service rule project.

A RuleApp is a file and can be stored in any file system or source code repository for continuous integration build processes.

RuleApp creation is a task that can be automated by using a set of provided Ant tasks. For more information, see the IBM Knowledge Center at:

http://www.ibm.com/support/knowledgecenter/SSQP76_8.7.1/com.ibm.odm.dserver.rules.ref.res/ant_tasks/tpc_ant_tasks_ruleapps_intro.html?lang=en

6.3 Situation-driven decisions in ODM Decision Server Insights

Situation-driven decisions are those where you likely do not know that a decision must be made until a series of separate criteria are met. In these decisions, the information needed to make the decision evolves over time from potentially disparate sources. DSI, the ODM component that supports decisions where history and context are important, offers the capability to detect situations. DSI consumes all information sent as events and stores it as part of the context to build the history needed for future decisions.

DSI is an event processing system with rule-based capabilities for situation detection and decisions. DSI provides an agent driven architecture that allows implementation of business type artifacts such as action rules or Java based artifacts for complex decisions. It also provides integration with IBM SPSS Scoring services to enhance rule logic with predictive analytics.

DSI provides a scalable architecture that can grow to meet an organization's specific requirements relevant to detecting situations at the time of interaction. The main goal for DSI is to anticipate situations before they occur.

Figure 6-17 shows the main components of Decision Server Insights.

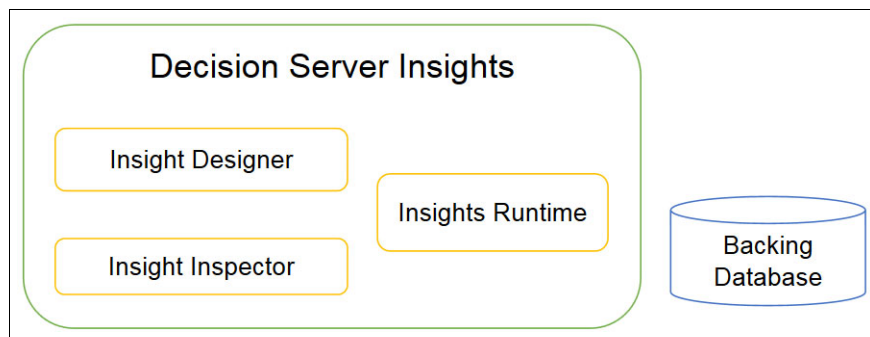


Figure 6-17 Decision Server Insights main components

6.3.1 Decision Server Insights components

Decision Server Insights consists of the components and tools that are needed for solution implementation, execution, and monitoring. This section covers the following components:

- ▶ Insight Designer
- ▶ DSI runtime environment
- ▶ Insight Inspector
- ▶ Backing database

Insight Designer

Insight Designer is an Eclipse-based integrated development environment (IDE) for creating and maintaining DSI solutions. Figure 6-18 shows the development cycle tasks, modeling, authoring, and deployment provided in the tool.

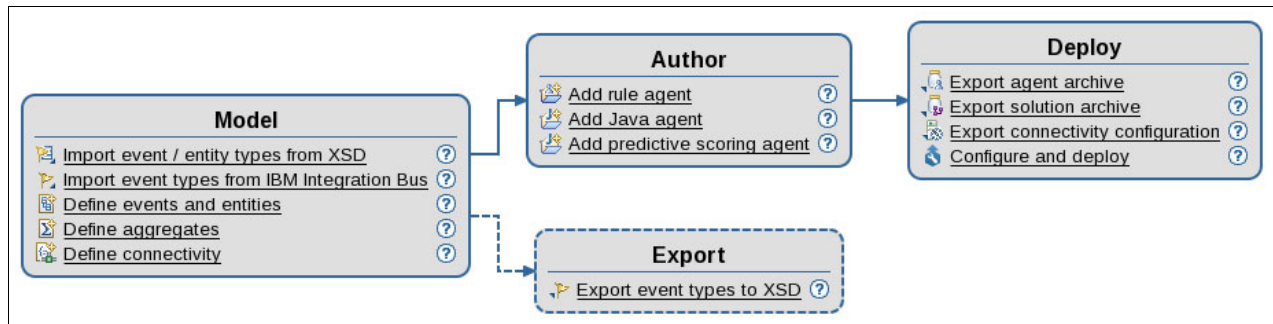


Figure 6-18 Insight Designer solution map

Insight Designer provides various views and wizards for building solutions. You can use Insight Designer to create the following project types:

- ▶ Solution project
- ▶ Rule Agent project
- ▶ Java project
- ▶ Predictive Scoring Agent project
- ▶ Extension project

Insight Designer also provides capabilities to create the following artifacts:

- ▶ Business model definition
- ▶ Transformation file
- ▶ Connectivity definition
- ▶ Global aggregate
- ▶ Data provider extension
- ▶ Entity Initialization extension
- ▶ Action rule

DSI runtime environment (Insights Server)

Decision Server Insights is designed for scalability and high availability, and does not require shutdown during product minor version upgrade or system updates. The DSI run time is also referred as Insights Server. It can be deployed as one or many servers depending on the topology or usage required. For more information, see 9.5, “ODM Decision Server Insights topologies” on page 222.

The DSI runtime server stack supports the following components and capabilities (as shown in the light background color in Figure 6-19 on page 90):

- ▶ Event aggregates
- ▶ Entity aggregates
- ▶ Rule agents
- ▶ Java agents
- ▶ Predictive scoring agents
- ▶ Connectivity (Java API, JMS, and HTTP)
- ▶ Decision engine, scheduling, state management
- ▶ Management and monitoring

The following list includes internal components that are part of DSI run time, which are highlighted in Figure 6-19 with a shaded background color:

- ▶ Elastic entity store (WebSphere eXtreme Scale)
- ▶ Compute grid (WebSphere eXtreme Scale and X10)
- ▶ WebSphere Liberty

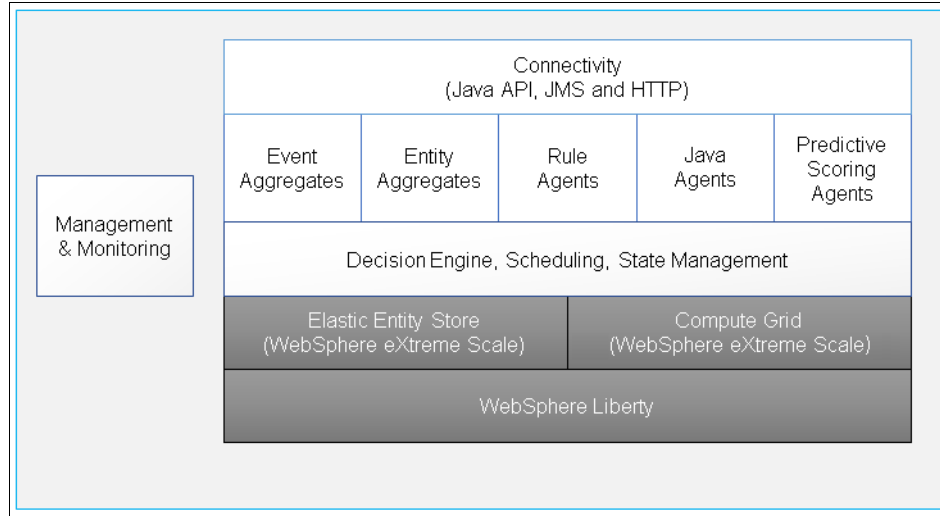


Figure 6-19 Decision Server Insights runtime architecture

For development purposes, all capabilities and internal components from Figure 6-19 are part of a server template called `cisDev`.

For production topologies, components are separated into different server types to provide scalability and high availability. For information about these server types, see 9.5.1, “Component definitions” on page 222.

Insight Inspector

Insight Inspector is a web tool that provides a graphical user interface for analyzing solution operation.

Figure 6-20 shows the interface of Insight Inspector. Checking the solution timeline helps to analyze events sequence and the data received by the events. Insight Inspector also provides information about entities, which is helpful when the outcome of an action rule requires updating specific entities. The Initial Value and Final Value provided on the lower right section provides a side-by-side comparison.

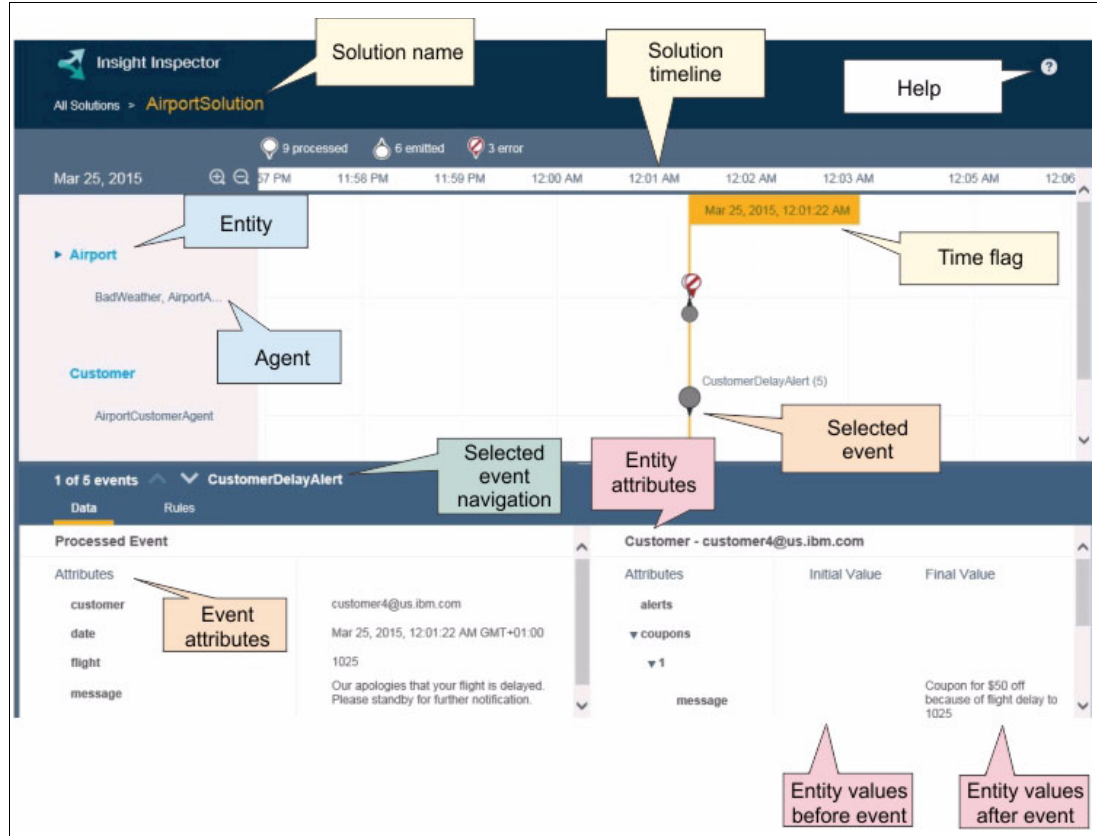


Figure 6-20 Insight Inspector console

Backing database

DSI can store the runtime state of the server in a database that includes entity business information, in-progress transactions, and the history of events and event data.

The following modes define when to write changes to the database:

- ▶ Write-through (Synchronous)
- ▶ Write-behind (Asynchronous)

Configuring the backend database to be used by DSI is a two-step process:

1. Database configuration: Run the script provided by ODM on the database server to configure the database including setting up the JDBC connectivity settings in the Container servers.
2. Object grid configuration: Modify the object grid configuration to specify the write mode. DSI provides two configuration templates: One for write-through mode and another for write-behind mode.

The steps to configure the database and the grid for database persistence are available in IBM Knowledge Center at the following web address:

http://www.ibm.com/support/knowledgecenter/SSQP76_8.7.1/com.ibm.odm.itoa.config/to/pics/tsk_config_deploy_loaders.html?lang=en

6.3.2 Business model definition

The BOM for a DSI solution is created from two possible sources:

- ▶ A business syntax file called business model definition (BMD).
- ▶ An XML schema with definitions of events and entities that follows annotation guidelines.

The possible contents of a BMD are as follows:

- ▶ Concept: A generic representation of an object that can have multiple attributes. Entities and events can have multiple concepts.
- ▶ Entity: A business object that describes an actor in a system with an identifier. An entity can contain attributes and relationships to other entities.
- ▶ Attributes: Used to define characteristics that describe entities, events, and concepts. Attributes can have a type, which can be explicit or implicit.
- ▶ Relationships: Defined between object types such as concepts, entities, and events.
- ▶ Event: An event is a business object with a time stamp that describes an occurrence or change in state.
- ▶ Enumerations: Refers to concepts with a predefined set of values.
- ▶ Data providers: Used to enrich attributes. Data providers are considered services that provide an output value based on the inputs.
- ▶ Mandatory and default values: Capability available for creation of events or entities to define attributes that can be optional or mandatory.
- ▶ Target namespaces: When events are to be exported, the target namespace can be defined.
- ▶ Facets: Used to define spatial or time behavior into concepts or events. For example, events that are identified by a time facet can be directly used with time operators.
- ▶ Gender: Used for languages with masculine and feminine grammar requirements.
- ▶ Dictionary: Used to define the translation for terms.

Example 6-2 shows a business model definition that illustrates the usage of these terms.

Example 6-2 Business model definition example

--Enumerations

an age can be one of: CHILD, ADULT, SENIOR.

-- Concept & Gender

a gate is a concept [gender : feminine].

-- Attributes

gate has a gate number.

-- Concept inherited from concept

an airport gate is a gate.


```
-- Entity
a passenger is a business entity identified by a passenger id.
a booking is a business entity identified by a booking id.

-- Relationship
a booking is related to a passenger.

-- Events
a late passenger is a business event time-stamped by a timestamp.

--dictionary
Spanish dictionary:
use pasajero for passenger with gender masculine and plural pasajeros,
```

BOM definition is one of the first tasks that are required to create a solution because it defines the basic building blocks for all the other artifacts that will be created. As shown in Figure 6-21, events and entities are used by agents to implement the decision logic.

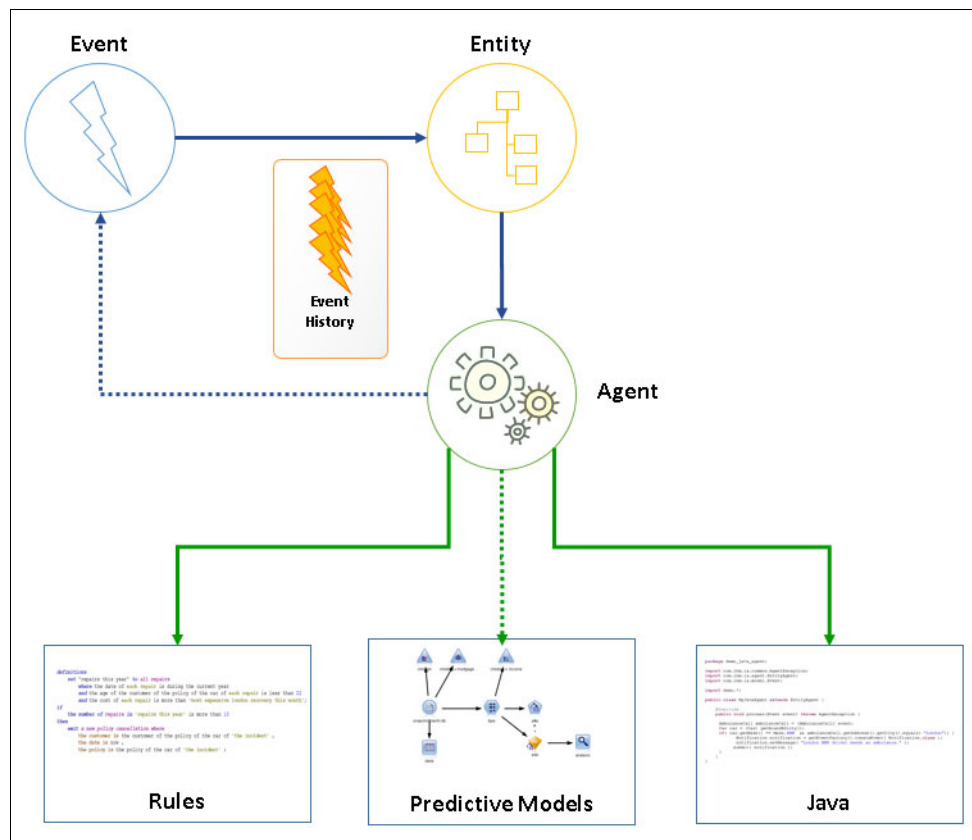


Figure 6-21 DSI programming model

During the building of the business model definition, DSI is continuously building and rebuilding the Java model for every entity, event, and relationship. This process is automatic, and is triggered every time the BMD file is saved. Details about how the translation takes place between the model definition and Java classes are available in the IBM Knowledge Center at the following link:

http://www.ibm.com/support/knowledgecenter/SSQP76_8.7.1/com.ibm.odm.itoa.develop/modeling_topics/con_java_model_project.html

DSI can also export event definitions to XML Schema Definition (XSDs) from the BMD. The process to generate these files is described in the IBM Knowledge Center at the following link:

http://www.ibm.com/support/knowledgecenter/SSQP76_8.7.1/com.ibm.odm.itoa.develop/modeling_topics/tsk_export_xsd_from_bom.html

6.3.3 Agents

In DSI, Agents are where your business logic is run. Agents handle create, read, update, and delete operations on the Entities, one of the key capabilities to build the context.

This section shows an agent's ability to detect a complex situation, and make a business decision based on this situation.

In contrast to DSR where many rule artifacts are available (see 6.2.3, "Modeling and authoring artifacts" on page 79), DSI is much simpler in terms of structure in the decision making. DSI has agent descriptors, which determine when an agent is run, and then the agent's logic itself. How this logic is run depends on the type of agent, but conceptually it is the same across all types.

Agent descriptors

All agents have an agent descriptor file, called `agent.adsc`, in the agent project root. This descriptor file tells the agent which event types to listen for, and what the agent's bound entity is. Additionally, you can do some filtering to exclude certain events that you are not interested in, and set a specific time horizon for each event type that you process. Example 6-3 shows all of these concepts.

Example 6-3 Example agent descriptor showing all the possible concepts

```
'candy store agent' is an agent related to a product,
processing events :
    - sold lollipop when the flavour of this sold lollipop equals "strawberry",
where this product comes from the barcode of this sold lollipop
    - sold icecream, where this product comes from the barcode of this sold
icecream, with a horizon of 30 minutes
```

The agent descriptor example includes the following components:

- ▶ **'candy store agent' is an agent related to a product:** This describes the bound entity type for the agent.
- ▶ **processing events: - sold lollipop:** This describes the interested event types.
- ▶ **when the flavour of this sold lollipop equals "strawberry":** This is a filter to obtain only the events where this condition is true.
- ▶ **where this product:** This is the bound entity.
- ▶ **comes from the barcode of this sold lollipop:** This gives the identifier for the bound entity.
- ▶ **with a horizon of 30 minutes:** This gives a specific time horizon for this type of event (defaults to solution property `maxTimeHorizon` if not set).

Agent types

There are three types of agents in DSI:

- ▶ Rule agent
- ▶ Java agent
- ▶ Predictive scoring agent

Table 6-2 contains a quick comparison of the three.

Table 6-2 Quick comparison of Rule, Java, and Predictive Scoring agents

Agent type	Language	Bound entity?	Event history?	SPSS integration?
Rule agent	Business Event Rule Language (BERL)	Always	Yes	No
Java agent	Java	Optional	No	No
Predictive scoring agent	Java	Optional	No	Yes

The following sections describe when each type of agent is most useful.

Rule agents

Rule agents take advantage of DSI's powerful natural language-like BERL, which has built-in support for powerful temporal and geospatial logic. Rule Agents also support the collection and retention of event history, so you can create powerful logic and aggregates looking back over historical events, enabling even more complex situation detection than with entity attributes alone. The capabilities of the rule agents can be expanded even further with OSGi service integration, which is described in more detail in Appendix A, "Integration of ODM Decision Server Insights with custom services" on page 231.

From an action rule within a rule agent, the following access is available for entities and events:

- ▶ Read and write access to all entity attributes the agent is bound to.
- ▶ Read access to attributes from remote entities that are the ones being accessed through a relationship.
- ▶ Read access to event attributes

For more information about rule agents, see IBM Knowledge Center at the following web address:

http://www.ibm.com/support/knowledgecenter/SSQP76_8.7.1/com.ibm.odm.itoa.develop/topics/tsk_create_rule_agent_intro.html

When to use them

Rule agents are most useful in the following scenarios:

- ▶ When your business logic changes often
- ▶ When your business logic contains complex temporal or geospatial elements
- ▶ When you need to look back across historical data to make a decision
- ▶ When you want to use Event Aggregates
- ▶ When your business logic requires audit by business users

Java agents

Java agents are agents that are coded in pure Java. They do not need to be bound to an entity, so they are efficient for stateless or read-only operations. Additionally, they are efficient for simple stateful logic such as updating the fields on an entity. However, consider whether your business logic is likely to change because it is far simpler to update the logic of a rule agent.

For more information about Java agents, see IBM Knowledge Center at the following web address:

http://www.ibm.com/support/knowledgecenter/SSQP76_8.7.1/com.ibm.odm.itoa.develop/topics/tsk_create_java_agent_intro.html

When to use them

Java agents are often most useful in the following scenarios:

- ▶ When you are processing events with simple, static logic
- ▶ When you need to perform stateless or read-only operations
- ▶ When you are writing complex logic or computations

Predictive scoring agents

Predictive scoring agents are based on Java agents. The predictive scoring agent construct provides a set of tools for direct integration with an SPSS scoring service, with minimum configuration required during development.

Insight Designer provides a scoring wizard available during the predictive scoring agent definition to connect to and configure the scoring service. At the end, the wizard creates a new predictive scoring agent project with some additional artifacts ready to be updated with additional logic related to the solution.

Using a predictive scoring agent with IBM SPSS offers the following benefits:

- ▶ An additional API is available to simplify the scoring invocation with SPSS. Developers do not need to manually connect to the web service and generate a Java interface through the Web Services Description Language (WSDL) service.
- ▶ Developers do not need to manage the connectivity because it is automatically configured through the scoring wizard into the DSI runtime server (Insights server).
- ▶ The scoring wizard directly connects to the SPSS Collaboration and Deployment Server to look up all the metadata for the scoring configuration that is used to produce the Java class to be used by developers to specify the source of the fields at run time. Developers must define whether the required field is an attribute from an entity, a remote entity, an aggregate, and so on.

The time required to implement a predictive scoring agent is minimal compared to building and integrating full functionality into a Java agent from scratch.

If the scoring service is provided by different products other than IBM SPSS, then integration must be achieved through a standard Java agent or OSGi service. For more information, see Appendix A, “Integration of ODM Decision Server Insights with custom services” on page 231.

Implementation information for an example predictive scoring agent is described in 8.5.4, “Predictive analytics agent implementation” on page 159.

For more information about predictive scoring agents, see IBM Knowledge Center at the following web address:

http://www.ibm.com/support/knowledgecenter/SSQP76_8.7.1/com.ibm.odm.itoa.develop/topics/tsk_dev_predictive_agent.html

When to use them

Predictive scoring agents are often most useful in the following scenario:

- When you need to make a call to an SPSS scoring service on every agent invocation

6.3.4 Lifecycle of an event

Figure 6-22 shows a flow for the lifecycle of a typical event (Event A) in DSI.

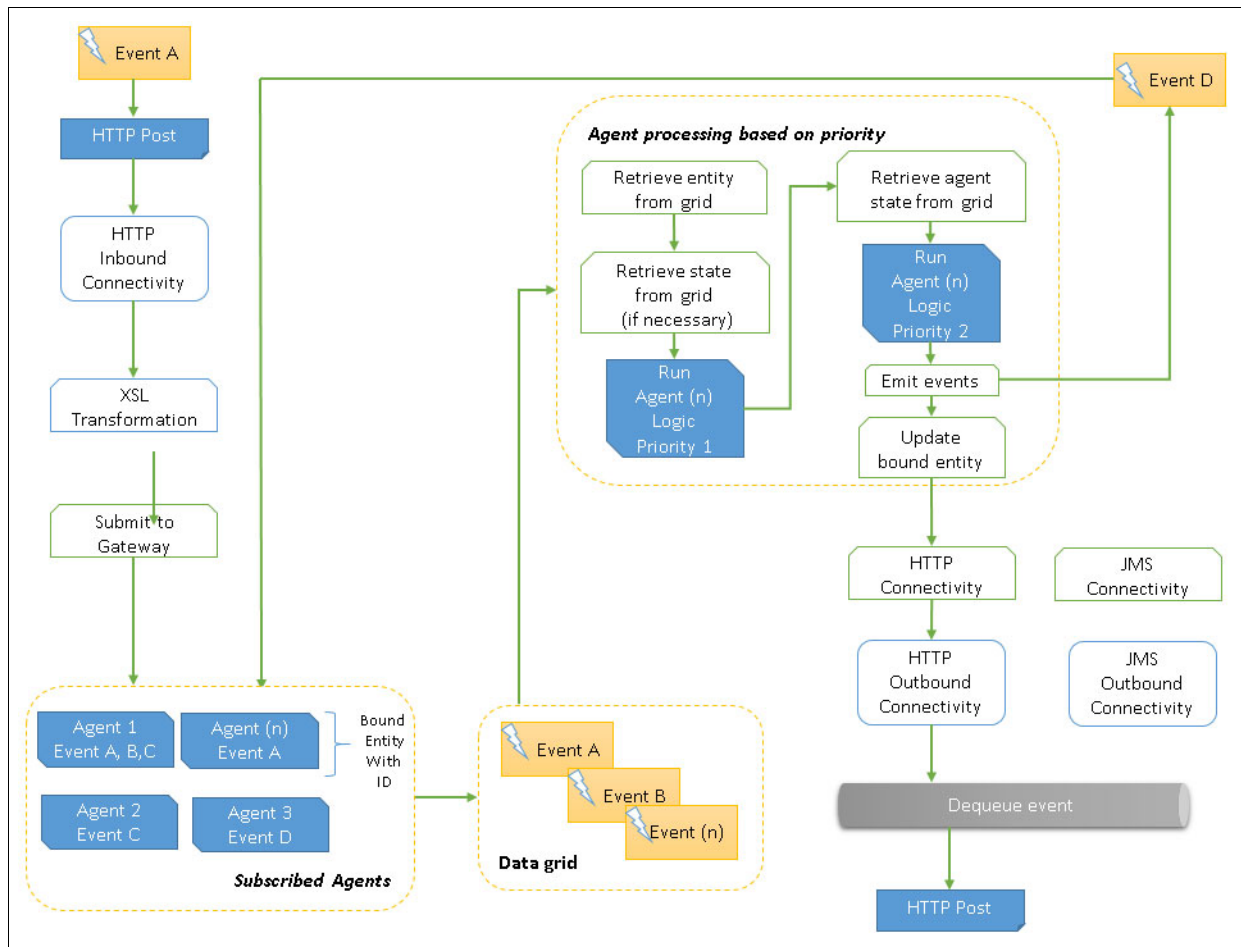


Figure 6-22 Lifecycle of an event in DSI

The event is submitted through an HTTP POST to the inbound connectivity layer. Events can also be submitted by using JMS or Java. This layer performs any transformations that are specified and then passes the event to the gateway. An event can then be filtered out if there are no agents subscribed to it.

If there is at least one type of agent subscribed to the event, then the event is put into the data grid to await processing.

Events are then processed in the order of receipt (on a per partition basis, so this FIFO behavior should not depend on solution logic). When an event is processed, it is picked up by the highest priority agent that is subscribed to that event. The agent retrieves the state from the grid if necessary (including entity and event history), performs its logic, and writes the updated state back to the grid. This process is repeated for all lower priority agents in order.

Any derived events sent from agent processing are picked up and filtered in the same way. If an outbound endpoint is configured to deliver this type of event, then the event is queued for delivery and dispatched.

6.3.5 Building context

This section describe the ways that you can build knowledge about the state of the world in a system of insight. The value for a system of insight is in correlating information coming to the system with the data it has stored during a specific time period. The combination of all this data and history of previously collected data is considered the *context*. DSI, for example, builds its context from entities, events history, and global aggregates.

Entities

In every solution, entities represent the data that is required for event processing. Entities contain a set of attributes that describe their state, which can be updated by agents during processing. DSI uses the context to correlate data based on defined identifiers to evaluate rules that drive identification of patterns.

At design time, when you create the business model definition, you define attributes associated to entities and events. When building the model, you might have some attributes that must be initialized. DSI allows initialization of attributes from incoming events.

Example 6-4 shows how to initialize an account entity from an open account event. The initialization also includes additional information for the specific attributes credit limit, and customer.

Example 6-4 Entity initialization from an event

an account is initialized from an open account event, where this account comes from the account of this open account event:

- set the credit limit of this account to the limit of this open account event
 - set the customer of this account to the customer of this open account event.
-

In other cases, the requirement might be to derive the value of an entity based on some existing attributes.

Example 6-5 shows the initialization of an attribute based on some information from the same entity where the projected yearly balance attribute comes from current monthly balance and a computation.

Example 6-5 Initializing an entity attribute

the projected yearly balance of a customer is the current monthly balance * 12.

When entity initialization data must be retrieved from external sources, DSI also provides for this scenario. This is useful for those cases when data needed is out of your domain, so you must rely on additional service providers.

When using a data provider, returned attribute values are cached, which prevents calling the same provider for the same data multiple times. Example 6-6 shows an airport entity enriched by weather services, which takes two parameters: latitude and longitude.

Example 6-6 Entity enriched with a service provider

```
an airport is enriched by the weather provider,  
  given the latitude from the latitude of this airport  
  and the longitude from the longitude of this airport,  
  setting the temperature to the temperature of this weather provider,  
    the wind speed to the wind speed of this weather provider  
    and the precipitation level to the precipitation of this weather provider.
```

A data provider can return any number of values, as shown in Example 6-6 on page 99 where three values are returned to initialize the entity: temperature, wind speed, and precipitation level.

For additional design considerations for entity initialization and enrichment, see IBM Knowledge Center at the following web addresses:

► Entity initialization:

http://www.ibm.com/support/knowledgecenter/SSQP76_8.7.1/com.ibm.odm.itoa.develop/topics/con_entityinitialization.html?lang=en

► Entity enrichment:

http://www.ibm.com/support/knowledgecenter/SSQP76_8.7.1/com.ibm.odm.itoa.develop/topics/con_data_enrichment.html?lang=en

Event history

When DSI processes events, they can be configured to remain available in internal storage to become part of the event history. This is part of the context that can be used to identify situations, such as how many times a late payment event for specific customer has been received.

Example 6-7 shows how you can check the event history for past events (withdrawals) to use their count in a validation for an action.

Example 6-7 Action rule using events history (withdrawals) to identify past events

```
when a withdrawal occurs  
definitions  
  set 'the withdrawals history' to all withdrawals during the last period of 50  
  days ;  
if  
  there are at least 3 withdrawals in 'the withdrawals history'  
then  
  ...
```

Aggregates

The concept of aggregates for DSI means the system can automatically compute globally scoped values related to entities or events. This feature is useful for scenarios when you want to keep a counter or average related to a specific entity, event, or attribute. In a typical situation without DSI, every application is responsible for performing these computations based on the events that are sent or received. For example, you might want to capture the average amount of money spent by a group of customers with certain characteristics. DSI can compute these values for you automatically and make them available to you at decision time.

Computations that are provided by DSI for aggregates can have these characteristics:

- ▶ average: Computes the average of a numeric field from a collection of objects
- ▶ maximum: Computes the maximum value of a numeric field from a collection of objects
- ▶ minimum: Computes the minimum value of a numeric field from a collection of objects
- ▶ number of: Computes the number of elements from a collection of objects
- ▶ total: Computes the sum of a numeric field from a collection of objects

Aggregates in DSI are classified into two types: Entities and events. How these aggregates are computed depends on the type of aggregate.

Entity aggregates are computed at the scheduled period defined during the entity aggregate.

Example 6-8 shows an aggregate that computes the maximum number of ships in the system, computing the value every day at 12:01:00 AM. Notice the ability to set a default value of 5 when there are fewer than 2 ships in the system.

Example 6-8 Example of an entity aggregate

```
define 'my_entity_aggregate' as the maximum { the number of elements in all ships
} ,
    defaulting to 5 if there are less than 2 ships ,
    evaluated every day at 12:01:00 AM
```

Event aggregates are computed when events are received by the system.

Example 6-9 shows an event aggregate that computes the total number of train arrivals in the last two weeks with a default initializer.

Example 6-9 Example of an event aggregate

```
define 'total arrivals aggregate' as the number of train arrivals during the last
period of 2 weeks ,
    defaulting to 10 if there is less than 5 days of event history
```

For more information about aggregates, see the ODM IBM Knowledge Center at the following web address:

http://www.ibm.com/support/knowledgecenter/SSQP76_8.7.1/com.ibm.odm.itoa.overview/topics/con_aggregation.html

6.3.6 Using context

The power of DSI comes from its ability to detect situations in real time. To achieve this, you must perform some logic on the context that you have collected. This section describes some of the exciting things that you can do.

Geospatial

BERL offers the ability to reason over geometries as standard. This means, for example, that you can perform geofencing (Example 6-10).

Example 6-10 Example rule with geofencing

```
when an aircraft location update occurs
definitions
    set 'the aircraft' to the aircraft of this aircraft location update;
    set 'the airport' to the destination airport of this aircraft location update;
```

```

    set 'the airspace' to the airspace of 'the airport';
  if
    'the airspace' contains the location of this aircraft location update
  then
    emit a new air traffic notification where
      the airport is 'the airport',
      the aircraft is 'the aircraft';

```

Example 6-11 shows finding the nearest entity using geometries.

Example 6-11 Example rule finding the nearest entity

```

when an aircraft emergency occurs
definitions
  set 'the aircraft' to the aircraft of this aircraft location update;
  set 'the airports' to all airports where the emergency capacity of each airport
  is at least 1;
  set 'the nearest airport' to the nearest point among the locations of 'the
  airports' to the location of this aircraft emergency;
then
  emit a new aircraft emergency landing notification where
    the aircraft is 'the aircraft',
    the airport is 'the nearest airport';

```

For more information about geospatial reasoning, see the IBM Knowledge Center at the following web address:

http://www.ibm.com/support/knowledgecenter/SSQP76_8.7.1/com.ibm.odm.itoa.develop/topics/con_write_geospatial_rules.html

Temporal and event history

BERL can reason over time periods with a simple syntax. This means, for example, that you can see whether an event occurred during a specific period (Example 6-12).

Example 6-12 Example rule finding if an event occurred in a time period

```

when an expense occurs
definitions
  set 'the business trip duration' to the period between the start of 'the
  business trip' and the end of 'the business trip';
  set 'the new balance' to the expenditure of 'the business trip' + the amount of
  the expense;
if
  the time of the expense is during 'the business trip'
then
  set the expenditure of 'the business trip' to 'the new balance';

```

You can also find all events in a calendar period (Example 6-13).

Example 6-13 Example rule finding all events in a calendar period

```

definitions
  set 'current year expenses' to all expenses where the time of each expense is
  during the calendar year now;
if

```

```
the total amount of 'current year expenses' is more than 1000000
then
  print "It's been an expensive year!";
```

Example 6-14 shows determining whether time points are before or after each other.

Example 6-14 Example rule determining whether a time point is before another

```
when an expense occurs
definitions
  set 'the start' to the start of 'the business trip'
if
  the time of the expense is before 2 hours before 'the start'
then
  emit a new invalid expense alert where
    the expense is the expense,
    the reason is "Occurred too early";
```

You can also compare a time period with a calendar duration (Example 6-15).

Example 6-15 Example rule determining whether a time period is longer than a calendar duration

```
definitions
  set 'the business trip duration' to the period between the start of 'the
business trip' and the end of 'the business trip';
if
  'the business trip duration' is longer than 2 months
then
  emit a new employee trip home offer;
```

For more information about temporal reasoning, see the IBM Knowledge Center at the following web address:

http://www.ibm.com/support/knowledgecenter/SSQP76_8.7.1/com.ibm.odm.itoa.develop/topics/con_write_time_rules.html

Example 6-13 on page 101 also highlights another capability of rule agents: Performing aggregate calculations on sets of events. In this case, you sum the amounts of the expenses to obtain the total expense in this calendar year.

Another interesting interaction to note is to combine this geospatial and temporal reasoning as shown in Example 6-16. This example analyzes all GPS readings to determine whether a vehicle was at the fuel stop during the period when the fuel purchase was made.

Example 6-16 Example rule combining geospatial and temporal reasoning

```
when a fuel purchase occurs
definitions
  set 'the fuel stop' to the fuel stop of the fuel purchase;
  set 'the purchase time' to the time of the fuel purchase;
if
  there is no GPS reading where the distance between the location of this GPS
reading and the location of 'the fuel stop' in feet is less than 50 and the period
between the time of this GPS reading and 'the purchase time' is shorter than 5
minutes
```

```
then
    emit a potential fuel fraud notification;
```

6.3.7 Scheduling rule execution

You can use the concepts introduced for temporal reasoning in “Temporal and event history” on page 101 to schedule rule execution. You can schedule a rule to be started every day at a certain time (Example 6-17).

Example 6-17 Example rule scheduled to be started every day at 3:00 PM

```
if
    the time of day of now is 3:00:00 PM
    and it is true that the referee of ‘the soccer game’ is ready
then
    print “Kickoff!”;
```

In addition, you can set a rule to only be run between certain times (Example 6-18).

Example 6-18 Example rule to only be run during a specific time period

```
if
    the time of day of now is after 12:30:00 PM
    and the time of day of now is before 1:30:00 PM
    and it is not true that ‘the soccer player’ has eaten
then
    emit a new mobile alert where
        the recipient is ‘the soccer player’,
        the message is “Eat some lunch!”;
```

Example 6-19 shows running a rule on a specific day of the week.

Example 6-19 Example rule to be run on a specific day of the week

```
definitions
    set ‘the club to the soccer club of ‘the soccer player’;
    set ‘the training ground’ to the training ground of ‘the club’;
if
    now is on Sunday
    and the distance between the location of ‘the soccer player’ and the location
    of ‘the training ground’ in feet is less than 50
then
    emit a new mobile alert where
        the recipient is the head coach of ‘the club’,
        the message is the name of ‘the soccer player’ + “ is doing extra
        training.”;
```

6.3.8 Integration and connectivity

Given that DSI operates based on events received, clearly it is fundamental for some capabilities around connectivity to be present. This section provides descriptions of the necessary mechanisms for it to participate in the consumption and sending of events.

Consumption and sending of events are handled by dedicated inbound and outbound connectivity features.

Figure 6-23 illustrates the role of the connectivity features in a DSI solution.

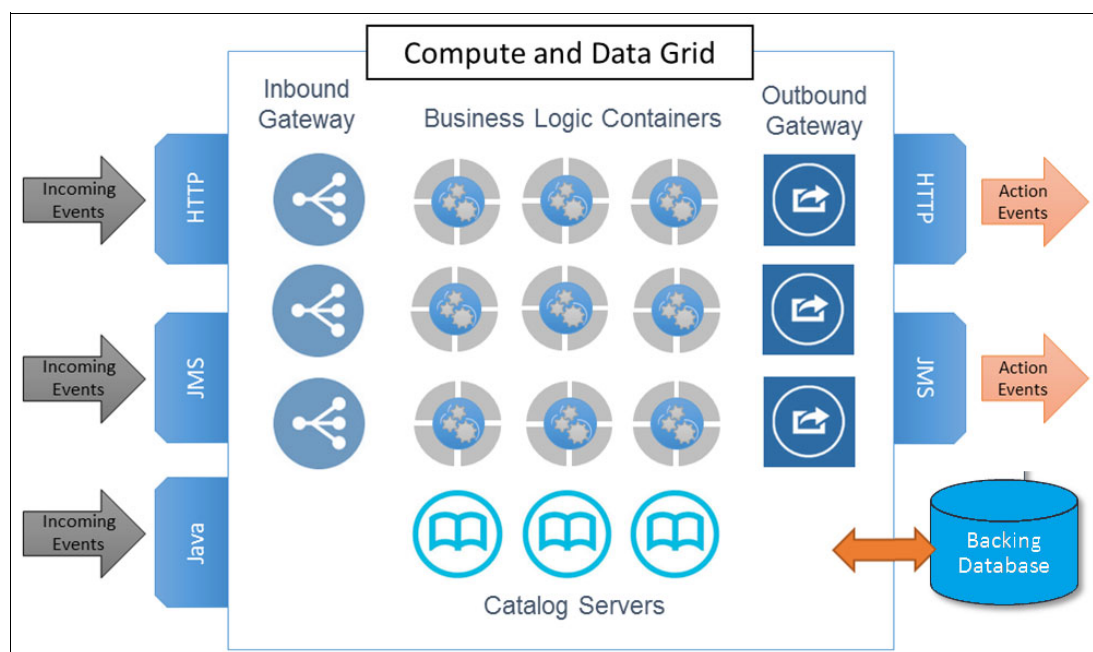


Figure 6-23 Decision Server Insights runtime components

Inbound connectivity

This component handles requests from source systems, submits inbound events to the compute grid, and handles initial filtering and routing of events according to the agent descriptors.

Outbound connectivity

Outbound connectivity is responsible for delivering outbound events to event destinations where the events trigger actions.

Message filtering

DSI is usually found at the higher levels of the event-hierarchy, which is to say that DSI processes “business events”. These are events that have a certain business meaning. Consumption of data in motion is detailed in Chapter 4, which presents the event hierarchy explained in Figure 4-2 on page 38. Therefore, it is a fair assumption that some basic form of filtering, enrichment, or transformation has been applied to the raw messages before these events reach DSI. Nevertheless, DSI provides some filtering and enrichment functions for further event enhancement.

The storage and retrieval of data within the data grid is often the most costly step in any solution (see 9.4.1, “Designing for performance” on page 220). Therefore, it is smart to filter unnecessary events before the events reach the containers. For example, because you know that only customers who are members of the airline loyalty program are eligible for a real-time promotions offer, a filter condition that ignores events related to customers who are not in the loyalty program can improve solution design and performance (Example 6-20)

Example 6-20 Agent descriptor including a filter condition

```
'promotions_agent' is an agent related to a customer ,
processing events :
- checkin event
  when the status of this checkin event is PROMOTED ,
  where this customer comes from the customer of this checkin event
```

Agent descriptor syntax details for “when” usage can be found at the following web address:

http://www.ibm.com/support/knowledgecenter/SSQP76_8.7.1/com.ibm.odm.itoa.ref/topics/ref_itoa_bdl_agents.html

By contrast, consider an alternative design that captures this logic within an action rule in a rule agent. In that case, every rule needs to contain an extra condition to only catch situations that are relevant. This makes the rules more complex and more difficult to understand. For this reason, and the previously mentioned storage considerations, filtering in agent descriptors is a much better idea if possible.

An example where this approach would not be valid is when the filtering logic is part of the business decision as opposed to part of the agent scope. For example, you might have a situation where a single rule agent has different rules that apply for Gold and Silver customers.

Message transformation

The connectivity features also allow you to specify a message transformation in an eXtensible Stylesheet Language (XSL) transformation file. This is useful when you cannot control the XML message schema in your event source or destination applications and you have defined a new event model in DSI with your BMD. Note that DSI does allow you to create BOM entries from an existing XSD where appropriate annotations are provided.

A guide to importing existing XML schemas is provided in the IBM Knowledge Center at the following web address:

http://www.ibm.com/support/knowledgecenter/SSQP76_8.7.1/com.ibm.odm.itoa.develop/modeling_topics/tsk_create_bom_from_xsd.html

Chapter 8 gives a real example of message transformation for outbound messages sent to an IBM Business Process Manager solution.

Protocol support

To take advantage of the connectivity features, event sources are required to submit events through either the HTTP or JMS protocols with an XML message payload. Likewise, event destinations must adhere to the same standards.

The configuration parameters that are required to define connectivity to HTTP and JMS endpoints are specified in a connectivity definition file (.cdef) within the Connectivity Definitions folder of your DSI solution project. The connectivity definition file is built using a natural language-like format that understands the event definitions from your BMD.

Example connectivity definitions are available in the IBM Knowledge Center at the following web address:

http://www.ibm.com/support/knowledgecenter/SSQP76_8.7.1/com.ibm.odm.itoa.develop/topics/tsk_define_inbound_conn.html

Alternatively, you can choose to connect through the event gateway Java API directly, without going through a connectivity layer. This is useful when your event source is a Java application to which you can add the connectivity code and when you have high event throughput requirements. However, you must manage grid connectivity, event filtering, and message transformation manually in your client application.

For situations where the source or destination system cannot adhere to the protocols of message formats listed here, an additional integration layer is required. Chapter 8 describes the use of DSI with a third-party mediation or integration layer.



Taking action

This chapter describes how to ensure that the observations and decisions that are made within a *system of insight* can be transformed into actions. *A system of insight provides no value if it does not enable action to be taken.*

Specifically, it describes common actions enabled by a system of insight, and how insight enables continuous improvement and action feedback.

Instructions for implementing some of the patterns described in this chapter are included in Chapter 8. The scenario provides a real example using IBM Operational Decision Manager (ODM) Decision Server Insights (DSI), IBM Business Process Manager, and IBM Integration Bus.

This chapter covers the following topics:

- ▶ Actionable insight
- ▶ Common actions
- ▶ Continuous improvement

7.1 Actionable insight

Triggering an action in a third-party system after the automatic decision to perform that action requires some systematic communication from the system of insight.

Consider the following actions:

- ▶ Add a job to a batch process running on a mainframe.
- ▶ Move an electronic actuator.
- ▶ Update a customer's loyalty status in five different systems of record.
- ▶ Assign a work item in an enterprise resource planning (ERP) system.
- ▶ Initiate a phone call through interactive voice response (IVR) between a service agent and a customer.
- ▶ Send mobile push notifications to customers that have your mobile application.

Although action is enabled by insight gained within the system of insight, action execution logically takes place outside the system, as shown in Figure 7-1. In some systems of insight, particularly those that are considered applications, the solution contains features that allow action execution within the application. In this case, although the capability is logically contained within a single solution, the tooling is usually a distinct component, and is applicable to the model shown in Figure 7-1. For example, the IBM Counter Fraud Management solution contains a Case Management component in which the actions take place.

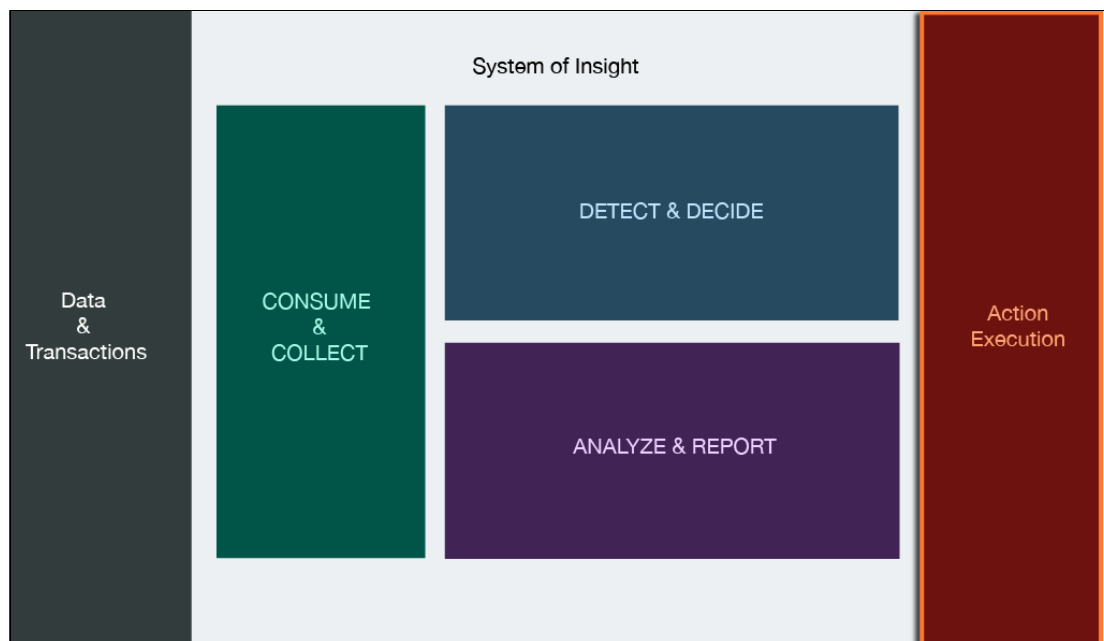


Figure 7-1 System of Insight model

Furthermore, in many scenarios, particularly where insight is requested from a source application, the action execution also takes place in the same source application. That is, in the model in Figure 7-1, the systems that are represented by *Data & Transactions* and *Action Execution* are the same.

In the scenarios listed in the beginning of this section, the system of insight will only adopt partial responsibility for the complexities of integration. Usually this responsibility is ensuring

that its outbound messages are delivered at least once. However, it is better to delegate orchestration and transformation where integration is complex or between heterogeneous systems.

For example, in the list, each of the actions require different message formats, transactionality, synchronous behaviors, using proprietary or standards-based protocols, and so on.

Often, the communication protocols or message formats supported by the system of insight are not compatible with the destination system. In these cases, a mediation layer is required to perform the transformation

Commonly, an enterprise service bus (ESB) is used to perform protocol and message format transformations. However, it is not unusual to also see the following technologies used:

- ▶ Third-party messaging, for example IBM MQ, especially in an event-driven architecture and when both the system of insight and destination systems provide support for this intermediate messaging.
- ▶ Message-driven beans (MDBs). Where a Java EE application server is available, custom Java code performs the required transformations.

Adopting an approach with an intermediate step can also improve the quality of service if implemented correctly. For example, assuring that messages are delivered exactly once, providing looser coupling, implementing a buffer to ease the effect of peaks in throughput, and gracefully handling failures.

7.2 Common actions

To begin, classify and describe actions that are commonly started by systems of insight:

- ▶ Asynchronous interactions
 - Notification
 - Process
 - Custom applications
 - Sensors, actuators, and devices
- ▶ Synchronous interactions
 - Contact center
 - Web application
 - Process

7.2.1 Asynchronous Interactions

These interaction patterns have an asynchronous nature, that is, actions described here are triggered in a fire-and-forget fashion by a system of insight when an interesting situation is detected. Typically, these actions occur in an event-driven architecture. However, their *asynchronous* nature does not imply that the action needs to be implemented with an asynchronous protocol.

Notifications

Providing a notification is regularly the most straightforward way to realize value from a detected situation or insight. In most cases, the appropriate response to a situation requires subjective input from an appropriate person to handle the output. However, in some

circumstances the system of insight can interact directly with a system of engagement (for example) which then performs an automatic set of steps.

In the following notification scenarios, the notification payload that is sent from the system of insight is typically small:

- For a push notification to a mobile device: A simple title, short message, and callback mechanism
- For system notification: Appropriate event classification and identifiers usually suffice

The size is determined by the context that allows the receiving person or system to perform the action, perhaps by first looking up some reference data given the context of the notification. This process has the advantage of reducing the message payload size in the network and keeping service interfaces compact. It is not common for event-driven systems to have messages with fully enriched data attributes. For more information about data size, see “Store only what you need” on page 221.

Interaction with employees and customers cover most notification scenarios. For example, employees being notified about breached key-performance indicators, or exceptions that need to be handled are common, as are sending warnings or information alerts to customers.

Customers

In a scenario where you send promotions to customers based on some insight about them, the system of insight decides what promotion to send and when, and emits an event. However, the notification service references the customer profile to determine their communication preferences (for example, email, push notification, or SMS), and message format.

Notifications to customers are common in systems of insight related to marketing, fraud detection, and next best offer use cases.

Employees

Scenarios relevant to employee notification are often related to improving operational effectiveness. For example, allowing employees to be more productive by alerting them to risk situations earlier.

For employees, the notification might be through a business application, intranet portal, or email. However, the system of insight is usually independent of the notification channel, with the precise route being decided by the operational notification service.

Systems

Few scenarios are expected to be completely automated. Notable exceptions are Internet of Things (IoT) use cases and industrial automation. However, where an action does lend itself naturally to IT automation, this is an obvious target for integration with a system of insight.

When an action cannot be fully automated, actions can be deferred for further human investigation. For example, in fraud detect solution, a low-risk detection might be completely automated, while higher risks situations are assigned for further investigation by a specialist. Similarly, in claims processing, certain low-value claims can be approved automatically and higher-value, more complex situations require handling by a knowledge worker.

Logging

Decisions that are made through a system of insight should also be logged. In addition to auditing, logging provides a data source for other systems that are looking for different (perhaps higher-level) insight. For more information, see 7.2.3, “Feedback” on page 113 and event column-value hierarchy in Figure 4-2 on page 38.

Process

This section uses the term *process* to refer to any set of business activities that when performed together achieve some wanted outcome. The steps can be in a predefined order or determined dynamically, and involve only people, only systems, or any combination of the two. Examples include case-handling for insurance claims, processing financial transactions, exception handling, or client onboarding.

The discipline of business process management (BPM) aims to improve business process through the application of a process modeling, automation, monitoring, and analysis. When a business process is implemented in a BPM platform, the system manages the orchestration of process activities. For example, it can provide workflow capabilities, coach users through the steps required to complete each activity, and integrate directly with systems of record.

However, the process categories described here also apply to business processes that exist within any IT systems, and also to those that are completely manual.

This section briefly describes interaction patterns for a business process that is started by an action from a system of insight.

Prescriptive resolution

This first type of process typically follows a predefined prescriptive set of activities that are required to accomplish a business function. Common examples are approval processes, review processes, and escalation processes that have a mixture of steps that are performed manually and steps that need the use of IT systems.

The resolution process is started directly as a result of some situation, insight, or decision that identified by a system of insight. Figure 7-2 shows a simple version of this process where the Business Process Model and Notation (BPMN) diagram has a Message Start Event followed by an activity. In reality, the process diagram is likely to have multiple process steps, process participants, decision points, and interactions.



Figure 7-2 Start a business process by an event trigger

The airline scenario used in Chapter 8 includes an example of a process to resolve the situation related to a passenger who is expected to miss their flight.

Build a case

Instead of starting a process that conforms to a pre-determined process flow, it is also common to start a case handling process as output from a system of insight. The case folder is built based on the context of a particular entity (perhaps a customer or transaction), from which a skilled knowledge worker decides the steps that are necessary to resolve the case.

Typically, this is an exception handling process (for example, open a case to handle a fraudulent scenario) and the resolution process has an investigative element.

Straight-through processing

In some use cases, the process might be completely automated and not require human interaction at all. Straight-through processing is used where multiple systems of record need to be updated, such as during the settlement of financial transactions.

Straight-through processing is achieved through the application of different orchestration technologies that include enterprise-service buses and business process management platforms.

This slide deck and associated article by Kim Clark describes the different integration patterns and gives guidance for their placement in a service-oriented architecture (SOA).

http://www.slideshare.net/kimjclark/s11-wtu2014-bpm-in-soa-runtime-component-placement-v16?tid=52f3fe32-c925-402a-8b72-772de824b162&v=default&b=&from_search=1

Context-driven process interaction

This pattern corresponds to use cases where an in-flight process requires correlation with a situation detected by a system of insight. The important distinction here is that the system of insight *knows* that a process has already been started and so can provide enough context to correlate with the running process. For example, the action might be to add further evidence to an existing case or to close a case because the appropriate situation for closure has been detected.

Figure 7-3 illustrates this pattern in BPMN. The activities in the diagram represent any generic activity or set of activities that are performed before and after the receipt of the intermediate event.



Figure 7-3 Wait for an event before continuing the process.

Often this scenario is better represented by an event-gateway as shown in Figure 7-4, where the path taken depends on the event received.

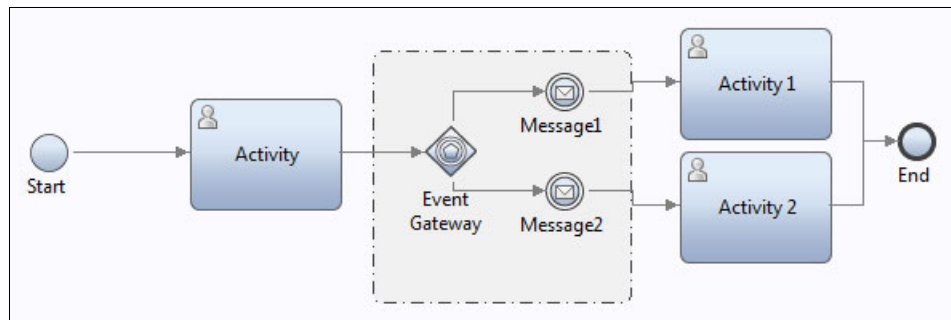


Figure 7-4 BPMN diagram of an Event Gateway

The airline scenario in 8.7.2, “Decision Server Insights outbound integration with IBM Business Process Manager (IBM BPM)” on page 186 uses BPM to implement manual case handling by an employee in the ticket office.

Business applications

Similar to scenarios where a process is run automatically, interaction with business applications only contain direct system interactions. Example use cases are updating a status in a CRM system, or placing an order through an ERP system.

Sensors, actuators, and devices

These scenarios cover use cases where a physical action occurs as a result of the derived action. This might be to close vents in an automated building control system through moving an actuator or to sound an alarm in an emergency,

7.2.2 Synchronous interactions

Systems that start a system of insight synchronously make a request that provides some additional contextual data and wait for a response. The response contains a decision or answer that is used by the source system.

The Decision Automation and Predictive scoring solution examples in Chapter 2 provide examples of this pattern.

Within the model of a system of insight (Chapter 3), these examples have the *Data and Transactions* component and *Action Execution* component within the same system.

The following sections describe common examples of systems that have synchronous interactions with systems of insight.

Contact center application

When a client is interacting with a call center, the call center application uses a system of insight to improve call handling. For example, based on the specified reason for the call and previous interaction history, the system of insight helps to diagnose the client's problem or makes recommendations to the call center agent. Recommendations are often for next best action or next best offer. The recommendation is displayed to the agent who then chooses how to respond.

Web application

In a web application with some dedicated window areas for marketing offers, the browsing history provides additional context that is used to decide which targeted advertisement to display.

Process

In an automated workflow, specific steps start a system of insight to request insight at an appropriate stage. Examples include fraud assessment, risk scoring, and next best action.

7.2.3 Feedback

Insight can be used to provide feedback back to the system of insight itself and provide integration systematically with operational systems.

Direct event feedback

Consider a scenario where you receive readings from pressure sensors in a water network. Individually, these pressure sensor readings might not be particularly interesting, but in a scenario where you detect several drops in pressure among closely located pressure sensors, this pattern of events might indicate a leak. The leak scenario will have a direct action (for example, send out an engineer), but is also interesting for a system of insight. A pattern of several incidents, say a power outage combined with several leaks, might indicate more serious damage to a service duct that requires a different action (for example, notify emergency services).

In these types of situation, the system of insight publishes events that it also consumes directly. The new event is sometimes referred to as a derived event or a synthetic event because it did not come from one of the configured data sources. In IBM DSI, this concept is transparent to authors of agents that process events. That is, DSI makes no distinction between events sourced from one of its inbound connectivity sources and events created within agents. It is the connectivity definitions and agent descriptors that define the routing of events.

This pattern is called *direct event feedback* (Figure 7-5).



Figure 7-5 *Direct event feedback*

Indirect event feedback

A more subtle feedback loop occurs when an action initiated by a system of insight causes side effects in some other system that are detected and used by the system of insight to make a future decision.

This behavior is called *indirect event feedback* (Figure 7-6).

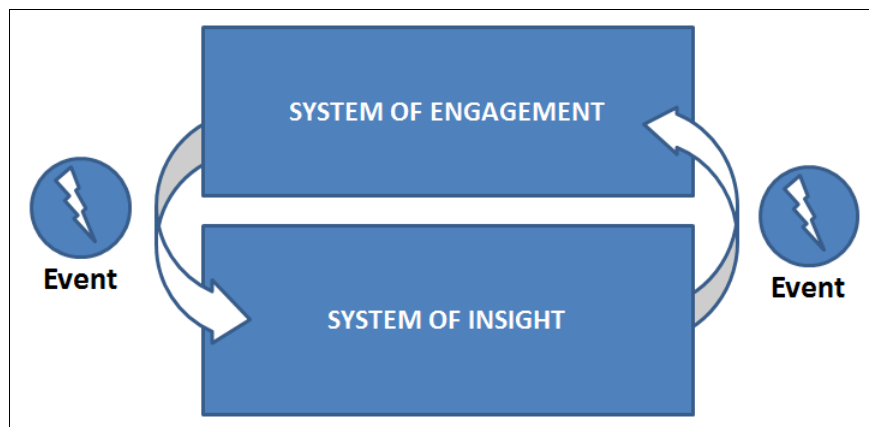


Figure 7-6 *In-direct event feedback*

Feedback from business processes

A common pattern of indirect event feedback occurs when a BPM solution is used with a system of insight. Many business processes are likely to be both event producers and consumers from the perspective of systems of insight. For example, a process that is started by an event received from a system of insight and also notifies the same system when it has completed.

Emit event at intermediate stage

When a significant milestone is reached that might be of interest to a system of insight, explicitly send an event. Figure 7-7 illustrates this pattern using BPMN. Note the black message notation compared to the white notation used in the previous examples.

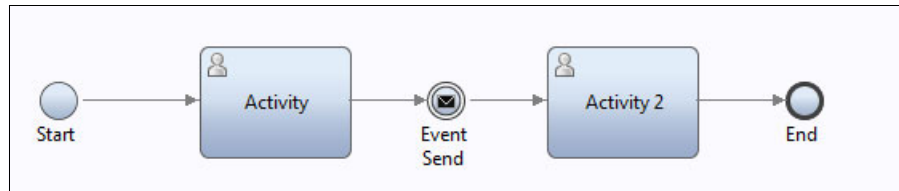


Figure 7-7 Emit an event at an intermediate process stage.

Emit an event on process completion

Similar to the previous pattern, event emission on process completion is shown in Figure 7-8. Note the thicker border of this message event notation compared to Figure 7-7.

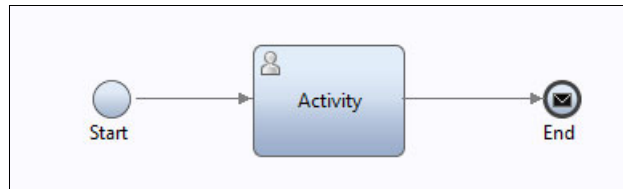


Figure 7-8 Emit an event when a process completes.

7.3 Continuous improvement

In the first instance, observations made within a system of insight can provide event data back into analytic data repositories, which enables continuous improvement. This can take the form of retraining predictive models, providing feedback to learning models, or expanding the operational data store or corpus.

Alternatively, for the operational systems and processes involved within or around a system of insight, there can exist opportunities to perform process improvement. Reporting and analysis might show bottlenecks in process implementation and efficiencies can be made in operating procedures and IT systems. Furthermore, the logic that automates the detection and decisions can be refined over time, perhaps by tweaking thresholds or expanding coverage to increase the number of cases that are handled automatically. Continuous improvement is a mature undertaking in many of these IT domains. However, systems of insight provide new drivers for change.



Integration examples with ODM Advanced

This chapter provides an example of an airline check-in application to demonstrate how a systems of insight solution helps an airline company to improve their customer satisfaction during the check-in process.

Airlines face multiple challenges every day moving passengers from their departure location to their destinations. Some challenges come directly from customers, whereas others are related to external factors such as flight delays, bottlenecks at security checkpoints, and so on. No matter where the challenges come from, the airline company wants to address the issues proactively, if possible, to keep the customer satisfied, and thus grow their business.

One common issue is missing a flight. This might be caused by a passenger running late or a delay in flight connection. If airline companies can learn ahead of time when a passenger has a high chance of missing their flight, the airline company can take action to speed up the check in process or, one step further, to reschedule the passenger, including their checked luggage, on another flight.

The fictitious scenario is used to demonstrate the concepts that were covered in previous chapters.

This chapter covers the following topics:

- ▶ Scenario overview
- ▶ Architecture
- ▶ Solution setup
- ▶ Consume and collect example
- ▶ Analyze and report example
- ▶ Detect and decide example
- ▶ Taking action example

8.1 Scenario overview

The scenario selected for this book focuses on the airline check-in process, and the ability to predict whether a passenger or group (for example, a family) will be able to board their flight on time. Figure 8-1 illustrates this scenario, the process starting at the top left section of the figure. Customers can check in to their flights in multiple ways such as mobile phone, kiosk, or bag drop desk. At the moment of checking in, the system uses information about the security check point queues to determine whether the customer has enough time to get to the departure gate or not. The cloud at the center of Figure 8-1 represents all the events being captured in the system from different sources. At the same time, events exit the system as actions to alert passengers, or trigger more business processes, for example, to reschedule a booking.

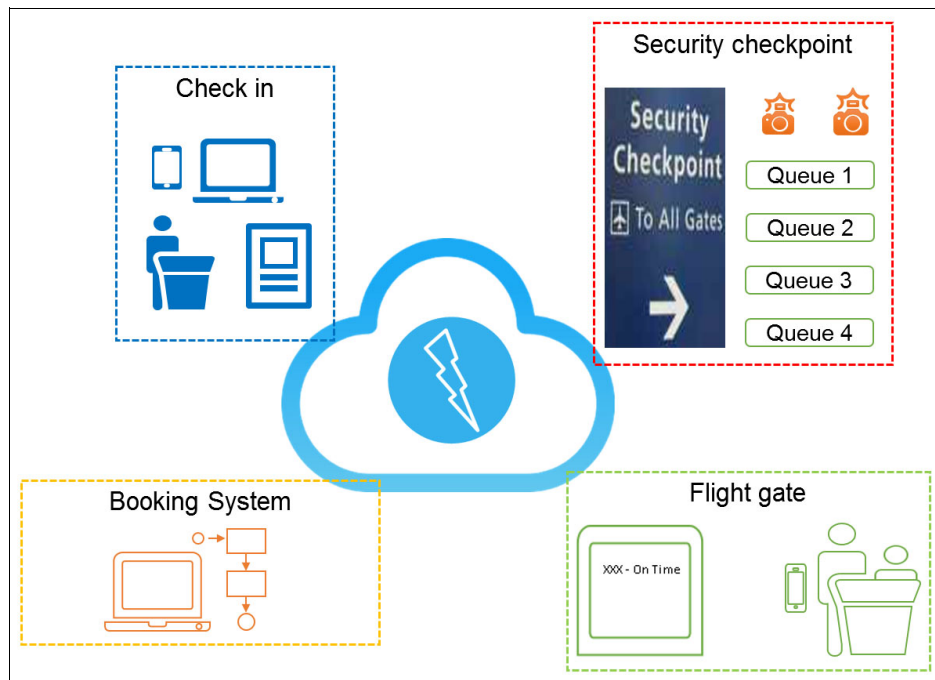


Figure 8-1 Scenario implementation overview

This scenario considers a passenger or group of passengers who travel from City A to City B on a specific date. These passengers have a reservation and have the following check-in options:

- ▶ Check in at the ticket counter at the airport.
You are interested in this situation because the passenger is already at the airport and you want to determine whether the passenger will be able to reach the gate in time to board the plane. If you determine the passenger will be late for the flight, you want to take actions to properly advise the passenger.
- ▶ Check in using a mobile phone when physically at the airport.
You are interested in this situation for the same reasons as the check-in at the ticket counter option described in the first bullet point.
- ▶ Check in at the bag drop counter when at the airport.
You are interested in this situation for the same reasons as the check-in at the ticket counter option described in the first bullet point.

- Online check-in 24 hours before the flight departure time.

The scenario implementation only validates interactions with passengers who are physically at the airport and close to their flight departure time. Therefore, in this scenario this case is ignored.

After the check-in process has taken place, the system validates whether the passenger or group of passengers in the same booking will be able to get to the departure gate in time. The following factors are considered, among others:

- Number of people crossing security check points. This scenario considers a list of queues available per security check point and the number of people in those queues.
- The age of the passengers because families with children often take longer than a single adult.
- Special needs of the passengers, such as disabilities, that might affect access to certain areas or affect the time needed to get to the gate.
- Location where the check-in was performed to determine the distance, and therefore the travel time, to the gate.

Based on available information from check-in process, the system predicts the time that is required to get to the gate, which is then used to determine recommendations to the customer.

When determining the customer recommendations, loyalty is a heavily-weighted consideration. A loyal customer can receive a special offer from the airline company. This in turn an effective incentive for customers to keep using the airline company for their future travel.

8.2 Architecture

A system of insight typically consists of various pieces of software for collecting and filtering events, analyzing information, detecting and deciding actions required, and performing these actions. In this scenario, the following products are used:

- IBM Business Process Manager (BPM)
- IBM InfoSphere Streams (Streams)
- IBM Integration Bus
- IBM Operational Decision Manager Advanced
 - Decision Server Rules (DSR)
 - Decision Server Insights (DSI)
- IBM SPSS Modeler (SPSS Modeler)
- IBM SPSS Collaboration and Deployment Services

Figure 8-2 shows the interaction between products that are used for the scenario implementation.

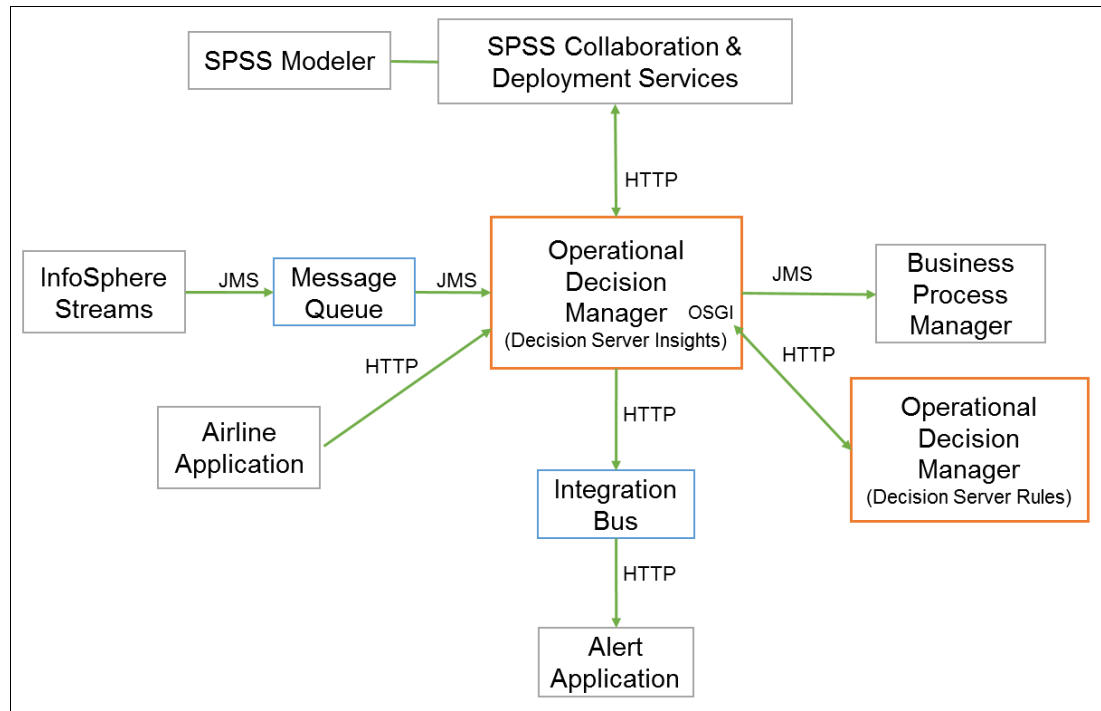


Figure 8-2 Scenario implementation architecture

In this scenario, DSI is used to manage the events that come from two main sources:

- ▶ **InfoSphere Streams:** Monitors the video streams from the security check point queues and sends derived events with information about the depth of these queues to an IBM MQ queue.
- ▶ **Airline application:** Sends events related to the passenger check-in and bag drop at the ticket counter.

The system needs to predict the length of time that a passenger requires to get to their gate. For this prediction, a call to SPSS Scoring service is used, with parameters of queue depths, type of passenger, and current passenger location. The scoring service then uses a predictive model to determine the length of time that is required to get to the gate in the current conditions.

DSI compares time to gate with the flight departure time. When passengers will be late, it requests a decision from DSR to recommend an action based on the passenger's loyalty status.

Based on this decision, DSI takes actions through different channels:

- ▶ Trigger a reschedule process through IBM BPM (automated or manual).
- ▶ Send notifications to an alerting application with JMS messages through IBM Integration Bus.

Note: In a real scenario, it is common to have all communication flow through the enterprise service bus (ESB, IBM Integration Bus). For example, in Figure 8-2 on page 120, the connection between the Airline Application and Business Process Management would be linked to IBM Integration Bus instead of directly communicating with DSI.

The underlying concepts for the scenario implementation are described in the following chapters:

- Chapter 4

Implementation of data collection by using InfoSphere Streams to send data to a message queue from where the DSI receives the incoming events.

- Chapter 6

Implementation of an application in DSI to start a DSR application to decide the next best action.

Implementation of a predictive analytics agent in DSI to retrieve scoring information from SPSS.

- Chapter 7

Implementation of all connectivity artifacts that are required to trigger both a manual and an automated process from IBM BPM, and an implementation of an Integration Bus flow to trigger alerts.

8.3 Solution setup

This section provides the initial steps required to set up the basic artifacts for the DSI solution project and references used in the example scenario. Details about DSI components and concepts are available in 6.3, “Situation-driven decisions in ODM Decision Server Insights” on page 88.

Use Insight Designer to perform tasks that are defined in this section.

8.3.1 Solution definition

Complete the following steps to create a solution definition:

1. Create a Solution project using Rule Designer (**File** → **New** → **Solution Project**) as shown in Figure 8-3.

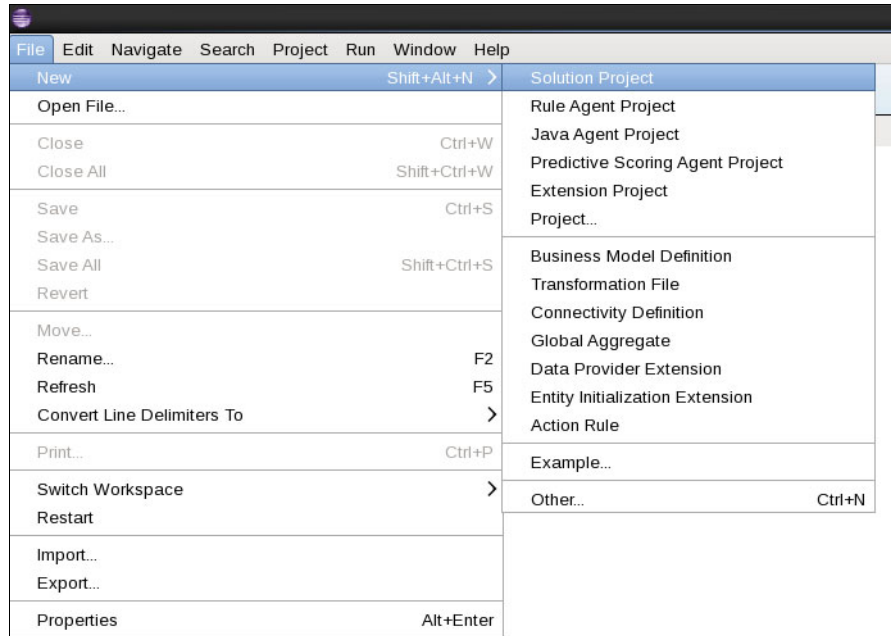
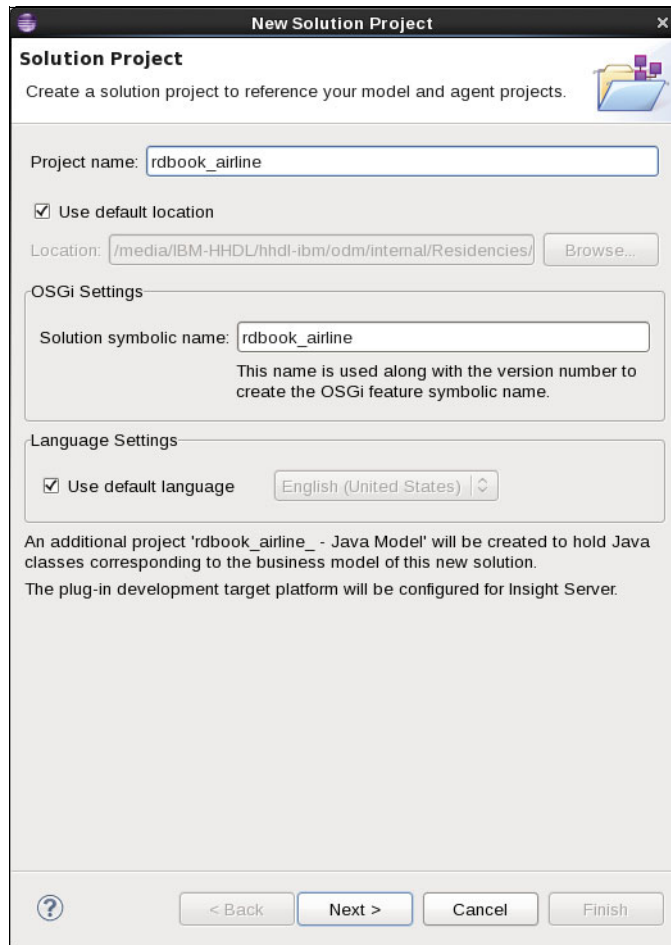


Figure 8-3 Creating a solution project

Important: If *Solution Project* is not available, make sure that your workspace is using the *Decision Insight* perspective.

2. Enter **rdbook_airline** as the project name and the solution symbolic name, then click **Next** (Figure 8-4).



The image shows a 'New Solution Project' dialog box with the following fields and options:

- Project name:** A text field containing 'rdbook_airline'.
- Use default location:** A checked checkbox.
- Location:** A text field containing '/media/IBM-HHDL/hhdl-ibm/odm/internal/Residencies/' and a 'Browse...' button.
- OSGi Settings:**
 - Solution symbolic name:** A text field containing 'rdbook_airline'.
 - A note: 'This name is used along with the version number to create the OSGi feature symbolic name.'
- Language Settings:**
 - Use default language:** A checked checkbox.
 - A dropdown menu showing 'English (United States)'.

At the bottom, there is a paragraph: 'An additional project 'rdbook_airline_ - Java Model' will be created to hold Java classes corresponding to the business model of this new solution. The plug-in development target platform will be configured for Insight Server.'

The bottom of the dialog features a help icon, and four buttons: '< Back', 'Next >', 'Cancel', and 'Finish'.

Figure 8-4 New solution project settings

3. On the BOM Project settings, make sure the option **Create empty BOM project** is selected as shown in Figure 8-5. Set the project name to **rdbook_arline_bom** and click **Finish**.

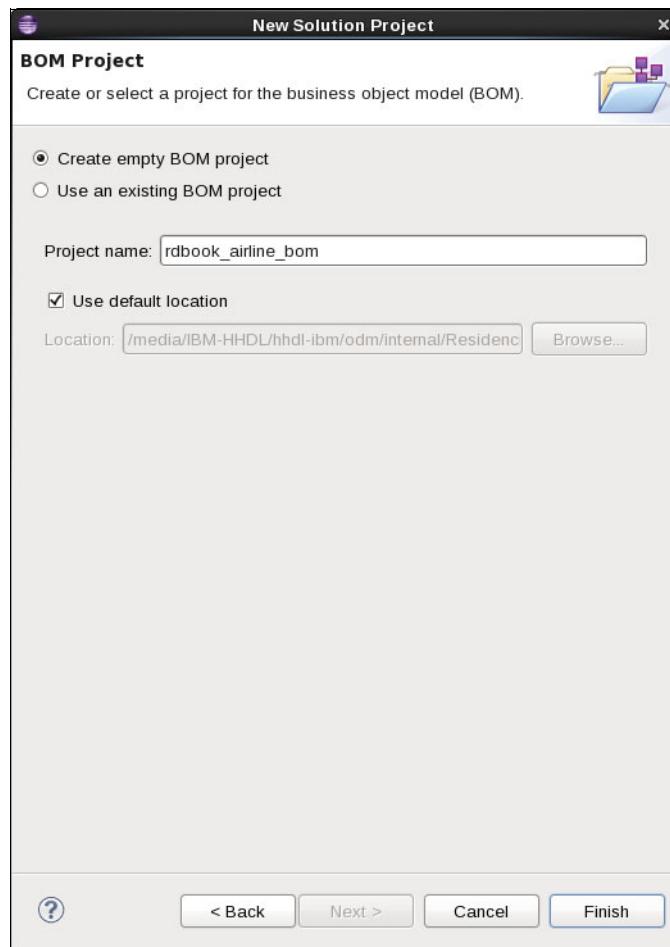


Figure 8-5 New BOM project settings

As a result, two projects are visible in the Solution Explorer view as shown in Figure 8-6.

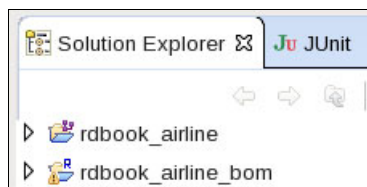


Figure 8-6 Projects created

8.3.2 Model definition

After a solution project is created, the next step is to work on the business model definition (BMD).

Complete the following steps to create a new BMD file:

1. Go to the Decision Insight perspective.
2. From the Solution Map view, select the **Define events and entities** link in the Model section as shown in Figure 8-7.

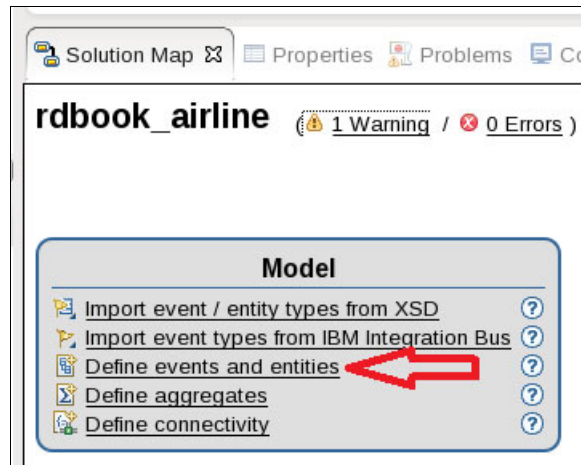


Figure 8-7 Define events and entities option in Solution Map

3. On the first window of the wizard, set the values as shown in Figure 8-8. Specify **rdbook_airline_bom** as the BOM project, **com.ibm.rdbook.airline** as the package name, and **entities** for the model definition.

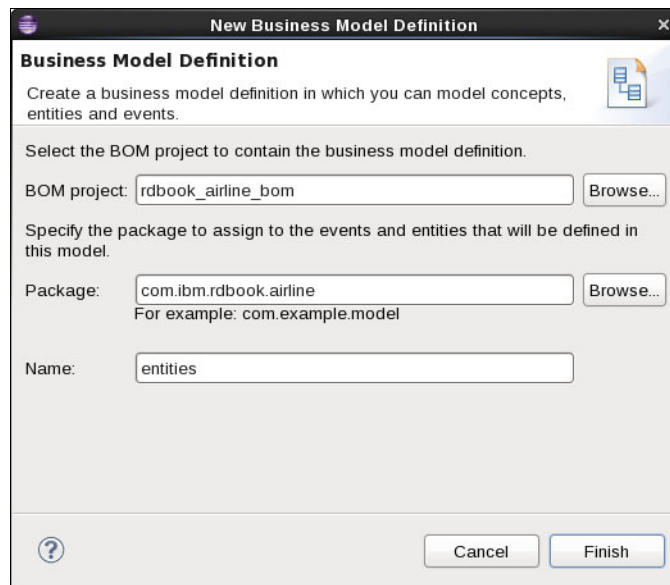


Figure 8-8 New business model definition settings for entities

4. Input the business model definitions to the entities.bmd file as shown in Example 8-1.

Example 8-1 Business model definition for entities

a booking is a business entity identified by a booking id.
a booking is related to a flight.
a booking is related to a passenger.
a booking has some exceptions.

an exception is a concept.
an exception has an action code.

a passenger is a business entity identified by a passenger id.
a passenger has a location (a point).
a passenger has a loyalty tier.
a passenger has an age.

an age can be one of: CHILD, ADULT, SENIOR.

a flight is a business entity identified by a flight id.
a flight has a gate .
a flight has a departure time (a time).

a gate is a concept .
a gate has a gate number .
a gate has a location (a point) used as the default geometry.

a desk is a business entity identified by a desk id.
a desk has a location (a point) used as the default geometry.
a desk has a queue depth (integer , 0 by default).
a desk has a zone.

a check in desk is a desk.
a security desk is a desk.
a gate desk is a desk.
a bag drop desk is a desk.

Note: Note that after pasting the content in the business model definition editor, it will be automatically parsed and validated to make sure that it complies with the correct structure. Different font colors will highlight entities, business operators, and attributes to clarify structure and syntax.

5. Repeat the process to create the events definition. From the Solution Map view, select the **Define events and entities** link in the Model section as shown in Figure 8-7 on page 125.

- Set the values as shown in Figure 8-9. Specify **rdbook_airline_bom** as the BOM project, **com.ibm.rdbook.airline** as the package name, and **events** for the model definition.

New Business Model Definition

Business Model Definition

Create a business model definition in which you can model concepts, entities and events.

Select the BOM project to contain the business model definition.

BOM project:

Specify the package to assign to the events and entities that will be defined in this model.

Package:

For example: com.example.model

Name:

Figure 8-9 New business model definition settings for events

- Input the business model definition for the events.bmd file similar to Example 8-2.

Example 8-2 Business model definition for events

```
a queue status is a business event time-stamped by a timestamp.
a queue status is related to a desk.
a queue status has a queue depth (integer).

a desk interaction is a business event time-stamped by a timestamp.
a desk interaction is related to a desk.
a desk interaction is related to a booking.

a late passenger is a business event time-stamped by a timestamp.
a late passenger is related to a booking.
a late passenger has a lateness (a time period).

a created exception is a business event time-stamped by a timestamp.
a created exception is related to a booking.
a created exception has an action code.

a notification is a business event time-stamped by a timestamp.
a notification has a message content.

a mobile alert is a notification.
a mobile alert is related to a booking.

a process invocation is a business event time-stamped by a timestamp.
a reschedule process invocation is a process invocation.
a reschedule process invocation is related to a booking.
```

a customer contact invocation is a process invocation.
a customer contact invocation is related to a booking.

Important note: When the model definition file is saved, a Java project is automatically created that includes all the Java Interfaces defined in the .bmd files. At the same time, a Business Object Mode is automatically created or updated to reflect changes in the model definition. Figure 8-10 shows the structure you should have after the Java Model project has been created and the Business Object Model updated. Additional information about the Java model is available in the IBM Knowledge Center at:

http://www.ibm.com/support/knowledgecenter/SSQP76_8.7.1/com.ibm.odm.itoa.develop/modeling_topics/con_java_model_project.html

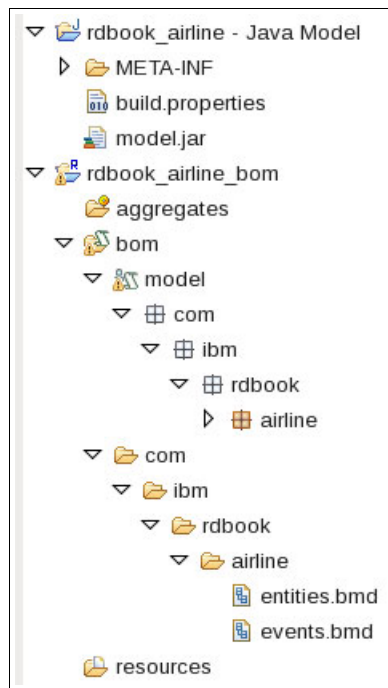


Figure 8-10 Solution explorer view with solution and bom projects.

8.3.3 Agents definition

Two agents are defined for this scenario: The booking agent and the desk agent. For more information about how to define agents, see 6.3.3, “Agents” on page 94.

Booking agent

The booking agent is a rule agent that processes exceptions as part of the check-in process. To create this agent, use the following steps:

1. Click **File** → **New** → **Rule Agent Project** and name the rule agent **booking_agent**.

Complete the agent descriptor with the contents of Example 8-3.

Example 8-3 Agent.adsc for the booking_agent

```
'booking_agent' is an agent related to a booking ,
processing events :
- created exception, where this booking comes from the booking of this created exception
```

2. To create the action rule to process exceptions, right-click the rules folder under the booking_agent project and click **New** → **Action Rule**. Place the rule in the com.ibm.rdbooks.airline.agent package and call it **process exception**.

Complete it with the contents of Example 8-4.

Example 8-4 process exception rule in booking_agent

```
when a created exception occurs
definitions
    set 'the action code' to the action code of this created exception;
then
    add a new exception where the action code is 'the action code' to the exceptions of
    'the booking';
    print 'the action code';
```

Note: For simplicity, in this example the Booking agent was created as a rule agent with a rule but no business logic associated with it. Lacking reference to past events makes this agent a clear Java agent candidate.

Desk agent

The desk agent is a rule agent responsible for updating queue depth based on queue status events that are received from the InfoSphere Streams application.

Create the desk agent with the following steps:

1. Click **File** → **New** → **Rule Agent Project** and name the rule agent **desk_agent**.

Complete the agent descriptor with the contents of Example 8-5.

Example 8-5 Agent.adsc for the desk_agent

```
'desk_agent' is an agent related to a desk ,
processing events :
- queue status , where this desk comes from the desk of this queue status
```

2. To create the action rule, right-click the rules folder under the booking_agent project and click **New** → **Action Rule**. Place the rule in the com.ibm.rdbooks.airline.agent package and call it **update queue depth**.

Complete it with the contents of Example 8-6.

Example 8-6 Updated queue depth rule in desk_agent

```
when a queue status occurs , called 'queue status'
then
    set the queue depth of 'the desk' to the queue depth of 'queue status' ;
```

8.4 Consume and collect example

This section demonstrates the data collection from video capture devices that are at every security checkpoint in the airport. These devices monitor the queues at these checkpoints.

The example uses the InfoSphere Streams technology to capture this unstructured data and convert it into a set of meaningful events. These events contain the depth of each queue, which the DSI component of ODM can receive and use in its context building for decision making.

InfoSphere Streams places events in a message queue from which DSI consumes them. The following steps guide you through the implementation process of this portion of the scenario.

8.4.1 Message queue configuration

The first task is to create a message queue in IBM MQ with IBM WebSphere MQ Explorer:

1. Start the **New Local Queue** wizard to create a local queue as shown in Figure 8-11.

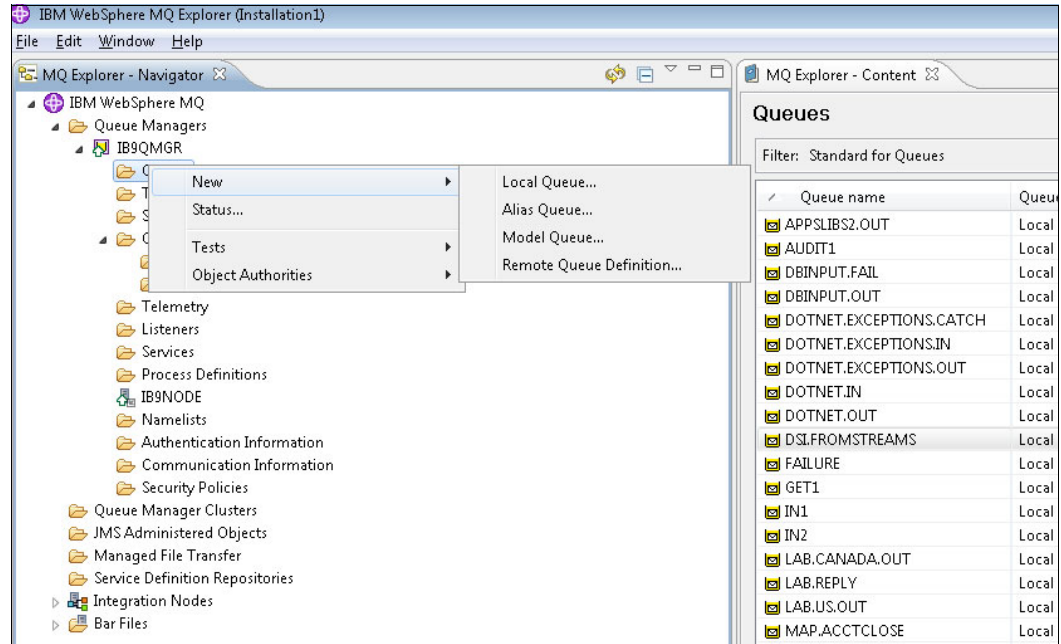


Figure 8-11 WebSphere MQ Explorer - Create a new queue

2. Specify the name of the local queue as **DSI.FROMSTREAMS** and leave all the other options unchanged (Figure 8-12).

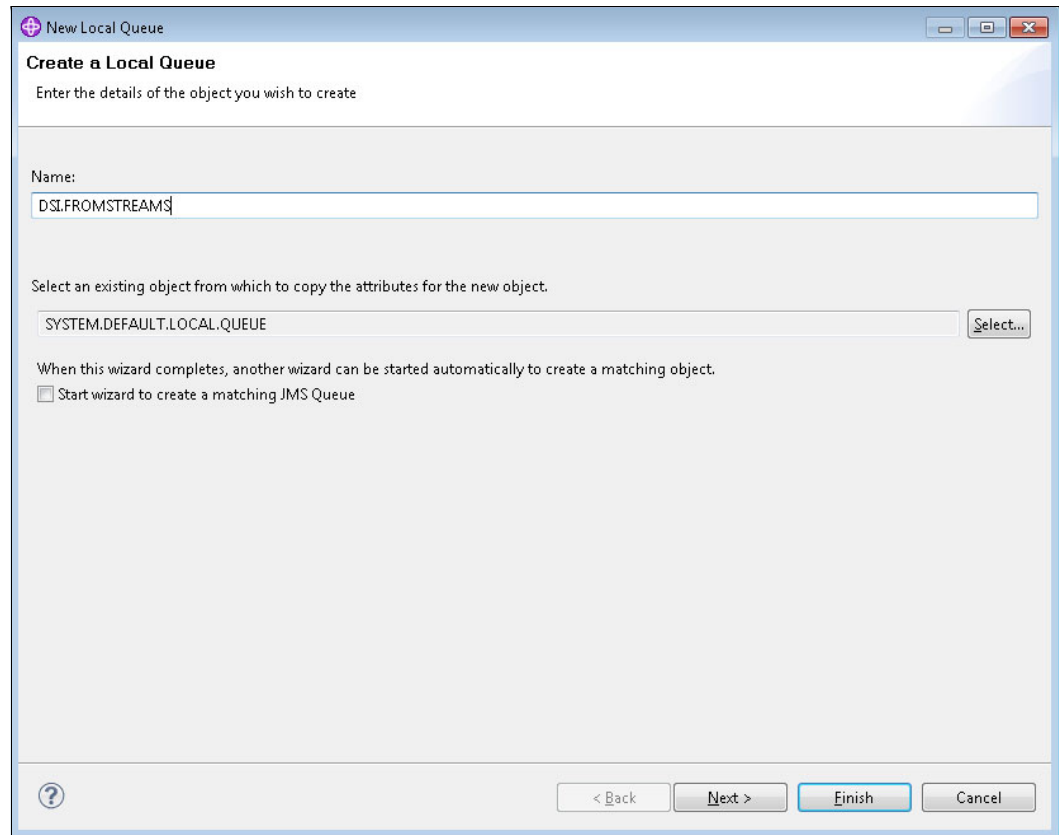
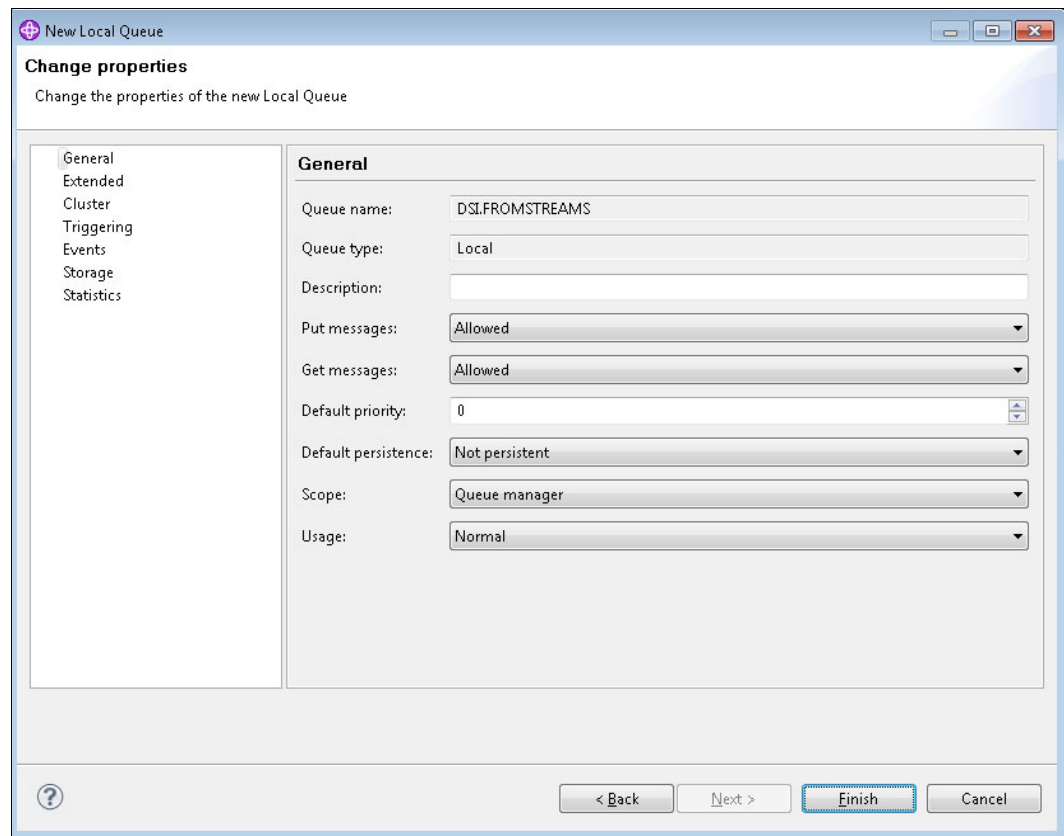


Figure 8-12 WebSphere MQ Explorer - Set name for new queue

3. On the Change properties window, leave all default values and click **Finish** (Figure 8-13).



The image shows a screenshot of the 'New Local Queue' dialog box in WebSphere MQ Explorer. The dialog has a title bar that says 'New Local Queue'. Below the title bar, there is a section titled 'Change properties' with the subtitle 'Change the properties of the new Local Queue'. On the left side, there is a tree view with the following items: General (selected), Extended, Cluster, Triggering, Events, Storage, and Statistics. The main area of the dialog is titled 'General' and contains the following fields and controls:

- Queue name: DSL.FROMSTREAMS
- Queue type: Local
- Description: (empty text box)
- Put messages: Allowed (dropdown menu)
- Get messages: Allowed (dropdown menu)
- Default priority: 0 (spin box)
- Default persistence: Not persistent (dropdown menu)
- Scope: Queue manager (dropdown menu)
- Usage: Normal (dropdown menu)

At the bottom of the dialog, there are four buttons: '< Back', 'Next >', 'Finish' (highlighted with a blue border), and 'Cancel'. There is also a help icon (?) on the bottom left.

Figure 8-13 WebSphere MQ Explorer - New queue properties definition

- After the local queue is created, it appears in the WebSphere MQ Explorer - Content view as shown in Figure 8-14.

InfoSphere Streams publishes messages into the DSI.FROMSTREAMS queue and from there, DSI consumes the messages.

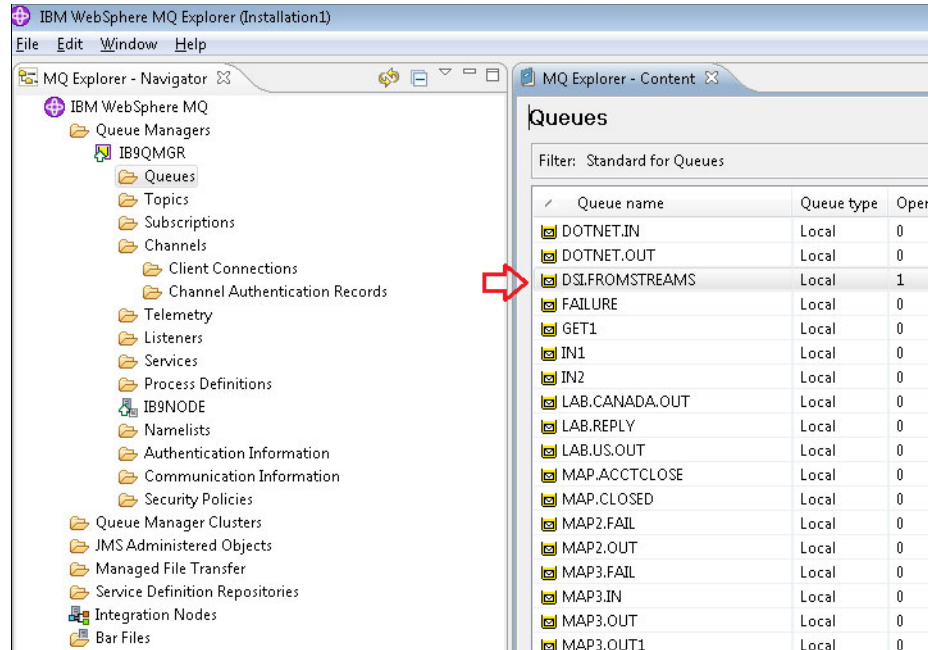


Figure 8-14 WebSphere MQ Explorer - Content view - New queue created

8.4.2 InfoSphere Streams configuration

To send events to the queue you created from InfoSphere Streams, use the JMS operator that is provided by the messaging toolkit. This section shows the steps needed to configure InfoSphere Streams so that the derived data from the security checkpoint can be put into a message and sent to the queue.

Prerequisites:

- ▶ The WebSphere MQ Client Library must be installed on InfoSphere Streams host.
- ▶ The InfoSphere Streams messaging toolkit must be added to the Streams workspace.

Complete the following steps to configure InfoSphere Streams:

- Create a JNDI script:

In this scenario implementation, InfoSphere Streams is hosted on a different machine from message queue. You need to register JNDI locally in the file system so that the queue can be looked up by the InfoSphere Streams JMS operator. Create a script **jmsJNDIClient.scf** as shown in Example 8-7.

Example 8-7 jmsJNDIClient.csp script

```
#-----
# Connection Factory for Client mode
# Delete the Connection Factory if it exists
DELETE CF(IB9CF)

# Define the Connection Factory
```

```

DEFINE CF(IB9CF) +
    SYNCPOINTALLGETS(YES) +
    TRAN(client) +
    HOST(9.12.7.22) CHAN(SYSTEM.DEF.SVRCONN) PORT(2414) +
    QMGR(IB9QMGR)

# Display the resulting definition
DISPLAY CF(IB9CF)

#-----
# Queue Object
# Delete the Queue if it exists
DELETE Q(DSI.FROMSTREAMS)

# Define the Queue
DEFINE Q(DSI.FROMSTREAMS) QUEUE(DSI.FROMSTREAMS)

# Display the resulting Queue definition
DISPLAY Q(DSI.FROMSTREAMS)

```

2. Create a script **JMSAdmin.sh** to run JMSAdmin as shown in Example 8-8.

Example 8-8 JMSAdmin.sh script to run JMSAdmin

```

java -DMQJMS_LOG_DIR=$MQ_JAVA_DATA_PATH\log -DMQJMS_TRACE_DIR=$MQ_JAVA_DATA_PATH\errors
-DMQJMS_INSTALL_PATH=$MQ_JAVA_INSTALL_PATH com.ibm.mq.jms.admin.JMSAdmin %1 %2 %3 %4 %5

```

3. Register JNDI objects for the connection factory and the queue by running the command in Example 8-9.

Example 8-9 Example of JNDI objects registration command

```

JMSAdmin.sh < jmsJNDIClient.scf

```

4. Create an SPL Composite **QueueMonitor** as shown in Example 8-10.

Example 8-10 SPL Composite QueueMonitor code

```

namespace acme.streaming;

use com.ibm.streams.messaging.jms::* ;

composite QueueMonitor
{
    type
        QueueStatus = tuple<rstring desk, rstring queueDepth, rstring ts> ;
    graph

        stream<QueueStatus> QueueData = FileSource()
        {
            param
                file : "queue_data.csv";
                format : csv;
        }

        // A stream of queue info from queue desk sensors
        // For illustrating the scenario, the stream data is from a file
        (stream<QueueStatus> MatchedQueueStatus) = Filter(QueueData)

```

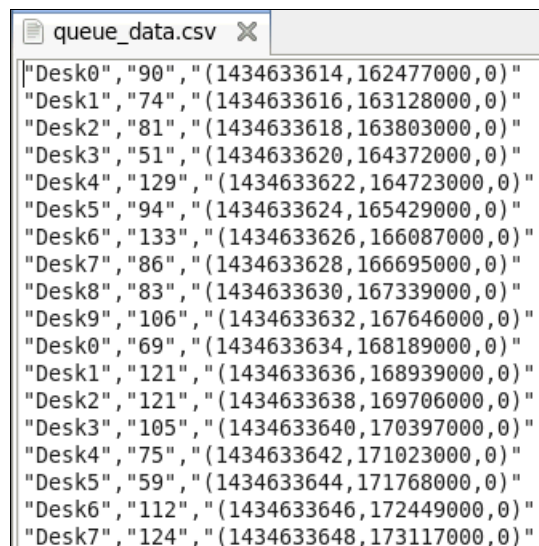
```

{
    param
        // Send queue status only when the queue depth is more than 80
        filter :((int32) queueDepth) >= 80 ;
    }
    () as MatchedQueueStatusScreenWriter = Custom(MatchedQueueStatus)
    {
        logic
            onTuple MatchedQueueStatus : printString("Matched queue status->"
                +(rstring) MatchedQueueStatus + "\n") ;
        }
        // Check the filtered queue data in file
        () as MatchedQueueStatusFileSink = FileSink(MatchedQueueStatus)
        {
            param
                file : "../data/filtered_queue_data.csv" ;
        }

        // Send queue status to IBM Integration Bus, to be consumed by ODM Decision
        Server Insight
        () as mySink = JMSSink(MatchedQueueStatus)
        {
            param
                connection : "IB9Conn" ;
                access : "QUEUESTATUS_ACCESS" ;
        }
    }
}

```

In this example, you stream the data from a file that contains queue status information that was captured every minute. The content of the file `queue_data.csv` is shown in Figure 8-15. Each row consists of data for the desk number, queue length, and the time stamp when the data is captured.



```

queue_data.csv
"Desk0","90","(1434633614,162477000,0)"
"Desk1","74","(1434633616,163128000,0)"
"Desk2","81","(1434633618,163803000,0)"
"Desk3","51","(1434633620,164372000,0)"
"Desk4","129","(1434633622,164723000,0)"
"Desk5","94","(1434633624,165429000,0)"
"Desk6","133","(1434633626,166087000,0)"
"Desk7","86","(1434633628,166695000,0)"
"Desk8","83","(1434633630,167339000,0)"
"Desk9","106","(1434633632,167646000,0)"
"Desk0","69","(1434633634,168189000,0)"
"Desk1","121","(1434633636,168939000,0)"
"Desk2","121","(1434633638,169706000,0)"
"Desk3","105","(1434633640,170397000,0)"
"Desk4","75","(1434633642,171023000,0)"
"Desk5","59","(1434633644,171768000,0)"
"Desk6","112","(1434633646,172449000,0)"
"Desk7","124","(1434633648,173117000,0)"

```

Figure 8-15 Sample content

A queue status is sent to the message queue only when the line at the security check point is past a certain depth. This is accomplished by a filter as shown in Example 8-11.

Example 8-11 Code example to validate when queue status is too long

```
(stream<QueueStatus> MatchedQueueStatus) = Filter(QueueData)
{
    param
        // Send queue status only when the queue depth is more than 80
        filter :((int32) queueDepth) >= 80 ;
}
```

The code that sends the queue status to the message queue is shown in Example 8-12.

Example 8-12 Code to send the queue status to IBM Integration Bus

```
() as mySink = JMSSink(MatchedQueueStatus)
{
    param
        connection : "IB9Conn" ;
        access : "QUEUESTATUS_ACCESS" ;
}
```

5. The JMS connection settings can be found in
<Sample-Project-Folder>/etc/connections.xml as shown in Example 8-13.

Example 8-13 Connection settings details from file: connections.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<st:connections xmlns:st="http://www.ibm.com/xmlns/prod/streams/adapters"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">

    <connection_specifications>
        <connection_specification name="IB9Conn">
            <JMS user="Administrator" password="passw0rd"
                initial_context="com.sun.jndi.fscontext.RefFSContextFactory"
                provider_url = "file:///opt/mqm/jndi"
                connection_factory="IB9CF"/>
        </connection_specification>
    </connection_specifications>

    <access_specifications>
        <access_specification name="QUEUESTATUS_ACCESS">
            <destination message_class="xml" delivery_mode="persistent"
                identifier="DSI.FROMSTREAMS"/>
            <uses_connection connection="IB9Conn"/>
            <native_schema>
                <attribute name="desk" length="15" type="String"/>
                <attribute name="queueDepth" length="15" type="String"/>
                <attribute name="ts" length="15" type="String"/>
            </native_schema>
        </access_specification>
    </access_specifications>
</st:connections>
```

- Opening the **QueueMonitor** with graphical SPL editor shows the stream processing graphically, as in Figure 8-16.

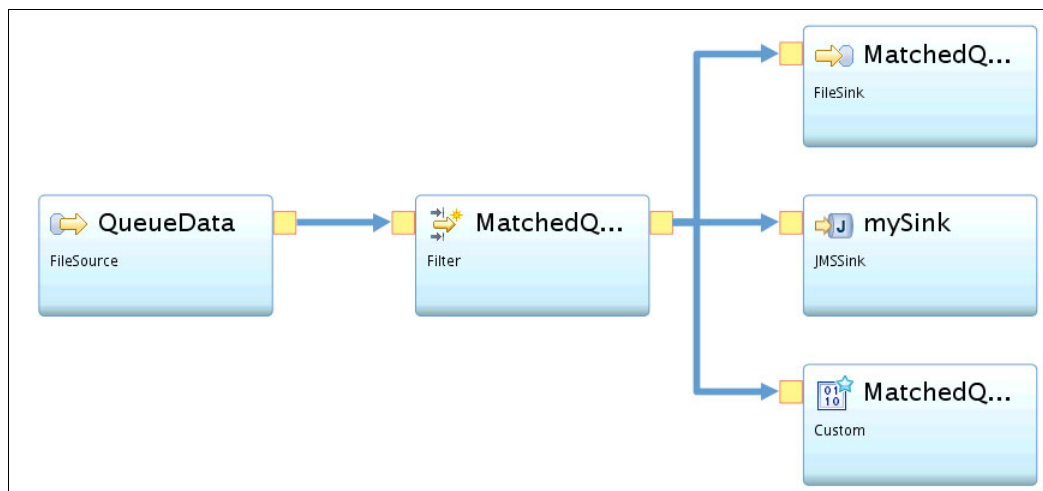


Figure 8-16 QueueMonitor from the graphical SPL editor

- To test the stream, right-click **QueueMonitor** and select **Launch**, then click **Continue** to start processing the queue data. The processed queue status events that are sent to the message queue are shown in the Studio Console as in Example 8-14.

Example 8-14 Output of the processed queue data sent to IBM Integration Bus

```
Matched queue status->{desk="Desk0",queueDepth="90",ts="(1434633614,162477000,0)"}
Matched queue status->{desk="Desk2",queueDepth="81",ts="(1434633618,163803000,0)"}
Matched queue status->{desk="Desk4",queueDepth="129",ts="(1434633622,164723000,0)"}
Matched queue status->{desk="Desk5",queueDepth="94",ts="(1434633624,165429000,0)"}
Matched queue status->{desk="Desk6",queueDepth="133",ts="(1434633626,166087000,0)"}
Matched queue status->{desk="Desk7",queueDepth="86",ts="(1434633628,166695000,0)"}

```

8.4.3 DSI configuration (connectivity and server configuration)

An inbound binding and inbound JMS endpoint must be created for DSI to consume messages from the queue that you created.

Complete the following steps to configure connectivity:

1. Open the **connectivity.cdef** file under the Connectivity Definitions folder in the solution project to add the inbound binding and inbound endpoint for the DSI.FROMSTREAMS queue.

Note: If the connectivity file has not been created before, use the **Define connectivity** link from the Model section in the Solution Map view to create one (Figure 8-17).

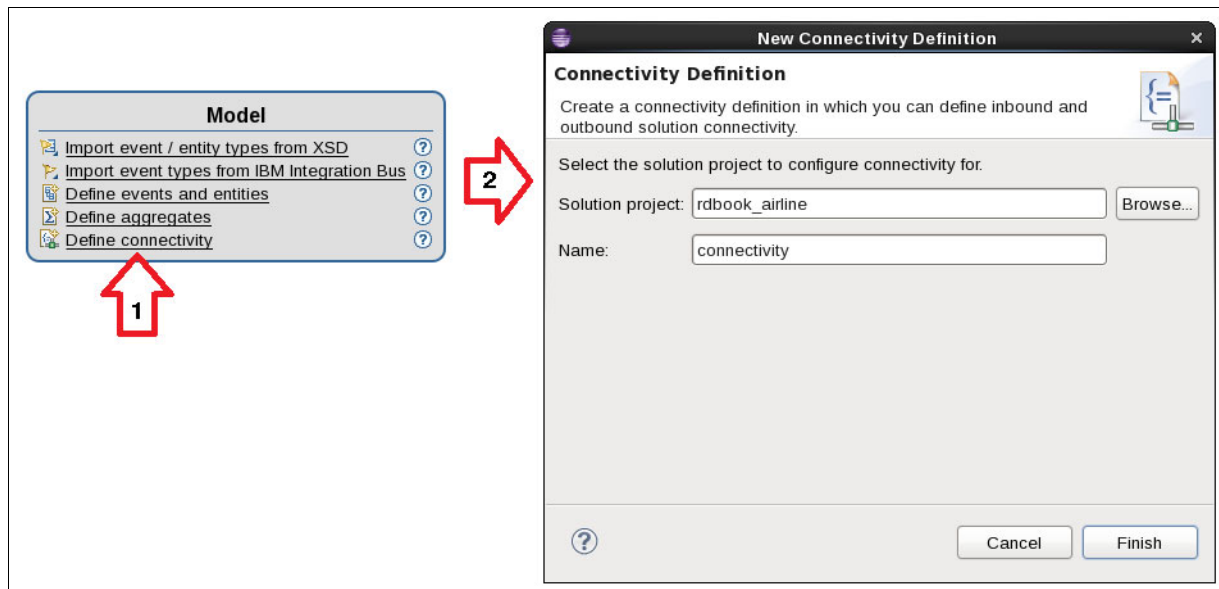


Figure 8-17 Define connectivity configuration from Model section in Solution Map

2. Define inbound binding **queuesBinding** and the **streamsEndPoint** endpoint as shown in Example 8-15.

Example 8-15 Definition of inbound binding and endpoint for streams

```
define inbound binding 'queuesBinding'
  using
    message format application/xml ,
    protocol JMS ,
  accepting events :
    - queue status .

define inbound JMS endpoint 'streamsEndPoint'
  using
    binding 'queuesBinding'.
```

3. Use the **Export connectivity configuration** link from the Deploy section in the Solution Map view to start the configuration wizard as shown in Figure 8-18.

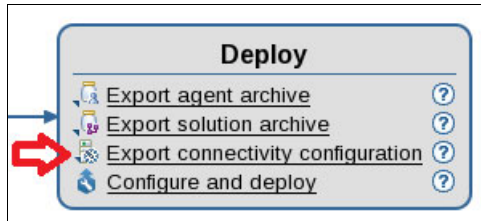


Figure 8-18 Export connectivity configuration from Deploy section in Solution Map

4. Select the solution project, and specify a folder location and connectivity name as shown in Figure 8-19.

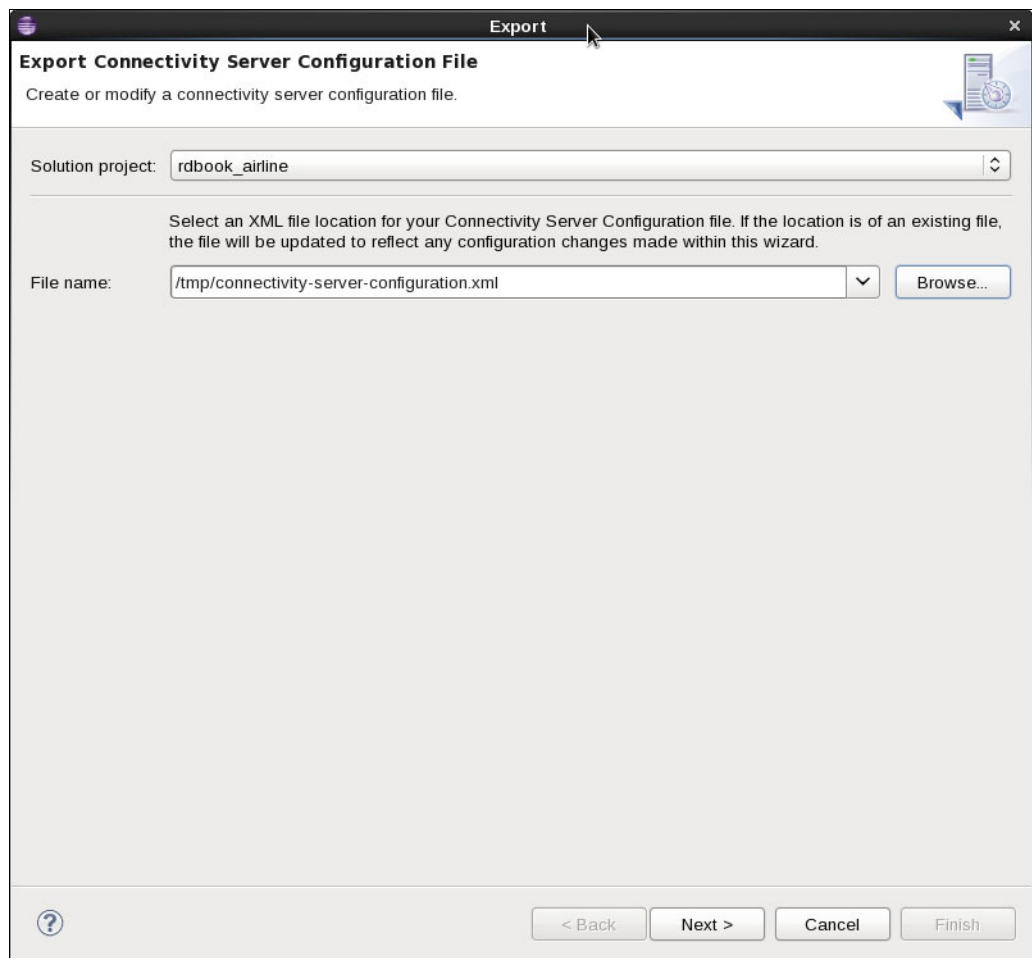


Figure 8-19 Export connectivity server configuration file

Note: Take a note of the location and name that are defined here because this information is needed in the next steps when you work with the connectivity XML file. In this example, that is the `connectivity-server-configuration.xml` file.

- From the list of Inbound Endpoints, select **JMS** and **streamsEndPoint** as shown in Figure 8-20.

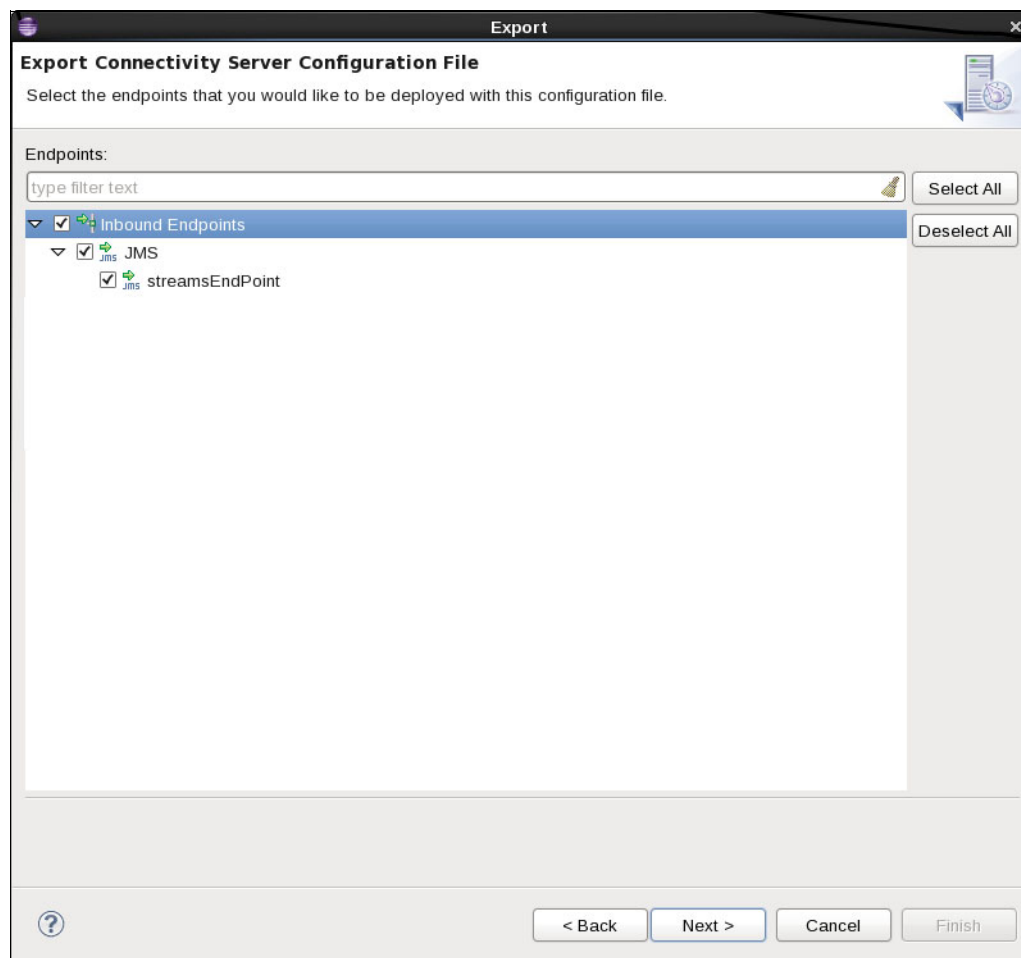


Figure 8-20 Export connectivity server configuration file wizard

- In the Configure and Deploy window (Figure 8-21 on page 141), complete the Inbound JMS Resources as shown in Table 8-1.

Table 8-1 Inbound JMS Resources data

Provider Type	WebSphere MQ
DestinationType	javax.jms.Queue
Destination name	DSI.FROMSTREAMS
Connection Type	CLIENT
Queue manager	IB9QMGR
Host name	9.17.7.22
Port	2414
Channel	SYSTEM.DEF.SVRCONN

Note: Make sure your values of queue manager, host name, and port match your WebSphere MQ configuration.

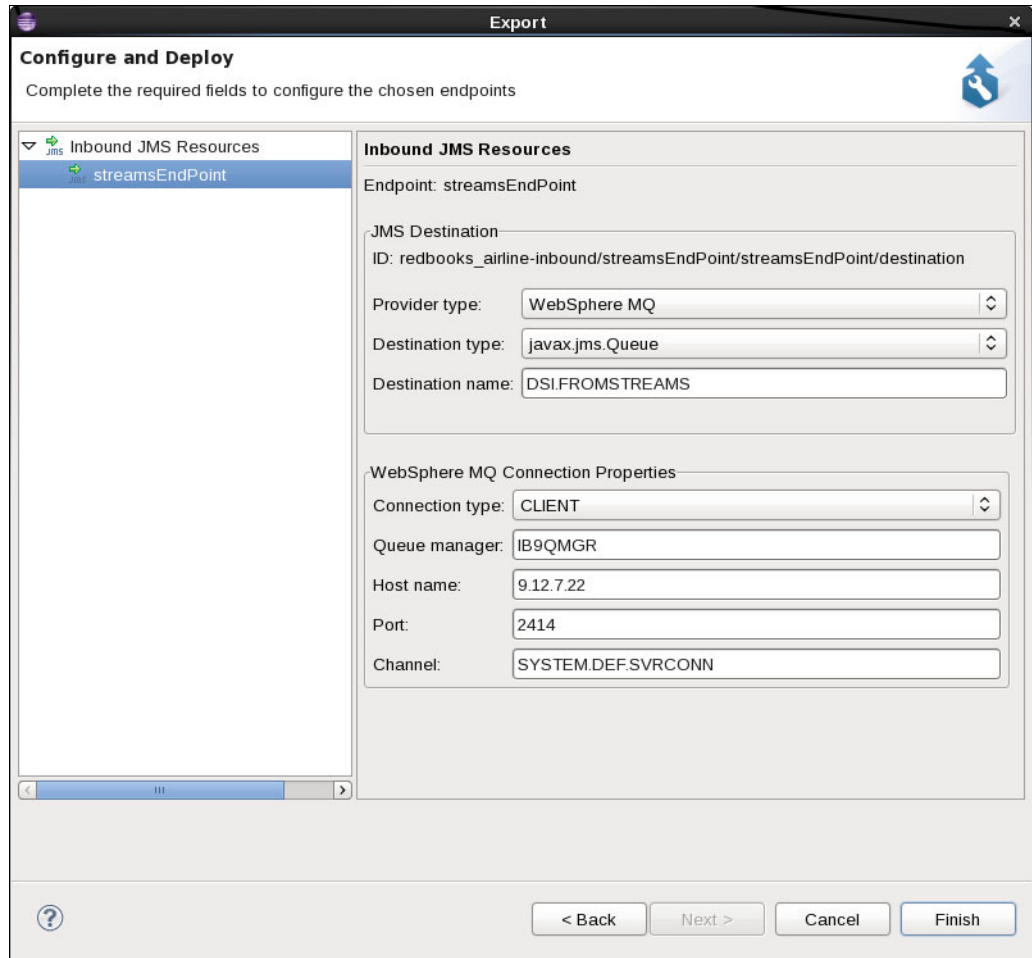


Figure 8-21 Export wizard - configure and deploy

7. Open the server.xml file from your target DSI server to add some <features> and a <variable> entry. For example, with a DSI server name **cisDev**, the server configuration file is at: <DSI-Installation-Folder>/runtime/wlp/usr/servers/**cisDev**/server.xml.
8. Add the ia:iaConnectivityInboundJMS-8.7.1 and ia:iaConnectivityOutboundJMS-8.7.1 features and the wmqJmsClient.rar file location to the server.xml file as shown in Example 8-16.

Example 8-16 server.xml sample for inbound and outbound connectivity features

```
<featureManager>
  <feature>ia:iaConnectivityInboundJMS-8.7.1</feature>
  <feature>ia:iaConnectivityOutboundJMS-8.7.1</feature>
  <feature>wmqJmsClient-1.1</feature>
  <feature>wasJmsClient-1.1</feature>
</featureManager>

<variable name="wmqJmsClient.rar.location"
value="<WMQ-Installation-Folder>/wmq/wmq.jmsra.rar"/>
```

9. Create the authentication data entry at the end of the `server.xml` file as shown in Example 8-17.

Example 8-17 authData definition sample in server.xml file

```
<authData id="iibAuth" password="test123" user="tester"/>
```

Note: Make sure that the user name that is specified in the `authData` entry has access to your queue.

Manually update the `connectivity-server-configuration.xml` file previously defined to include the `authDataRef` entry as shown in Example 8-18.

Example 8-18 Connectivity-server-configuration.xml extract with authDataRef

```
<!--WebSphere MQ messaging provider activation specification-->
  <jmsActivationSpec authDataRef="iibAuth"
id="rdbook_airline-inbound/streamsEndPoint/streamsEndPoint">
    <properties.wmqJms channel="SYSTEM.DEF.SVRCONN"
destinationRef="rdbook_airline-inbound/streamsEndPoint/streamsEndPoint/destination"
destinationType="javax.jms.Queue" hostName="9.12.7.22" port="2414"
queueManager="IB9QMGR" transportType="CLIENT"/>
  </jmsActivationSpec>
```

Note: Make sure that the name specified in the `authDataRef` matches the ID in the `authData` entry of the `server.xml` configuration file.

10. Redeploy the solution to make sure that all changes are deployed to the server. Highlight the solution project by right-clicking the project, then clicking **Deploy** → **DeploymentName** where *DeploymentName* corresponds to your deployment configuration.
11. Use the XML code in Example 8-19 to add a test queue status message to the queue.

Example 8-19 XML Example code for queueDepth message

```
<?xml version="1.0" encoding="UTF-8"?><m:QueueStatus
xmlns:m="http://www.ibm.com/ia/xmlns/default/rdbook_airline_bom/model"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.ibm.com/ia/xmlns/default/rdbook_airline_bom/model
../rdbook_airline/schemas/namespace1/model.xsd ">
<m:desk>desk-bd-03</m:desk><m:queueDepth>33</m:queueDepth><m:timestamp>2001-12-31T12:00:
00</m:timestamp></m:QueueStatus>
```

Note: Schema location value must be set to the location of the model schema definition. In the example, the schema location is a relative path:
`../../rdbook_airline/schemas/namespace1/model.xsd`

For the steps to create the model schema, see the IBM Knowledge Center at the following web address:

http://www.ibm.com/support/knowledgecenter/SSQP76_8.7.1/com.ibm.odm.itoa.develop/modeling_topics/tsk_export_xsd_from_bom.html

For the steps to create a sample file for an existing schema, see the IBM Knowledge Center at the following web address:

http://www.ibm.com/support/knowledgecenter/SSQP76_8.7.1/com.ibm.odm.itoa.develop/modeling_topics/tsk_create_event_xml.html

8.5 Analyze and report example

For the analyze phase, this book uses DSI, SPSS Modeler, and SPSS Collaboration and Deployment Services to demonstrate how predictive analytics results can be used to inform decisions. This example uses a predictive model to determine how long a passenger will take to arrive at the gate for their flight given their current location. If this time exceeds the length of time between now and the flight's departure time, the system generates an event to flag that this passenger will be late.

To implement this use case, SPSS Modeler was used to create a simple model that receives some input values and provides one output value. For the purposes of the scenario, an arbitrary return value is used in this example rather than one based on some specific analysis. As described in 5.2.4, “The predictive modeling process with SPSS” on page 57, SPSS Modeler provides robust capabilities for creating Predictive models, but this function is considered out of scope for the example implementation.

The following sections describe the process to achieve this.

8.5.1 Create scenario model

Use the following steps to create your model:

1. Open SPSS Modeler to create a new Stream as shown in Figure 8-22.

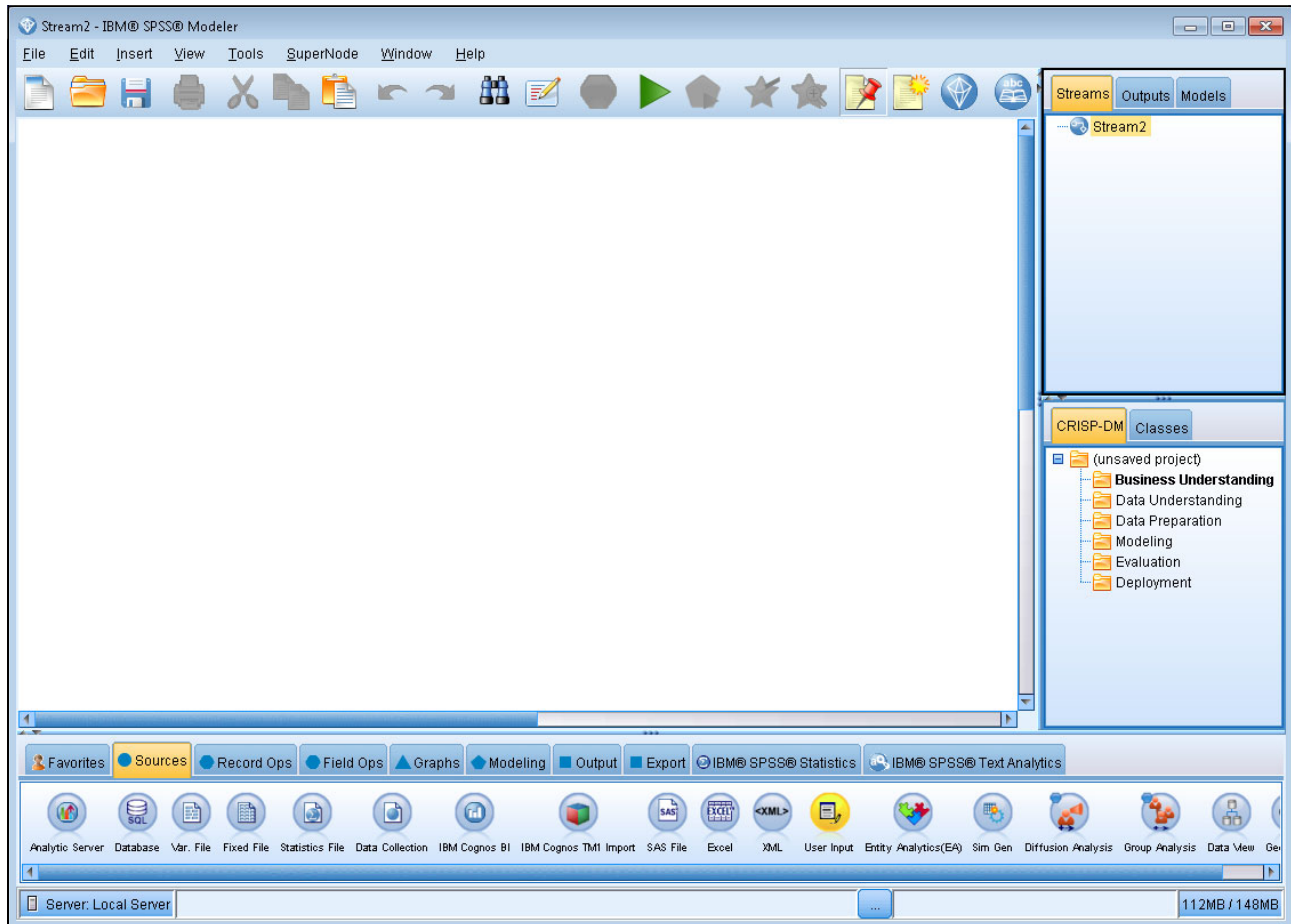


Figure 8-22 SPSS Modeler opened for a new stream

2. Create an **Input Source** with three fields: Zone, gate, and age. Match the structure of these fields to the details shown in Figure 8-23. Data for these fields is going to be passed to SPSS from DSI based on current location of the passenger (zone), gate location of the flight, and age range of the passenger.

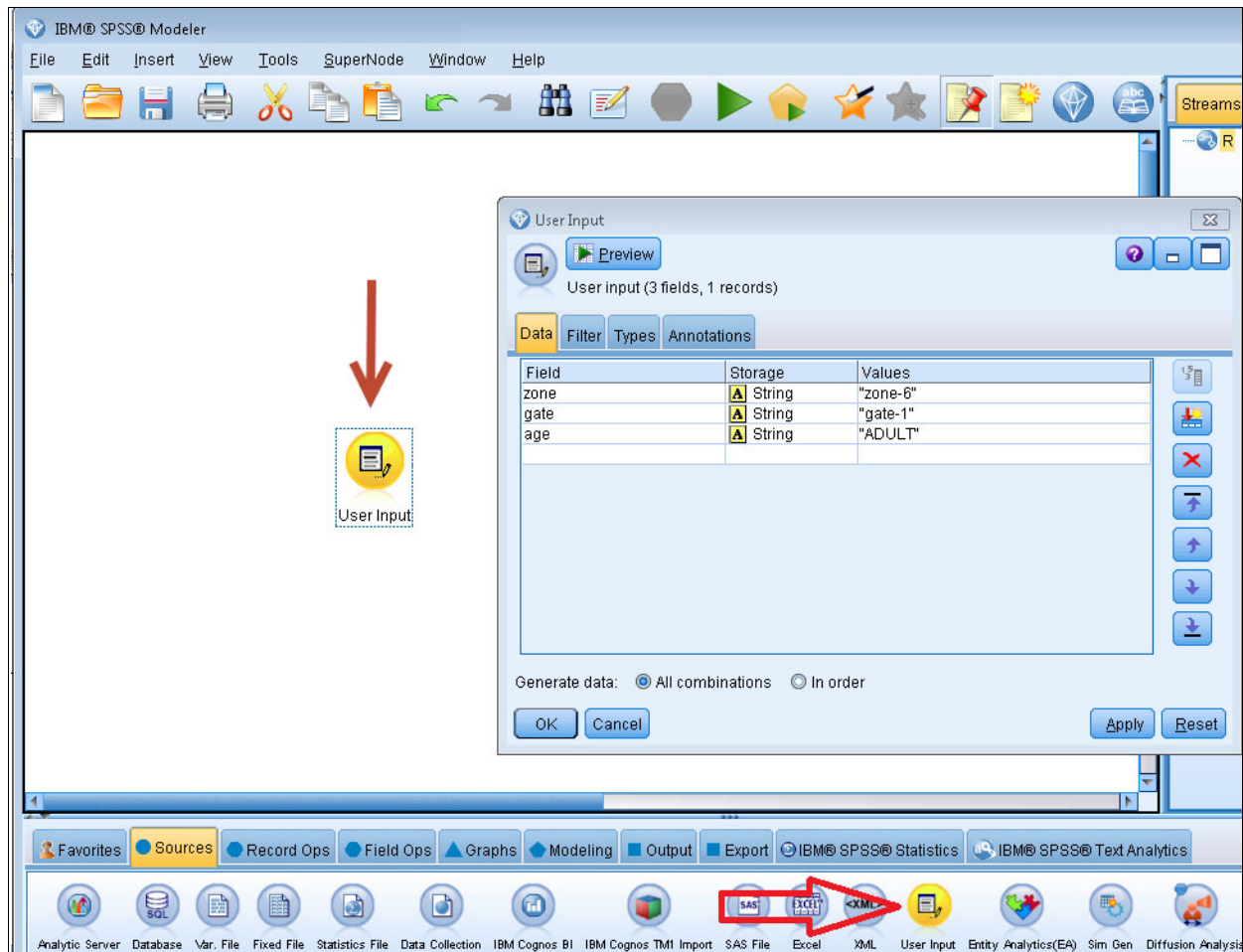


Figure 8-23 SPSS Modeler - create new user input

3. Create a **Derive** field operation and name it **timeToGate** with the code shown in Example 8-20 as the formula.

Example 8-20 Example of a Derive field logic

```

if zone == "zone-1" and gate == "gate-1" and age == "ADULT" then 10
elseif zone == "zone-1" and gate == "gate-1" and age == "CHILD" then 100
elseif zone == "zone-1" and gate == "gate-1" and age == "SENIOR" then 50
elseif zone == "zone-2" then 200
elseif zone == "zone-3" then 300
elseif zone == "zone-4" then 400
else 900 endif

```

Important note: Be aware that the code used in Example 8-20 is to simulate a return value back to caller application. It does not show any SPSS Modeler special functionality.

The timeToGate field details look like Figure 8-24.

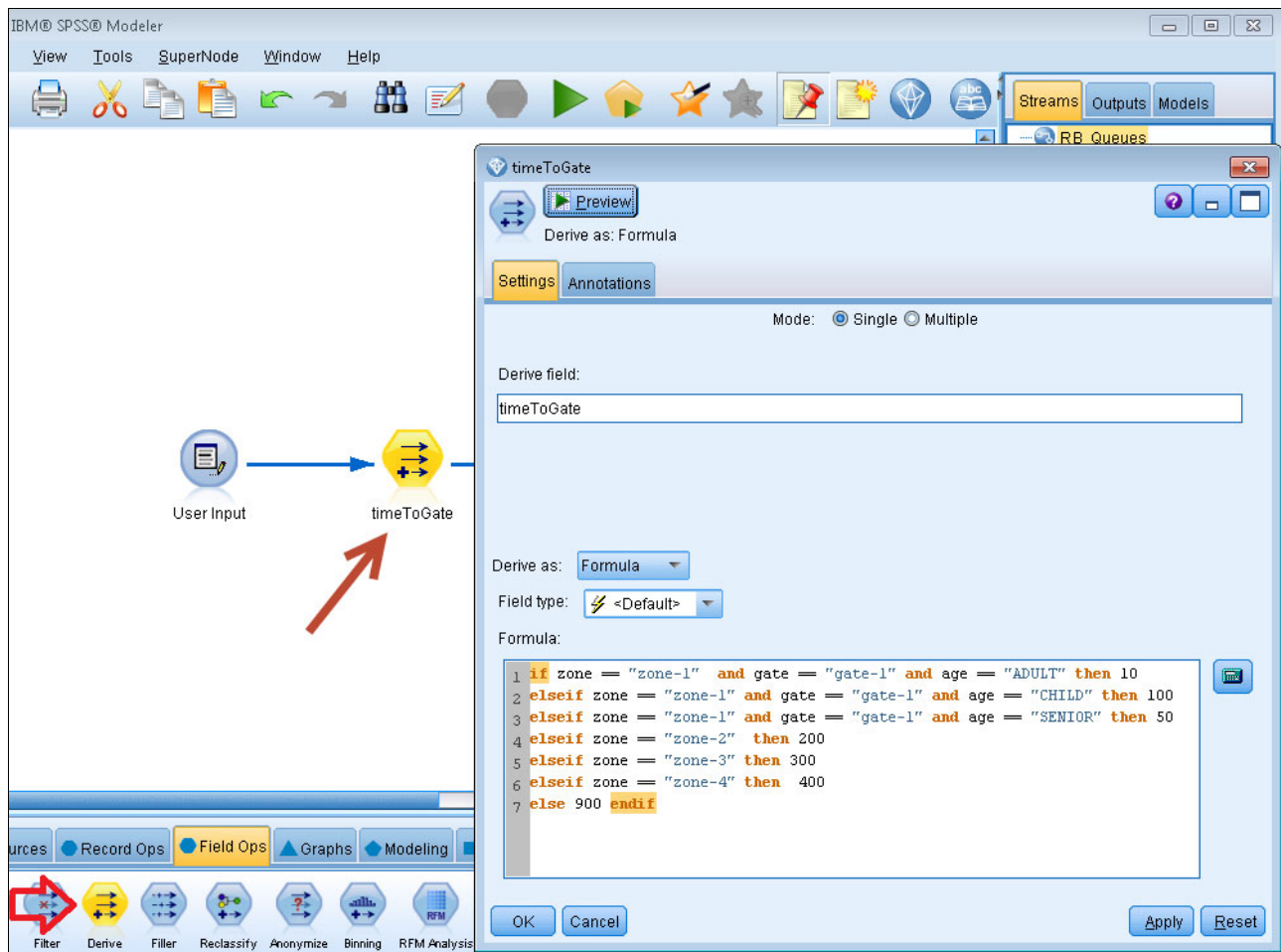


Figure 8-24 SPSS Modeler: Derive field definition

4. Create a **Filter** field operation to remove zone, gate, and age from the list and leave **timeToGate** not filtered as shown in Figure 8-25.

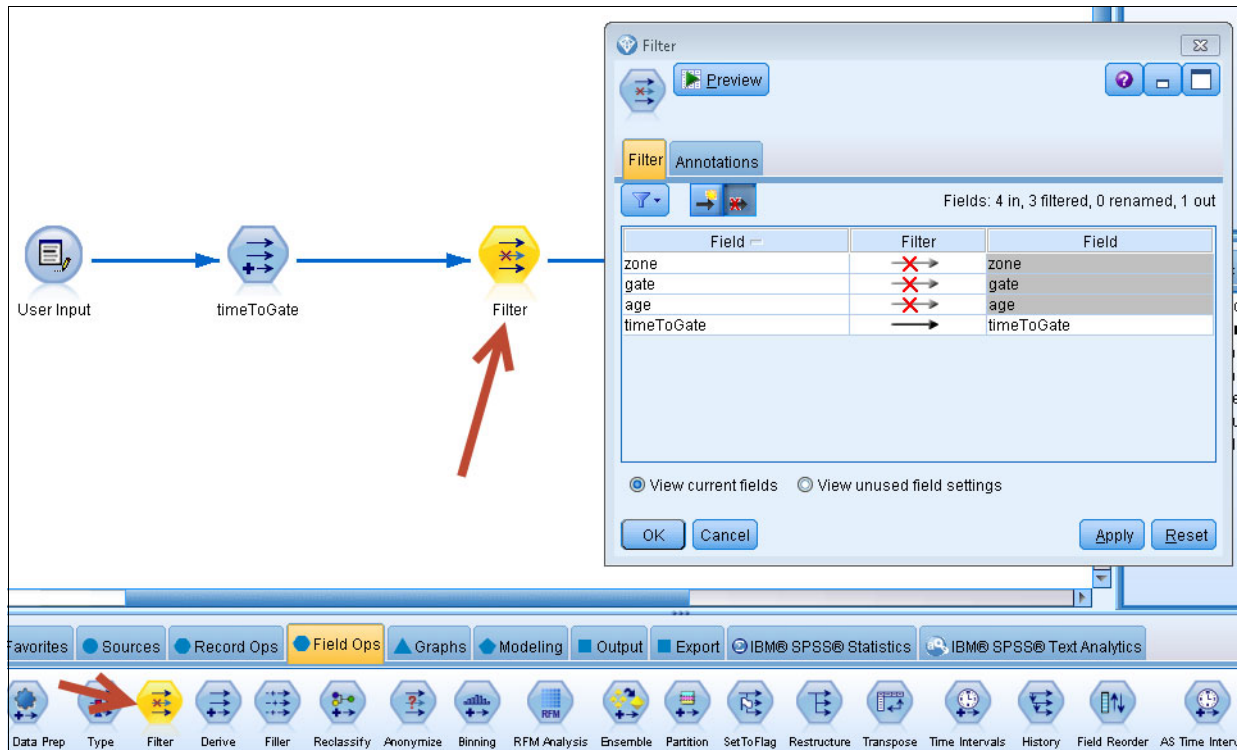


Figure 8-25 SPSS Modeler: Filter configuration

5. Add a **Table** output for the results to DSI as shown in Figure 8-26. In this case, only **timeToGate** is being sent out as the result to DSI.

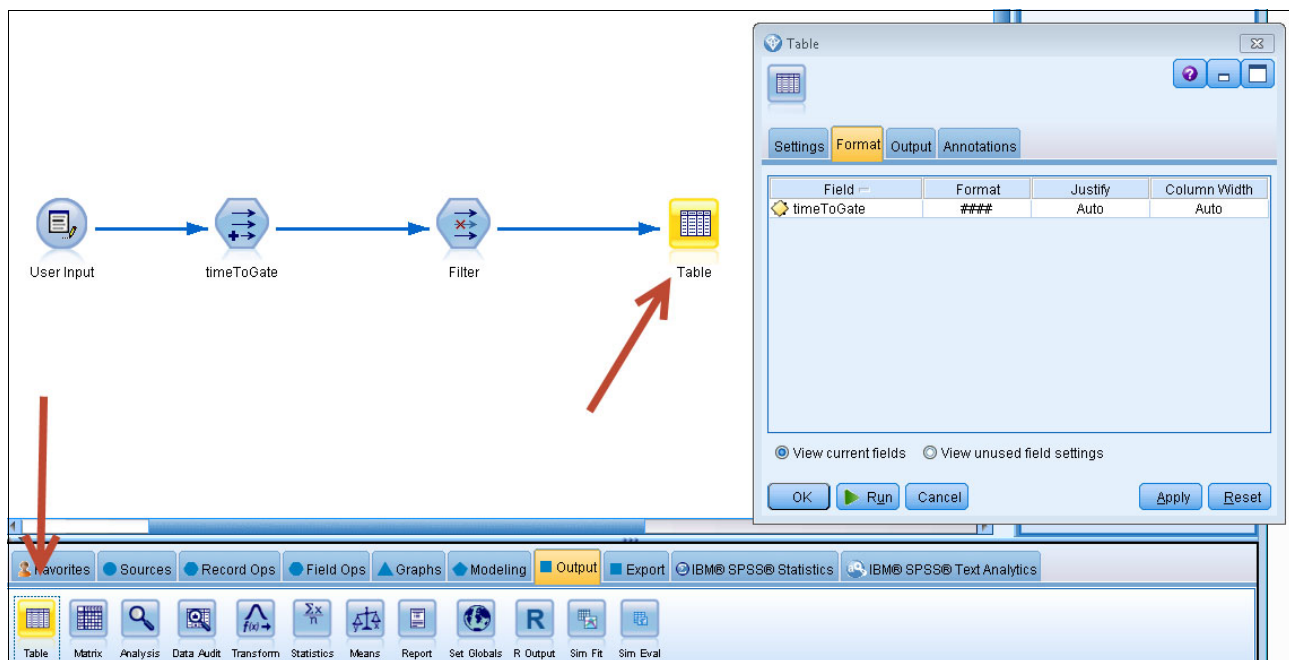


Figure 8-26 SPSS Modeler - table output definition

6. Link all nodes to create your model as shown in Figure 8-27.

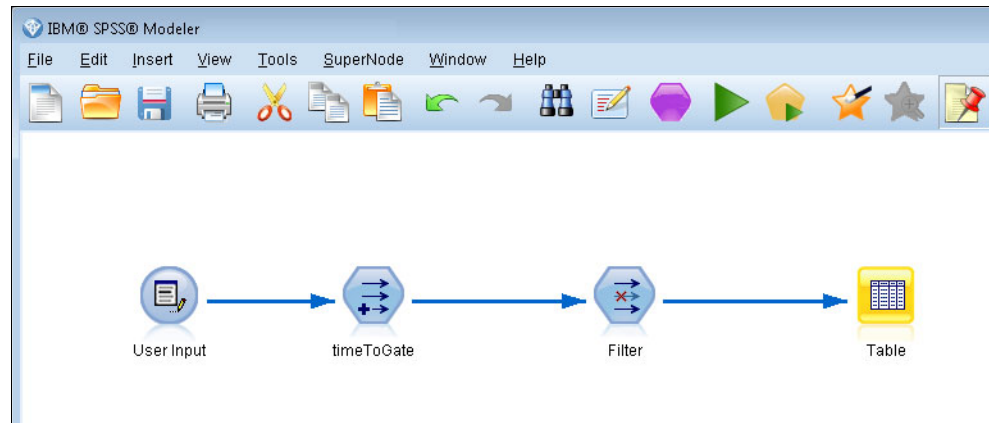


Figure 8-27 SPSS Modeler - Final version of the stream

The result is a simple model that just works as a pass-through flow with basic logic associated to it. In real life examples, models are complex and include many modeling algorithms and data manipulation.

8.5.2 Store model as a stream on the server

Complete the following steps to store the model as a stream on the server:

1. After the model is created, it must be stored in the Collaboration and Deployment Services (C&DS) Server. From SPSS Modeler, you can select the **Store as Stream** option under **Store** as shown in Figure 8-28.

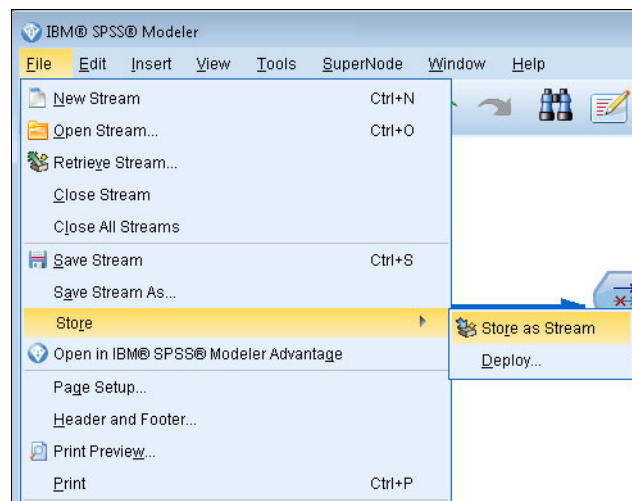
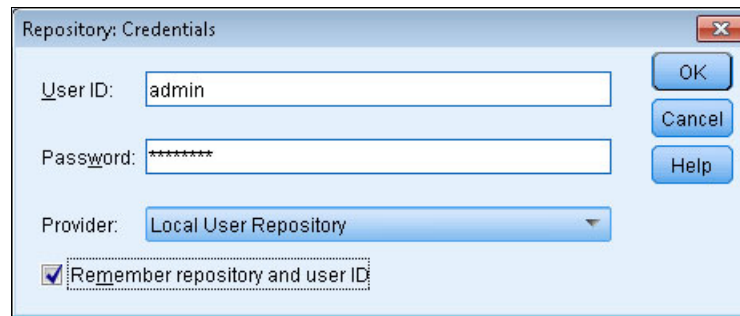


Figure 8-28 SPSS Modeler - Store as Stream menu

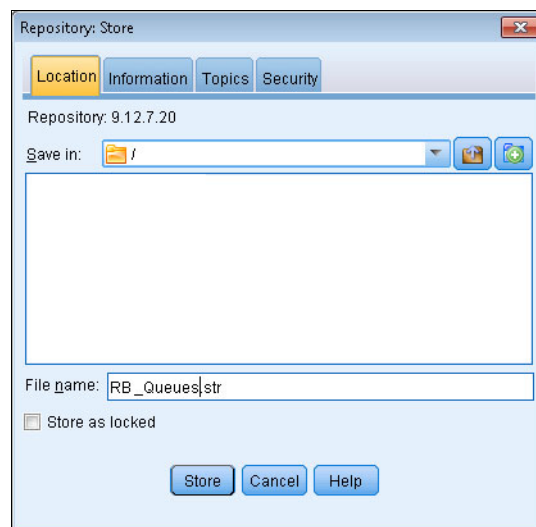
2. Provide repository credentials as shown in Figure 8-29.



The 'Repository: Credentials' dialog box is shown. It has a title bar with a close button. Inside, there are three input fields: 'User ID' with the text 'admin', 'Password' with seven asterisks, and 'Provider' with a dropdown menu showing 'Local User Repository'. To the right of these fields are three buttons: 'OK', 'Cancel', and 'Help'. At the bottom, there is a checkbox labeled 'Remember repository and user ID' which is checked.

Figure 8-29 SPSS Modeler - Repository credentials wizard

3. Specify location in the server repository where the model will be stored as shown in Figure 8-30.



The 'Repository: Store' dialog box is shown. It has a title bar with a close button. Below the title bar are four tabs: 'Location', 'Information', 'Topics', and 'Security'. The 'Location' tab is selected. Below the tabs, it says 'Repository: 9.1.2.7.20'. There is a 'Save in:' field with a folder icon and a dropdown menu showing 'f'. Below this is a large empty text area. At the bottom, there is a 'File name:' field with the text 'RB_Queue|str'. Below that is a checkbox labeled 'Store as locked' which is unchecked. At the bottom right are three buttons: 'Store', 'Cancel', and 'Help'.

Figure 8-30 SPSS Modeler - Repository store wizard

4. Validate that the model file was properly uploaded to the repository by accessing the C&DS Deployment Portal, as shown in Figure 8-31.

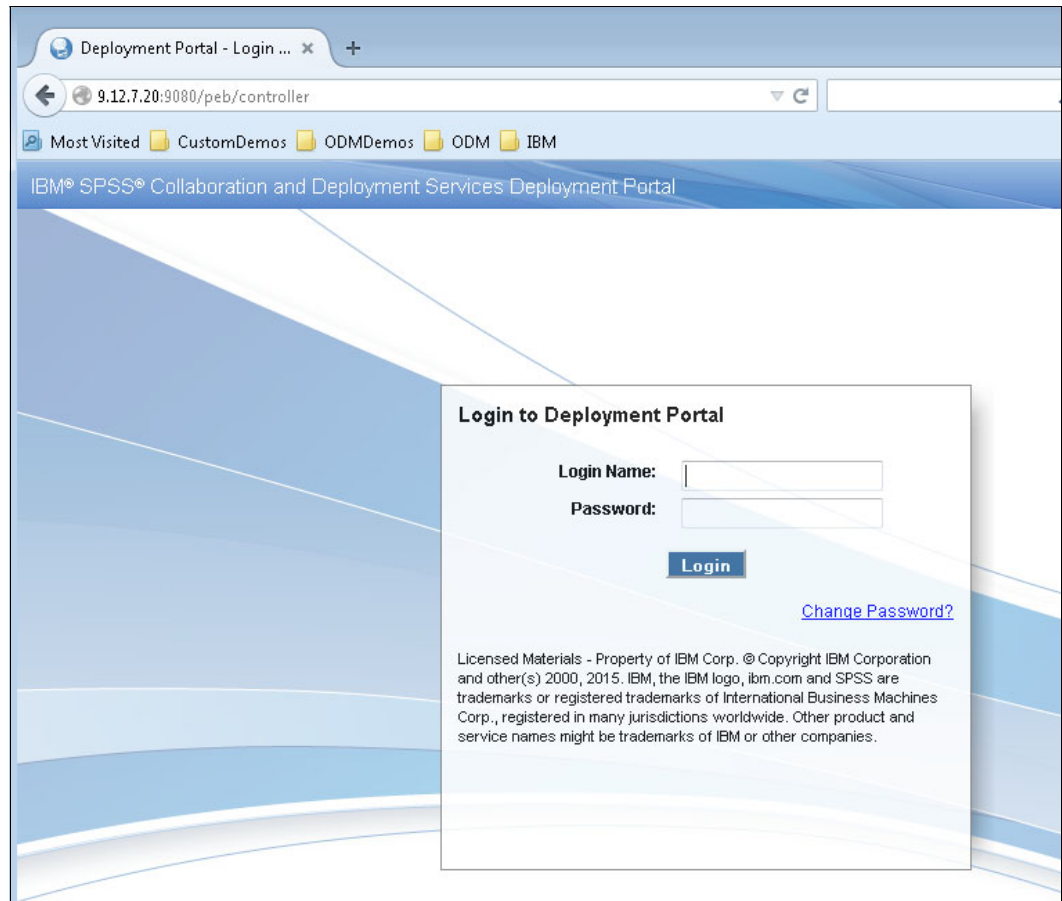


Figure 8-31 Deployment Portal login page

Deployment portal URL sample: http://<server>:<port>/peb/controller

5. Under the Content Repository section of the Deployment Portal, make sure that the model file is listed. In this example, the file name is `RB_Queue.str` as shown in Figure 8-32.

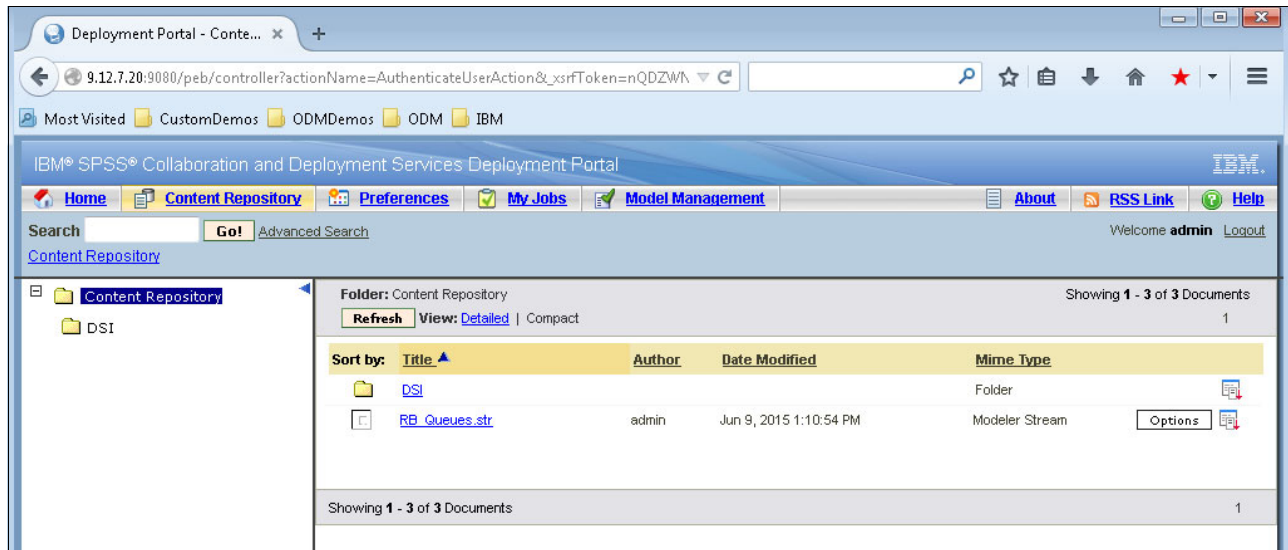


Figure 8-32 Deployment Portal - content repository

8.5.3 Scoring model configuration

The model is now properly uploaded to the repository and it is ready to be configured. To configure the scoring service, use these steps in the IBM SPSS C&DS Deployment Manager console:

1. Open the SPSS C&DS console and select the link to create a new administrative server in the Server Administration view as shown in Figure 8-33.

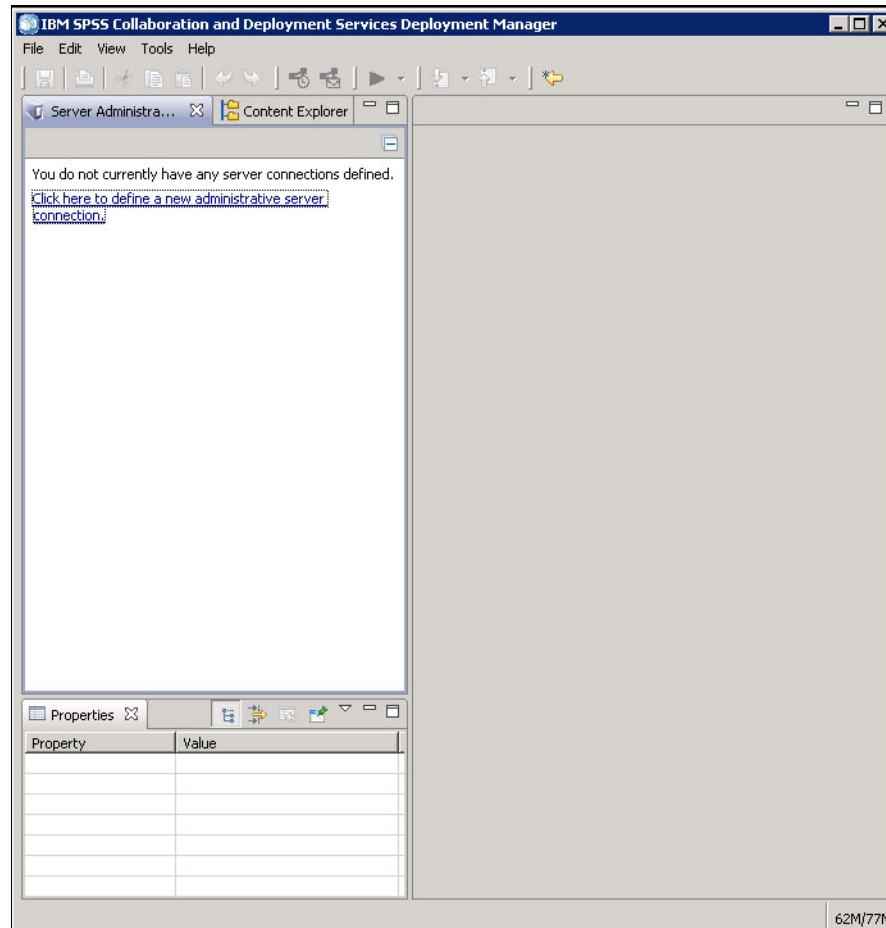


Figure 8-33 SPSS C&DS: Deployment Manager console

2. Provide a name to identify the repository server as shown in Figure 8-34.

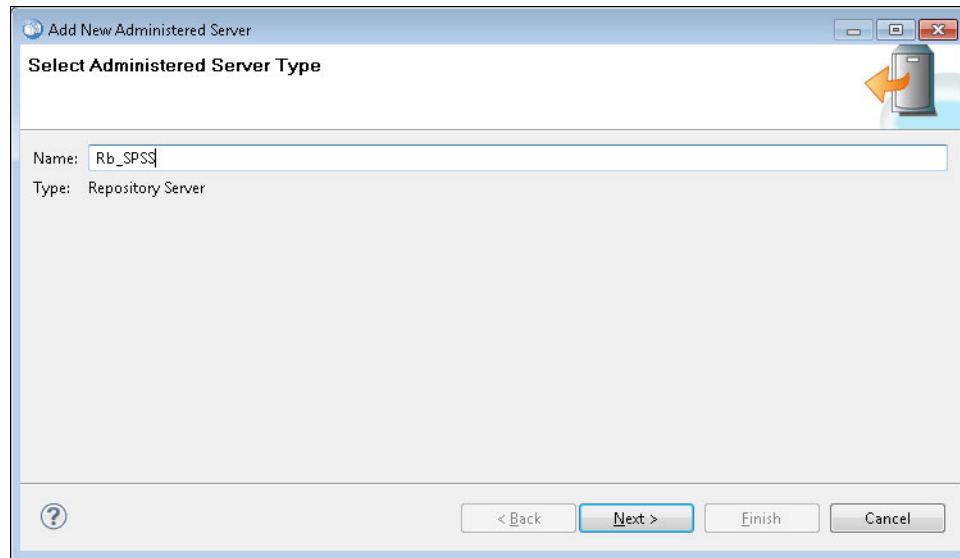


Figure 8-34 SPSS C&DS Deployment Manager: New administered server

3. Provide the server URL for the repository as shown in Figure 8-35.

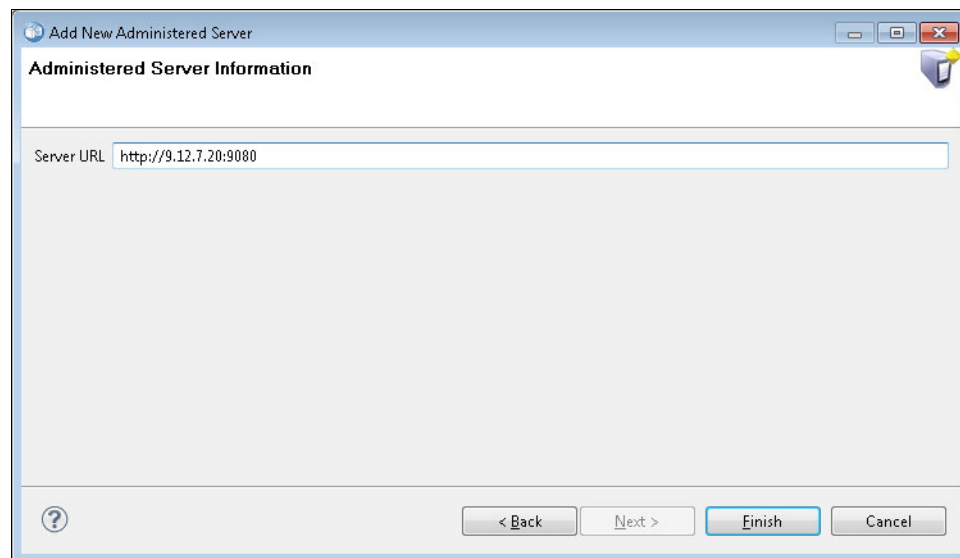


Figure 8-35 SPSS C&DS Deployment Manager console: Administered server URL

URL format: Make sure that the URL includes the protocol, server name, and port. For example: `http://localhost:9080`.

4. A successful connection to the server lists the server in the Server Administration view as shown in Figure 8-36.

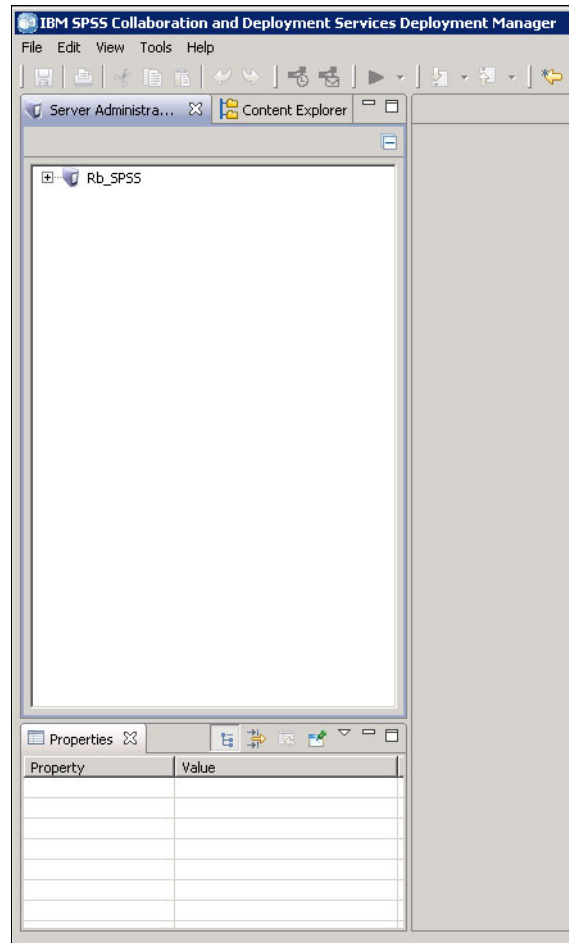


Figure 8-36 SPSS C&DS Deployment Manager console: Servers view

5. Log in to the server as shown in Figure 8-37.

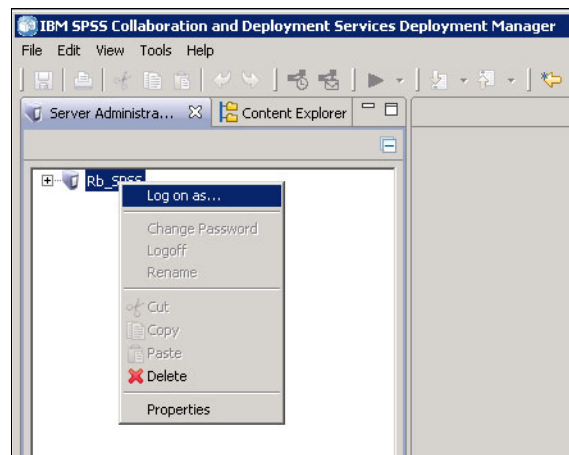


Figure 8-37 SPSS C&DS Deployment Manager console: Log on to server

6. Provide the appropriate credentials as shown in Figure 8-38.

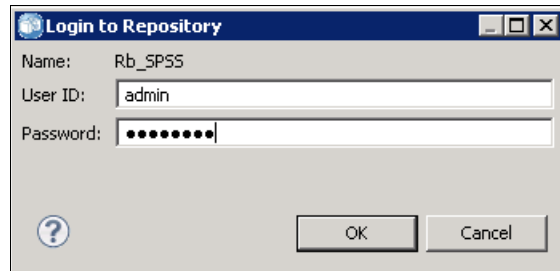


Figure 8-38 SPSS C&DS Deployment Manager console: Credentials for login

7. If the Content Explorer view is not open already, open it through the **View** menu as shown in Figure 8-39.

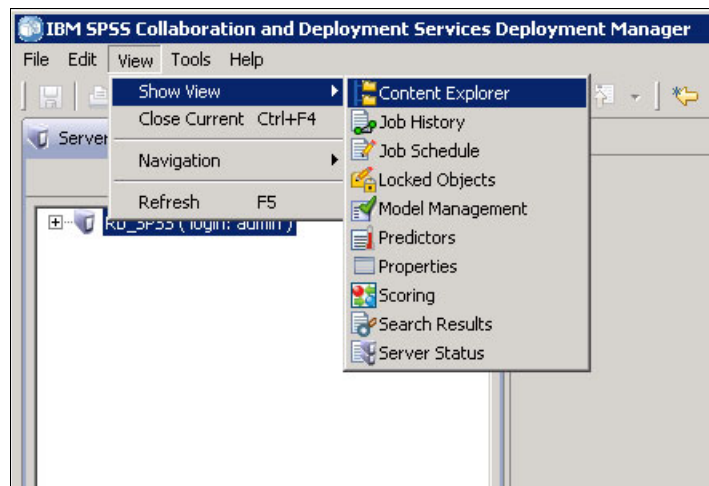


Figure 8-39 SPSS C&DS Deployment Manager console: Content Explorer view

8. Create a **Content Server Connector** as shown in Figure 8-40.

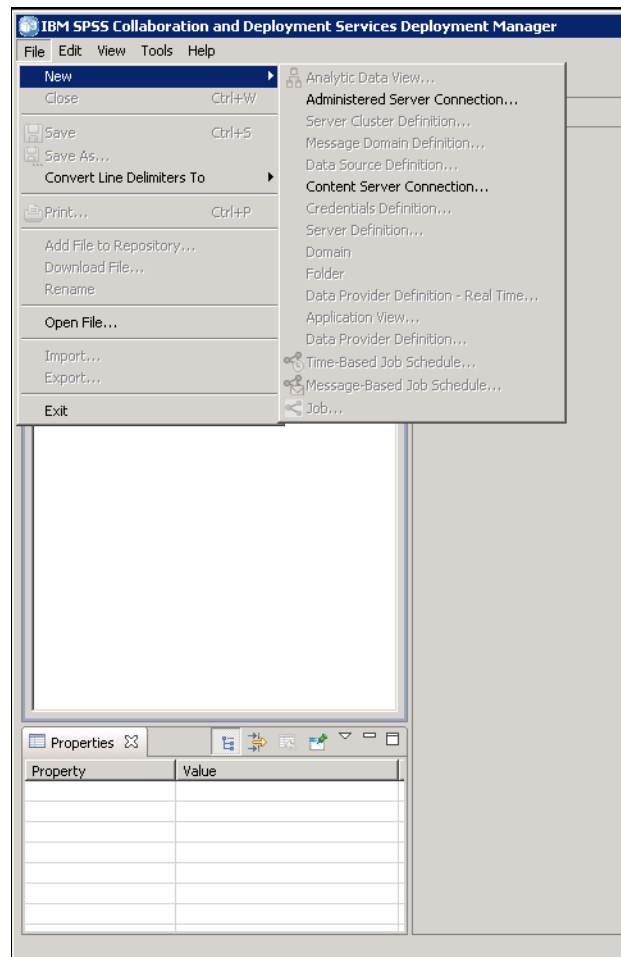


Figure 8-40 SPSS C&DS Deployment Manager console: New content server

9. Provide a name and server URL as shown in Figure 8-41.

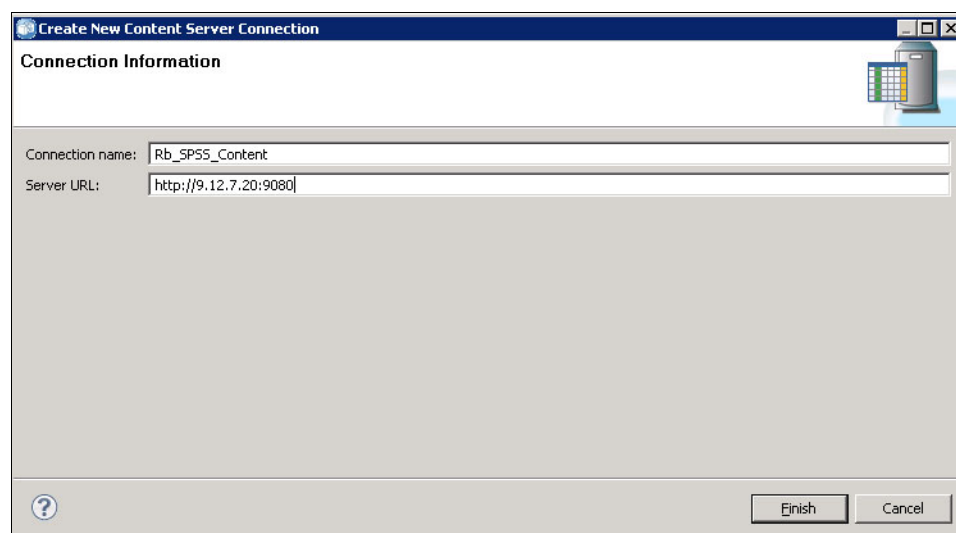


Figure 8-41 Content server URL connection information

10. Enter the appropriate credentials as shown in Figure 8-42.

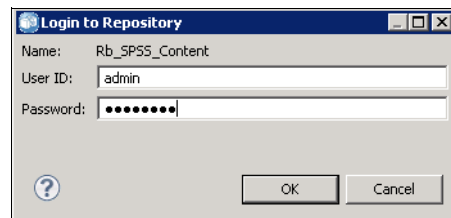


Figure 8-42 Content repository credentials

11. Expand the **Content Repository** and make sure that the model file created in the previous section is listed. In this example, it is **RB_Queue.str** as shown in Figure 8-43.

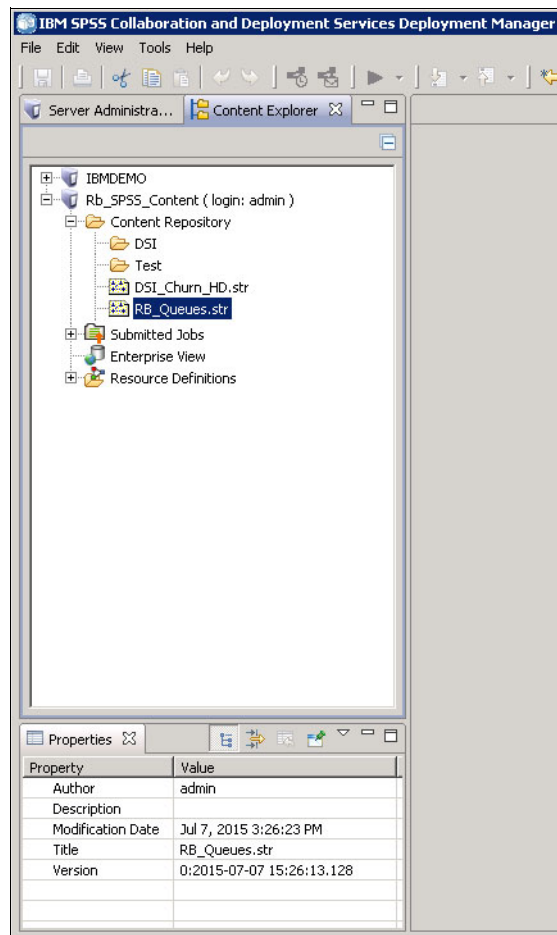


Figure 8-43 SPSS C&DS Deployment Manager console: Content Explorer view

12. Configure the scoring service as described in Figure 8-44.

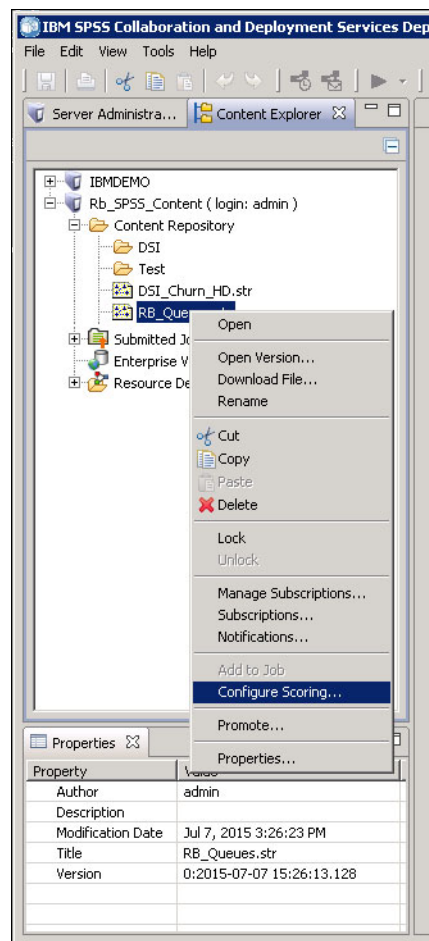


Figure 8-44 SPSS C&DS Deployment Manager console: Configure scoring

13. Define the Model File behavior for the scoring service as shown in Figure 8-45.

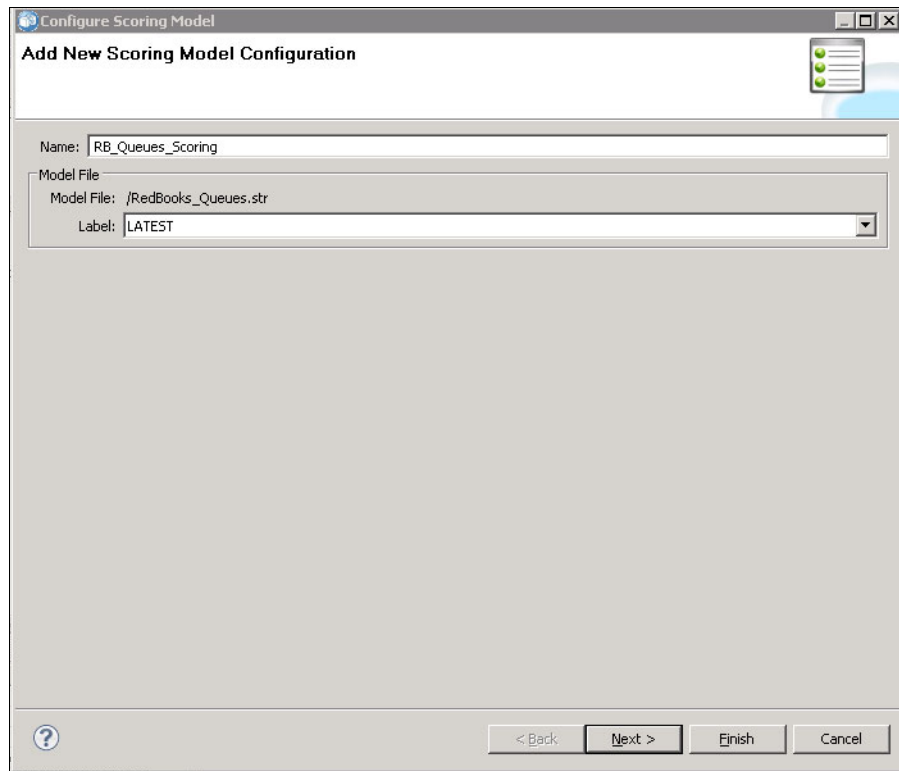


Figure 8-45 SPSS C&DS Deployment Manager console: Add new scoring model to server

14. Click **Finish** because no further configuration is necessary.

8.5.4 Predictive analytics agent implementation

The following steps describe the process used to implement the agent for the sample scenario:

1. Start the predictive scoring agent wizard using the link from the Solution Map view as shown in Figure 8-46

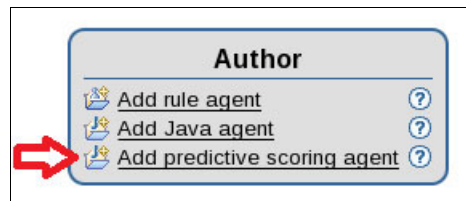
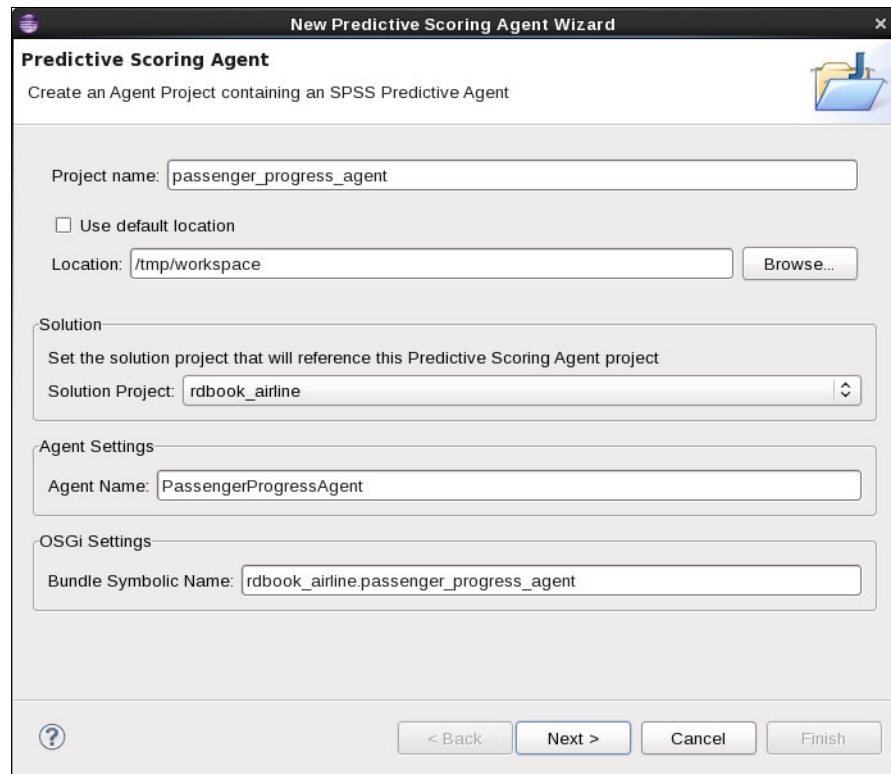


Figure 8-46 Insight Designer: Add predictive scoring agent in Solution Map

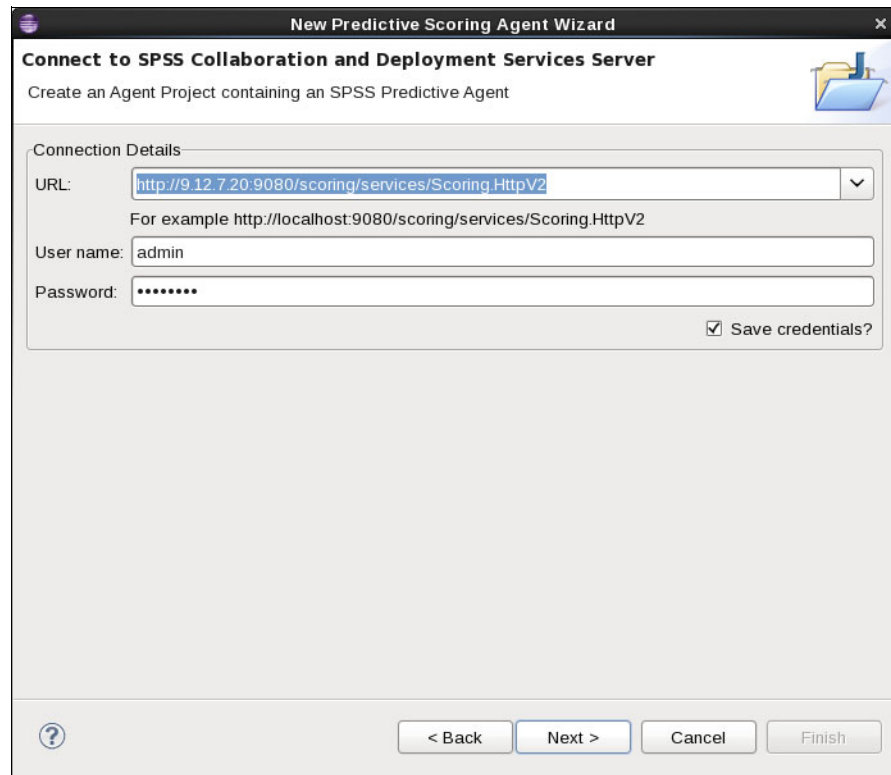
2. Provide requested information. You can use **passenger_progress_agent** as the project name and **PassengerProgressAgent** as the agent name (Figure 8-47).



The image shows a 'New Predictive Scoring Agent Wizard' dialog box. The title bar reads 'New Predictive Scoring Agent Wizard'. The main heading is 'Predictive Scoring Agent' with a subtext 'Create an Agent Project containing an SPSS Predictive Agent'. The dialog is divided into several sections: 'Project name' with a text field containing 'passenger_progress_agent'; a checkbox 'Use default location' which is unchecked; 'Location' with a text field containing '/tmp/workspace' and a 'Browse...' button; 'Solution' section with a subtext 'Set the solution project that will reference this Predictive Scoring Agent project' and a 'Solution Project' dropdown menu showing 'rdbook_airline'; 'Agent Settings' section with an 'Agent Name' text field containing 'PassengerProgressAgent'; and 'OSGi Settings' section with a 'Bundle Symbolic Name' text field containing 'rdbook_airline.passenger_progress_agent'. At the bottom, there is a help icon (question mark) and four buttons: '< Back', 'Next >', 'Cancel', and 'Finish'.

Figure 8-47 Insight designer: New predictive scoring agent wizard

3. Provide the URL where the scoring service is running and the credentials to access the service, as shown in Figure 8-48.



The image shows a software wizard window titled "New Predictive Scoring Agent Wizard". The main heading is "Connect to SPSS Collaboration and Deployment Services Server". Below this, it says "Create an Agent Project containing an SPSS Predictive Agent". The "Connection Details" section contains the following fields:

- URL:** A text box containing "http://9.12.7.20:9080/scoring/services/Scoring.HttpV2" with a dropdown arrow on the right. Below it is a hint: "For example http://localhost:9080/scoring/services/Scoring.HttpV2".
- User name:** A text box containing "admin".
- Password:** A text box with masked characters "*****".

At the bottom right of the form area is a checkbox labeled "Save credentials?" which is checked. The bottom of the window features a navigation bar with a help icon (question mark) on the left and four buttons: "< Back", "Next >", "Cancel", and "Finish".

Figure 8-48 New predictive scoring agent: Connection details

4. From the available scoring configurations, select the one created in the previous section, as shown in Figure 8-49.

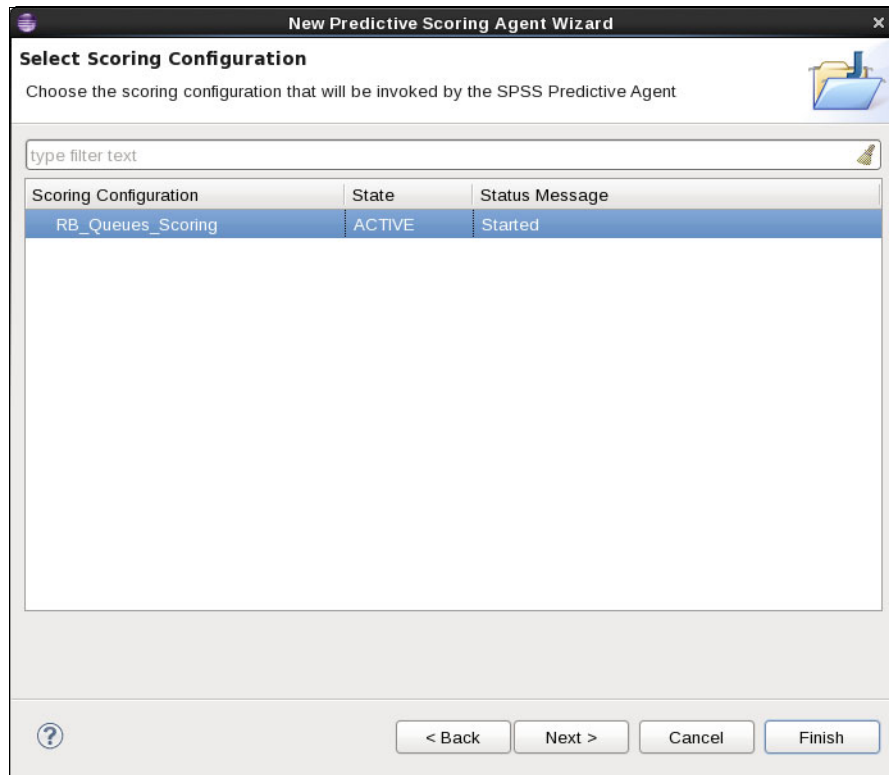


Figure 8-49 New predictive scoring agent wizard: Scoring configuration selection

5. Provide the Scoring Endpoint name as shown in Figure 8-50.

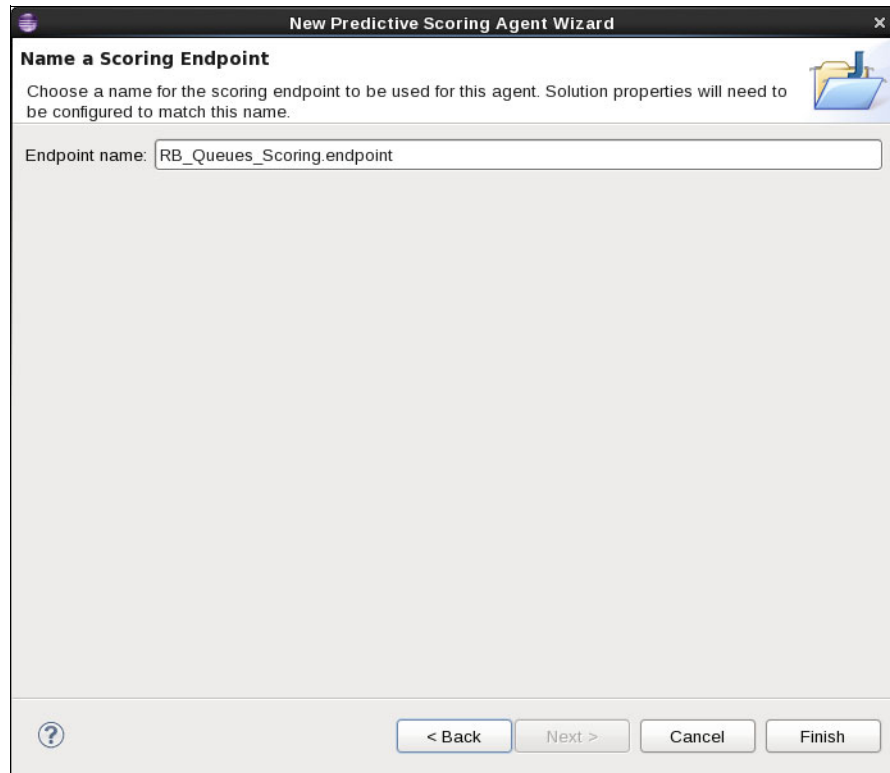


Figure 8-50 New predictive scoring agent wizard: Scoring endpoint selection

6. Update the agent descriptor to process **desk interaction** events as shown in Figure 8-51.



Figure 8-51 Predictive scoring agent descriptor

7. Update the `process()` method from **PassengerProgressAgent** class with code to retrieve data from the received event and assemble the data for SPSS invocation as shown in Example 8-21.

Example 8-21 Sample code for process method in *PassengerProgressAgent*

```
...
boolean scoreOnThisInvocation = true;

Booking eventBooking = (Booking)((DeskInteraction)event).getBooking().resolve();
Passenger passenger = (Passenger) eventBooking.getPassenger().resolve();
Flight flight = (Flight) eventBooking.getFlight().resolve();
Desk eventDesk = (Desk)((DeskInteraction)event).getDesk().resolve();
```

8. Assign column values to be used as input for the SPSS invocation as shown in Example 8-22.

Example 8-22 Values assignment sample for scoring fields

```
// Assign the value for column: zone
    table1row1.setColumnValue(ZONE, eventDesk.getZone());

// Assign the value for column: gate

    table1row1.setColumnValue(GATE, flight.getGate().getGateNumber());

// Assign the value for column: age
    table1row1.setColumnValue(AGE, passenger.getAge().name().toString());
```

9. Add the code needed to handle the SPSS scoring result and add some simple logic to validate if passenger is going to be late or not to their gate. Use the code from Example 8-23.

Example 8-23 Sample code to validate time to gate based on SPSS scoring result

```
...
ScoringOutputRow row0 = scoringOutput.getRow(0);
...
    // Use the value of the column: timeToGate
    String output1 = row0.getColumnValue(TIME_TO_GATE);

    Long timeToGate = Long.valueOf(output1);
    LocalTime now = LocalTime.now();
    if(now.plusMinutes(timeToGate.longValue()).isAfter(flight.getDepartureTime())){

        ZonedDateTime zdtFlightDepartureTime = ZonedDateTime.of(LocalDate.now(),
flight.getDepartureTime(), ZoneId.systemDefault());

        TimePeriod lateness = new TimePeriod(ZonedDateTime.now(),
zdtFlightDepartureTime);

        Relationship<Booking> bookingRel = ((DeskInteraction)event).getBooking();
        LatePassenger lpEvent =
getConceptFactory(ConceptFactory.class).createLatePassenger(bookingRel, lateness,
ZonedDateTime.now());

        emit(lpEvent);
    }
    else{
        System.out.println("----PassengerProgressAgent --> Passenger is ONTIME");
    }
}
```

10. Update **solution_properties.xml** file in your solution project to include SPSS properties related to your C&DS server as shown in Example 8-24.

Example 8-24 SPSS properties entries in solution_properties.xml file

```
<property
name="RB_QueueScoring.endpoint.url">http://9.12.7.20:9080/scoring/services/Scoring.HttpV2</property>
<property name="RB_QueueScoring.endpoint.user">admin</property>
<property name="RB_QueueScoring.endpoint.password">adminPass</property>
```

Note: The password value can be encoded in the `solution_properties.xml` file. The following link describes how to achieve this:

http://www.ibm.com/support/knowledgecenter/SSQP76_8.7.1/com.ibm.odm.itoa.admin/topics/tsk_encrypt_prop.html?lang=en

For more information about how to create a predictive scoring agent, see the IBM Knowledge Center at this location:

http://www.ibm.com/support/knowledgecenter/SSQP76_8.7.1/com.ibm.odm.itoa.develop/topics/tsk_dev_predictive_agent.html?lang=en

8.6 Detect and decide example

This section describes how to use the capabilities of ODM DSR to determine what to do with late passengers identified in the previous section.

The solution involves writing an OSGi service to start the DSR decision service from DSI, mapping this service to a BOM entry and verbalizing it, and then calling the service from a DSI Rule Agent.

8.6.1 Creating the decision service

In the decision service for this scenario, a decision table decides which course of action to pursue for late passengers based on their loyalty status. Begin the process by opening Rule Designer.

Creating the XOM

Your first task in creating the decision service is to create the Java XOM, from which you generate the BOM. Create a Java project by using Rule Designer (**File** → **New** → **Project** → **Java Project**). Give the project a sensible name. The example uses `rdbook_airline_xom`, and accept the default options.

The XOM class must contain getters, setters, and attributes for the parameters that you want to pass into your ruleflow. Create a class in the `com.ibm.rdbook.airline.rules.xom` package and call it **LatePassengerRequest**. Fill it out as shown in Example 8-25.

Example 8-25 LatePassengerRequest.java

```
package com.ibm.rdbook.airline.rules.xom;

import java.io.Serializable;

@SuppressWarnings("serial")
public class LatePassengerRequest implements Serializable {

    private String loyaltyStatus;
    private int latenessInMinutes;

    public String getLoyaltyStatus() {
        return loyaltyStatus;
    }
}
```

```

    public int getLatenessInMinutes() {
        return latenessInMinutes;
    }

    public void setLoyaltyStatus(String loyaltyStatus) {
        this.loyaltyStatus = loyaltyStatus;
    }

    public void setLatenessInMinutes(int latenessInMinutes) {
        this.latenessInMinutes = latenessInMinutes;
    }
}

```

This case has two parameters to make a decision on:

- ▶ The loyalty status of the passenger in question
- ▶ Lateness of the passenger, which is expressed as an integer number of minutes.

Creating the Rule Project

To create a rule project, from Rule Designer, click **File** → **New** → **Rule Project** and select **Standard Rule Project** under Decision Service Rule Projects (Figure 8-52).

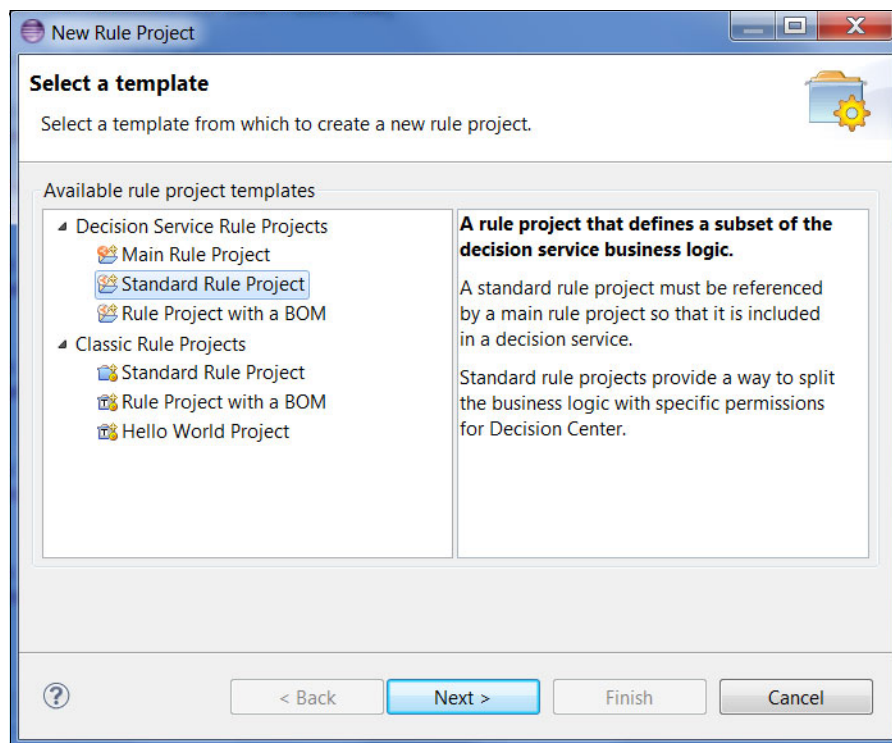


Figure 8-52 Creating a Rule Project

Name it **rdbook_airline_late_passenger** and click **Finish**.

Creating the BOM

Next, create the BOM from the XOM that you created earlier. Right-click the BOM folder under the **rdbook_airline_late_passenger** project and select **New** → **BOM Entry**.

Accept the defaults on the first page of the wizard, then click **Browse XOM** on the next page. Select the project that contains the XOM, then select the class in the tree view (Figure 8-53).

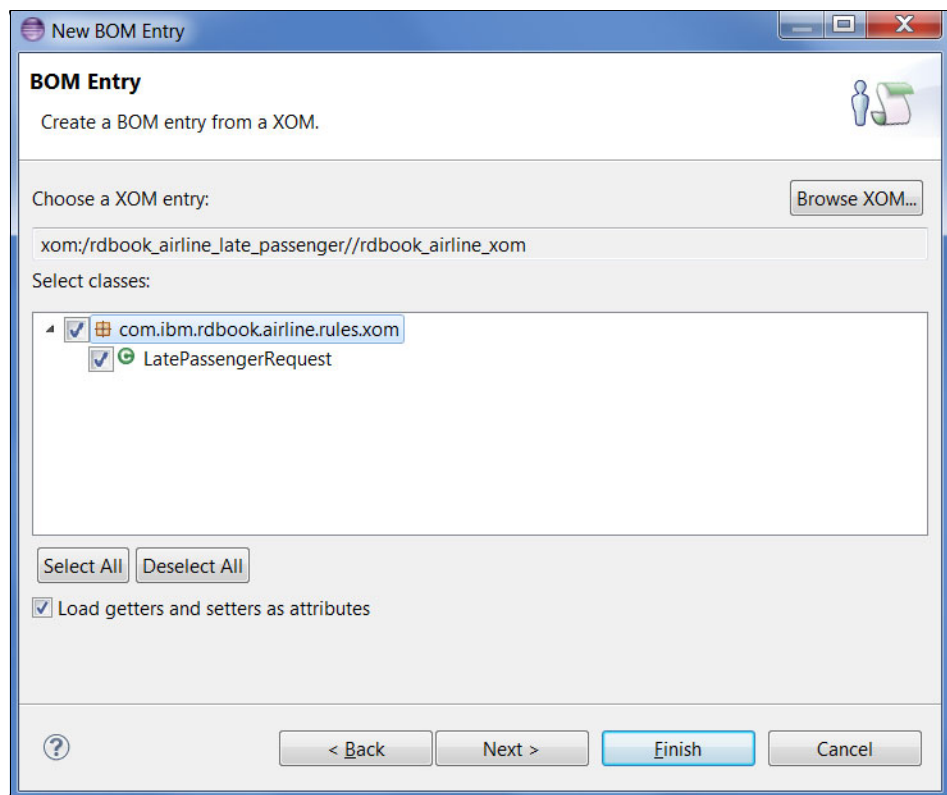


Figure 8-53 Creating the BOM from XOM

Click **Finish** to create the BOM.

Creating a variable set

To create a variable set, right-click the rules folder under the `rdbook_airline_late_passenger` project and click **New** → **Variable Set**. Call the variable set **parameters**, and leave the package name blank. Enter the variables shown in Figure 8-54.

Variable Set: parameters			
Name	Type	Verbalization	Initial Value
latePassengerRequestVar	com.ibm.rdbook.airline.rules.xom.LatePassengerRequest	late passenger request	
decisionCode	java.lang.String	decision code	"INIT"

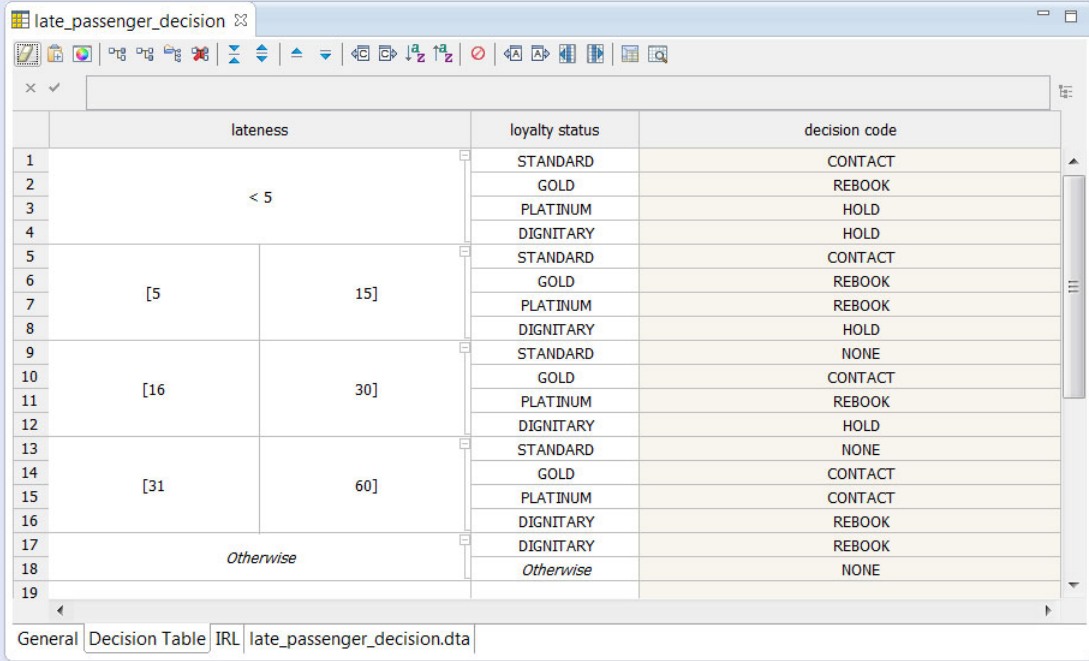
Figure 8-54 Setting up variables for the rule project

In this variable set, define every parameter that you will use in the decision service. “latePassengerRequestVar” is the request, and “decisionCode” is the return value.

For testing, it is useful to initialize the return value to something that you will not set as part of your rule execution, so you know that your rules are being called as you expect.

Creating a decision table

To create a decision table, right-click the rules folder again and select **New** → **Decision Table**. Package the decision table under `latepassenger` and call it `late_passenger_decision`. Fill it out as shown in Figure 8-55.



	lateness	loyalty status	decision code
1		STANDARD	CONTACT
2		GOLD	REBOOK
3	< 5	PLATINUM	HOLD
4		DIGNITARY	HOLD
5		STANDARD	CONTACT
6		GOLD	REBOOK
7	[5 15]	PLATINUM	REBOOK
8		DIGNITARY	HOLD
9		STANDARD	NONE
10		GOLD	CONTACT
11	[16 30]	PLATINUM	REBOOK
12		DIGNITARY	HOLD
13		STANDARD	NONE
14		GOLD	CONTACT
15	[31 60]	PLATINUM	CONTACT
16		DIGNITARY	REBOOK
17		DIGNITARY	REBOOK
18	Otherwise	Otherwise	NONE
19			

Figure 8-55 Decision table contents for the scenario

For more information about configuring a decision table, see the IBM Knowledge Center at the following web address:

http://www.ibm.com/support/knowledgecenter/SSQP76_8.7.1/com.ibm.odm.dserver.rules.designer.author/authoring_topics/tpc_dtables_intro.html?lang=en

Creating a ruleflow

To create a ruleflow, right-click the rules folder and select **New** → **Ruleflow**. Package the ruleflow under `latepassenger` and call the ruleflow `main_flow`. Add a start and end node, with a rule task in the middle, as shown in Figure 8-56.

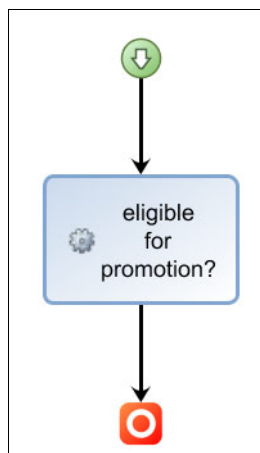


Figure 8-56 Ruleflow for the scenario

Finally, in the Properties window for the rule task, give it a descriptive name (such as “eligible for promotion?”) on the Rule Task tab (Figure 8-57).

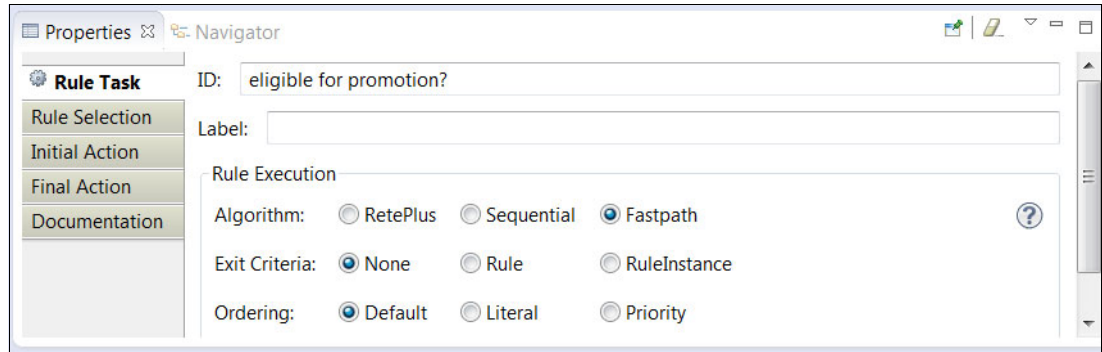


Figure 8-57 Rule Task properties tab for the ruleflow

Select the decision table that was created before on the Rule Selection tab (Figure 8-58).

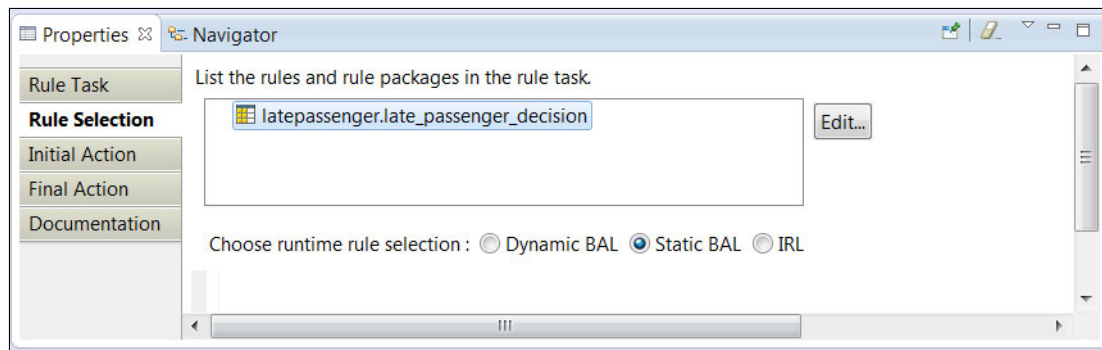


Figure 8-58 Rule Selection properties tab for the ruleflow

For more information about setting up the ruleflow, see the IBM Knowledge Center at the following web address:

http://www.ibm.com/support/knowledgecenter/SSQP76_8.7.1/com.ibm.odm.dserver.rules.designer.dev/shared_ruleflow_topics/con_ruleflow.html?lang=en

Creating the decision operation

To create a decision operation, right-click the deployment folder and select **New** → **Decision Operation**. Give it a name like **make_late_passenger_decision**. On the Overview tab, select the ruleflow that you created as the main ruleflow (Figure 8-59).

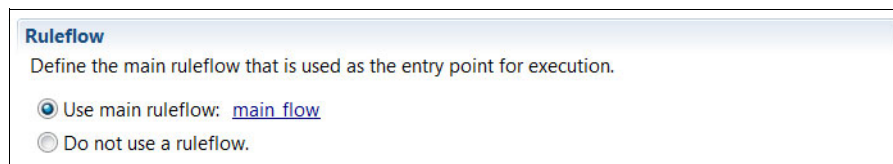


Figure 8-59 Decision Operation setup - choosing the correct ruleflow

Move to the Signature tab and add the relevant parameters as Input and Output parameters (Figure 8-60).

Decision Operation Signature - make_late_passenger_decision

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

- rdbook_airline_late_passenger
 - parameters
 - latePassengerRequestVar
 - decisionCode

Input Parameters
Define the parameters required to call the execution.

Parameter name	Verbalization	Type	Initial Value
latePassengerRequestVar	late passenger request	com.ibm.rdbook.airline.rules.xom.LatePassengerRequest	

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

Parameter name	Verbalization	Type	Initial Value

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

Parameter name	Verbalization	Type	Initial Value
decisionCode	decision code	java.lang.String	"INIT"

Figure 8-60 Decision Operation setup - configuring the signature

Creating the Rule Execution Server connection

In the Rule Execution Server Connections view in Rule Designer, right-click the empty list and select **New Server**. Complete the form with the correct information for your Rule Execution Server (RES) connections.

Creating the deployment configuration

To create a deployment configuration, right-click the deployment folder and select **New** → **Deployment Configuration**. Call it **rdbook_airline_app**. On the Decision Operations tab, click **Add decision operation** under Configured Decision Operations and select the decision operation that was created (Figure 8-61).

rdbook_airline_app

Deployment, Decision Operations - rdbook_airline_app

Configured Decision Operations
Select and configure decision operations for RuleApp deployment:

+ x

- Missing references
 - make_late_passenger_decision

Figure 8-61 Selecting the Decision Operation for deployment

On the Target Servers tab, click **Add target server** and select the RES connection that was previously configured (Figure 8-62).

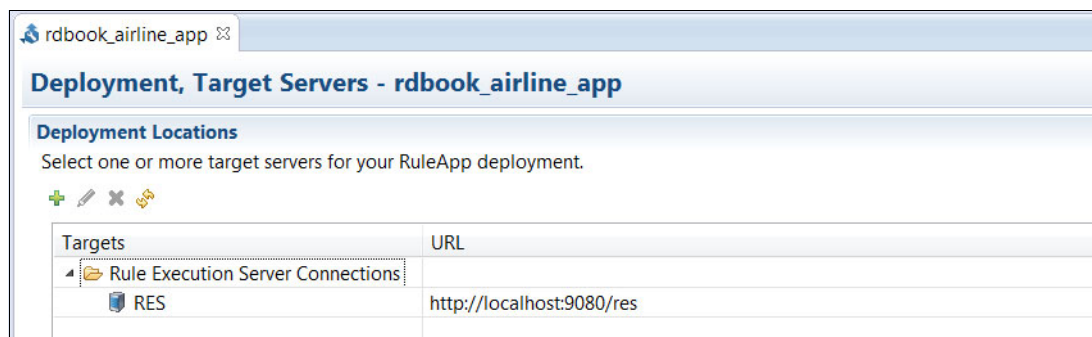


Figure 8-62 Setting up the target server for deployment

Deploy and test the rule project

For deploying and testing the rule project, right-click the **rdbook_airline_late_passenger** project and select **Rule Execution Server** → **Deploy**. Finish the process by accepting the defaults. Your decision service is deployed successfully.

To test the rule project, you need to access the RES console. For one running on your localhost with default port configuration, the URL of the RES console is:

`http://localhost:9080/res/protected/home.jsf`

Click **Explorer** → **rdbook_airline_app** → **make_late_passenger_decision** → **Test Ruleset**. Set up some test data, click **Execute**, and make sure that you get the result that you expect.

8.6.2 Creating the OSGi service

Now you have your decision service, you can look at how you want to start it from DSI. The method used in this example is to perform an HTTP POST to the REST API in Insight Designer.

Creating the OSGi project

The first step is to create the OSGi project, in this example called **res_invoker_service**. If you do not know how to do this, the instructions are in Appendix A.1.2, “Creating the OSGi bundle project” on page 232.

Next, create your interface in the `com.ibm.rdbook.airline.service.api` package with the code in Example 8-26.

Example 8-26 RESInvoker.java

```
package com.ibm.rdbook.airline.service.api;

public interface RESInvoker {

    public String getActionCode(String loyaltyStatus, int latenessInMinutes);

}
```

This method returns the action code for a passenger's loyalty status and lateness. Create the service implementation class `RESInvokerImpl` in the `com.ibm.rdbook.airline.service.impl` project.

For the example implementation of this callout, use the Apache HTTP client and the `javax.json` package. You can use a package manager such as Maven to include these libraries in your OSGi service project, but for simplicity, download the packages and include them as a required library.

Add a `lib` folder to the project (**File** → **New** → **Folder**) and copy the libraries into the folder (Figure 8-63).

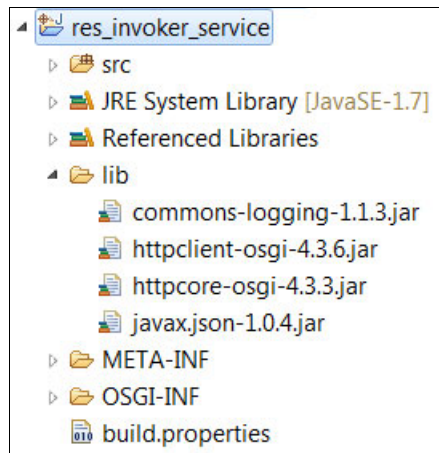


Figure 8-63 Required libraries for the Apache HTTP client used in the scenario

Open **META-INF/MANIFEST.MF** and set up the Dependencies tab as shown in Figure 8-64.

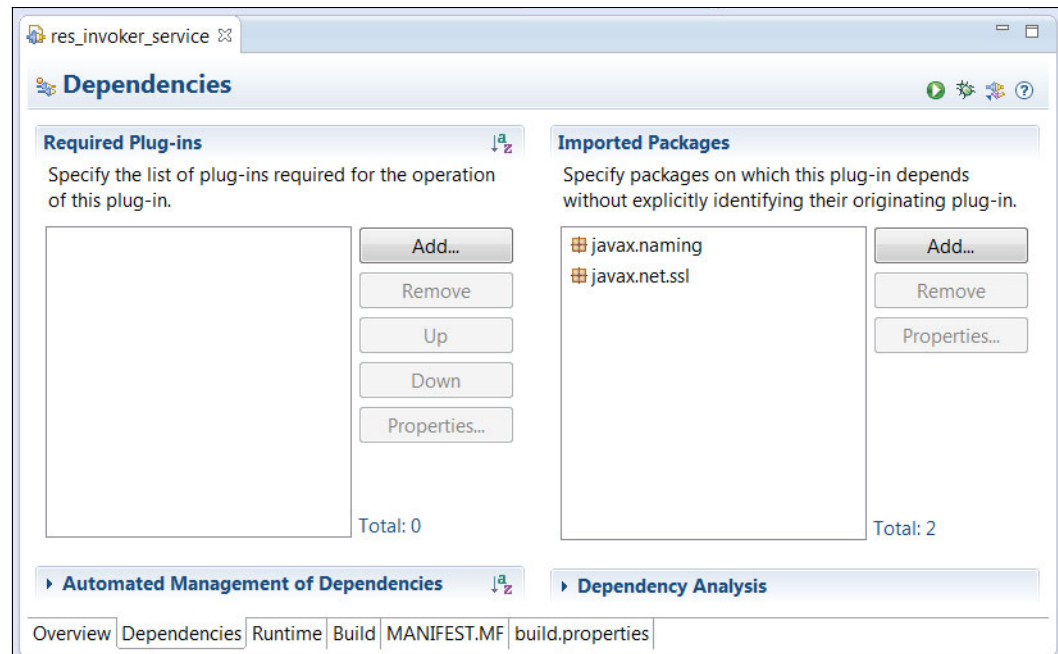


Figure 8-64 Setting up the dependencies for the service

Set up the Build tab as shown in Figure 8-65.

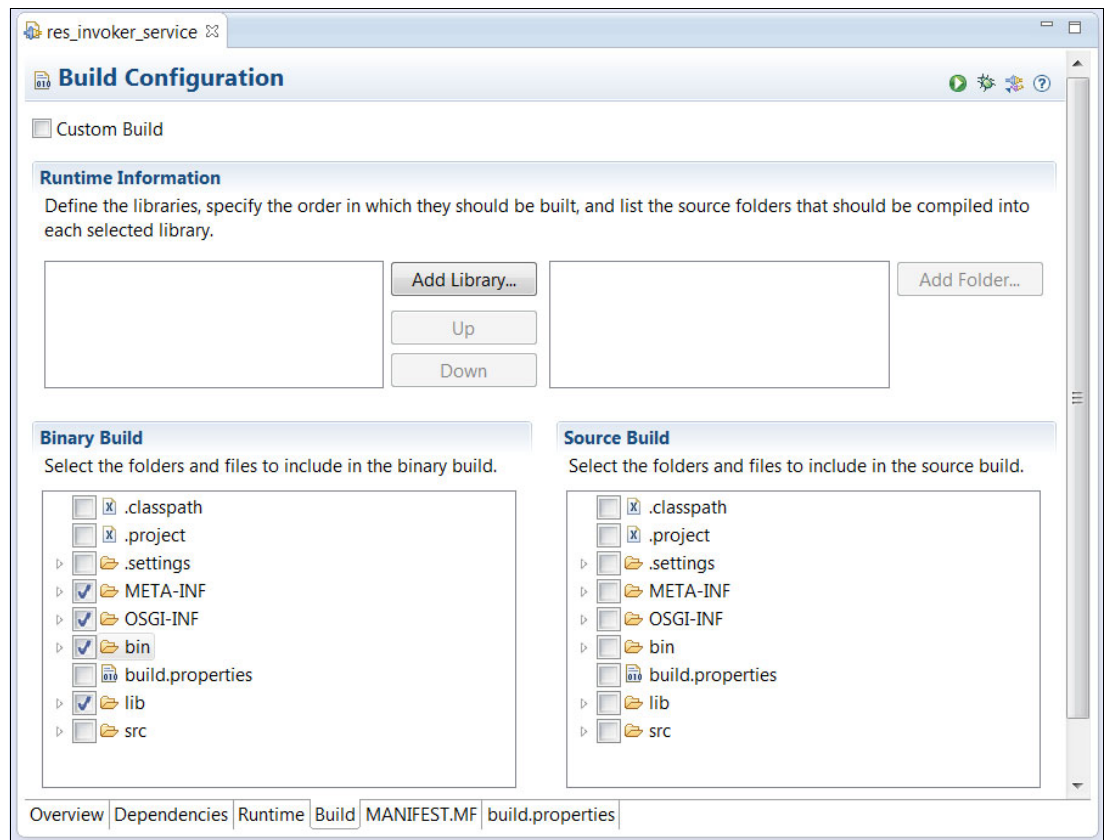


Figure 8-65 Setting up the build configuration for the service

Set up the Runtime tab as shown in Figure 8-66.

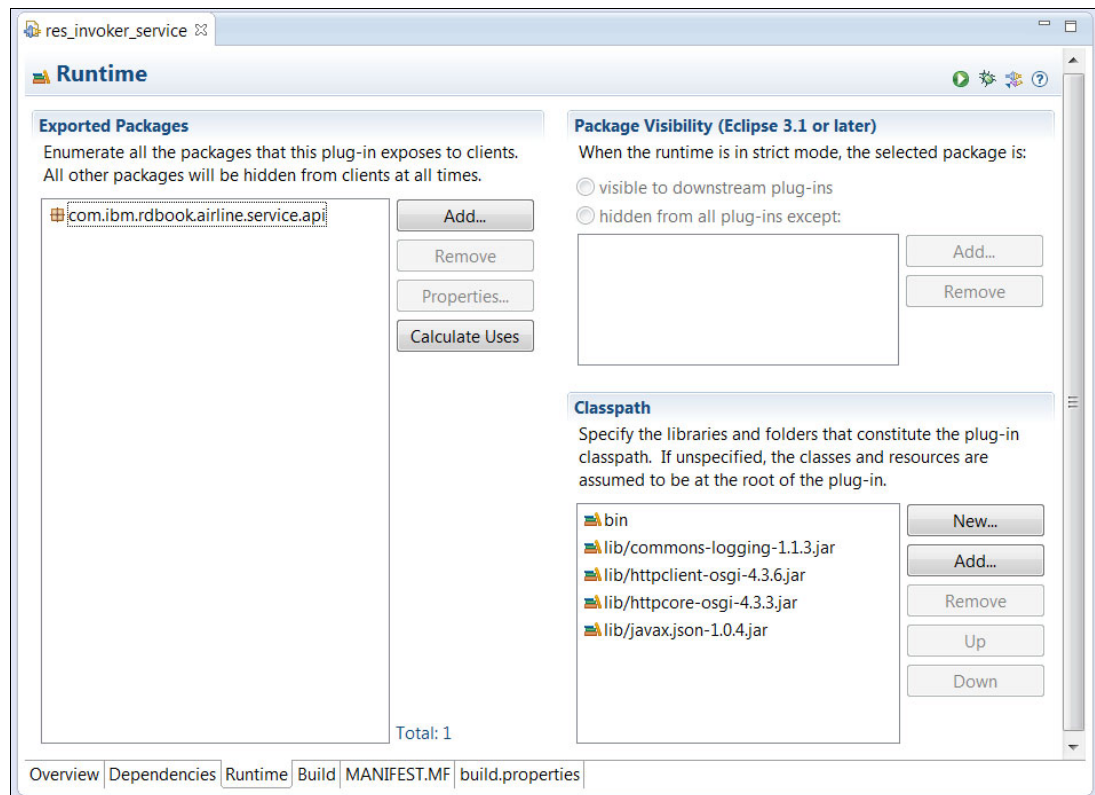


Figure 8-66 Setting up the runtime configuration for the service

The final task to make these libraries available is to set up the project class path. Right-click the service project and select **Properties** → **Java Build Path** → **Add Jars**. Then select the required jars from your lib directory as shown in Figure 8-67.

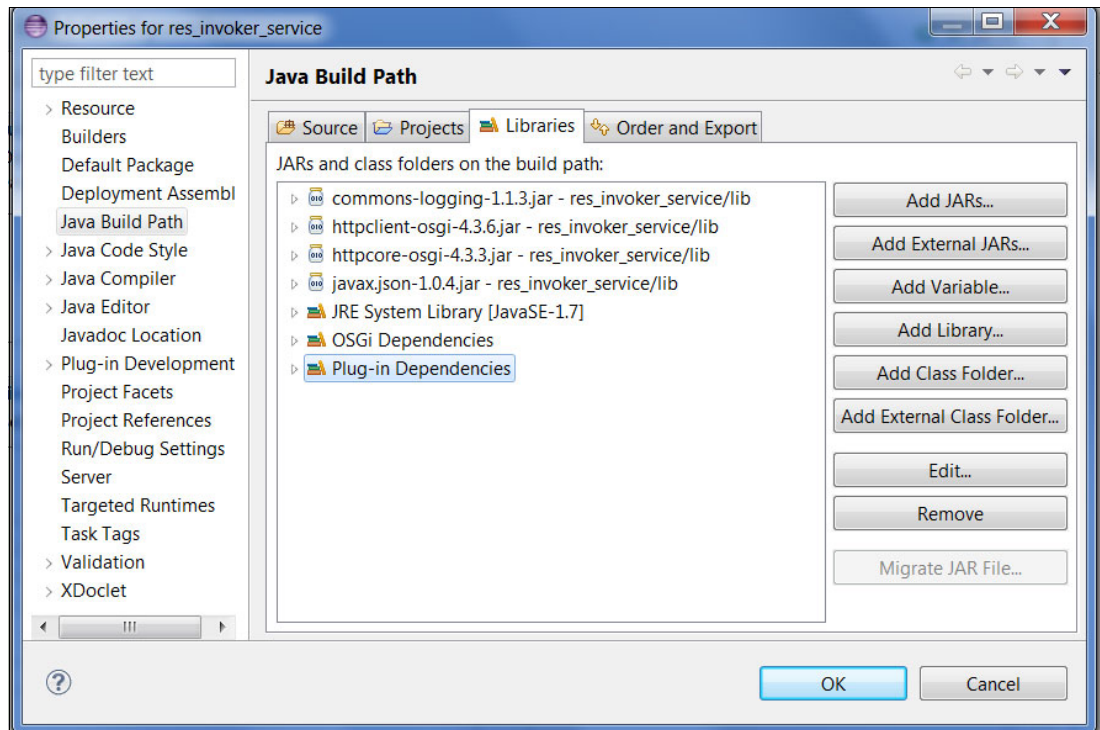


Figure 8-67 Configuring the service project class path

Now add a service to the blueprint.xml as shown in Figure 8-68. See Appendix A.1.5, “Configuring the blueprint” on page 238 for more details.

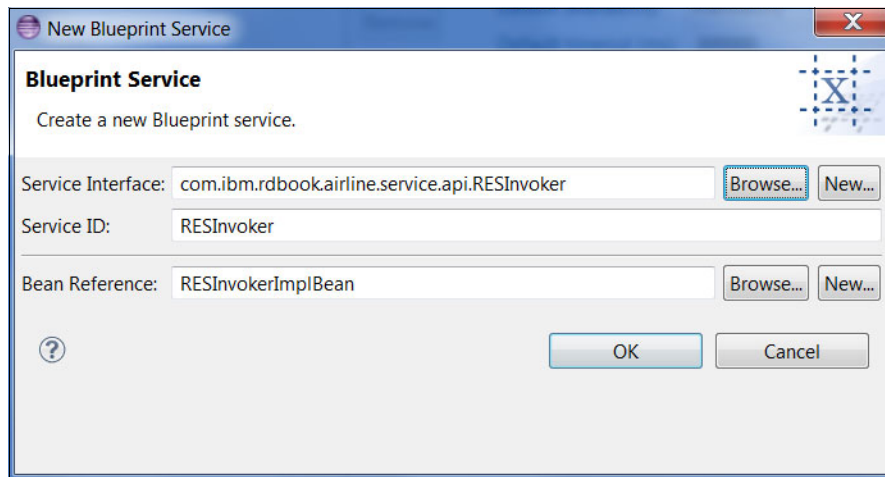


Figure 8-68 Configuring the blueprint service

Now you are ready to write the service implementation. Example 8-27 shows the code.

Example 8-27 RESInvokerImpl.java

```
package com.ibm.rdbook.airline.service.impl;

import java.io.IOException;
```

```

import java.io.StringReader;
import java.io.UnsupportedEncodingException;

import javax.json.Json;
import javax.json.JsonBuilderFactory;
import javax.json.JsonObject;
import javax.json.JsonReader;

import com.ibm.rdbook.airline.service.api.RESInvoker;
import org.apache.http.HttpStatus;
import org.apache.http.client.methods.CloseableHttpResponse;
import org.apache.http.client.methods.HttpPost;
import org.apache.http.entity.StringEntity;
import org.apache.http.impl.client.CloseableHttpClient;
import org.apache.http.impl.client.HttpClients;
import org.apache.http.impl.conn.PoolingHttpClientConnectionManager;
import org.apache.http.util.EntityUtils;

public class RESInvokerImpl implements RESInvoker {
    private String resURL =
"http://localhost:9080/DecisionService/rest/v1/rdbook_airline_app/make_late_passenger_decis
ion";
    private CloseableHttpClient client;

    @Override
    public String getActionCode(String loyaltyStatus, int latenessInMinutes) {
        // Set up payload
        JsonBuilderFactory factory = Json.createBuilderFactory(null);
        JsonObject payload = factory.createObjectBuilder()
            .add("loyaltyStatus", loyaltyStatus)
            .add("latenessInMinutes", latenessInMinutes)
            .build();

        // Create a client if necessary
        if (null == client)
            client = getClient();

        String result = null;

        try {
            // Set up HTTP POST
            HttpPost httpPost = new HttpPost(resURL);
            httpPost.addHeader("Content-Type", "application/json");
            httpPost.setEntity(new StringEntity(payload.toString()));

            // Execute HTTP POST
            CloseableHttpResponse response = client.execute(httpPost);

            // Check response is 200 OK
            if (response.getStatusLine().getStatusCode() != HttpStatus.SC_OK) {
                throw new IOException("HTTP status code was " +
response.getStatusLine().getStatusCode());
            } else {
                result = EntityUtils.toString(response.getEntity());
            }

            // Close request
            response.close();
        } catch (UnsupportedEncodingException e) {
            // Something is wrong with our request

```

```

        e.printStackTrace();
        return "ERROR";
    } catch (IOException e) {
        // Something is wrong with our connection
        e.printStackTrace();
        return "ERROR";
    }

    // Parse the result
    JsonReader reader = Json.createReader(new StringReader(result));
    JsonObject jsonResponse = reader.readObject();
    return jsonResponse.getString("decisionCode");
}

private CloseableHttpClient getClient() {
    // Set up connection pooling
    PoolingHttpClientConnectionManager cm = new PoolingHttpClientConnectionManager();
    cm.setMaxTotal(10);
    cm.setDefaultMaxPerRoute(10);

    // Build and return the client
    return HttpClients.custom().setConnectionManager(cm).build();
}
}

```

The `resURL` property contains the URL for the decision service that you created and deployed in 8.6.1, “Creating the decision service” on page 165.

To retrieve the URL for your decision service, complete these steps:

1. Access the RES console and navigate to your ruleset (**Explorer** → **rdbook_airline_app** → **make_late_passenger_decision**).
2. Click the **Retrieve HTDS Description** file.
3. Select the REST service protocol, select **Latest ruleset version** and **Latest RuleApp version** (Figure 8-69), and click **View**.

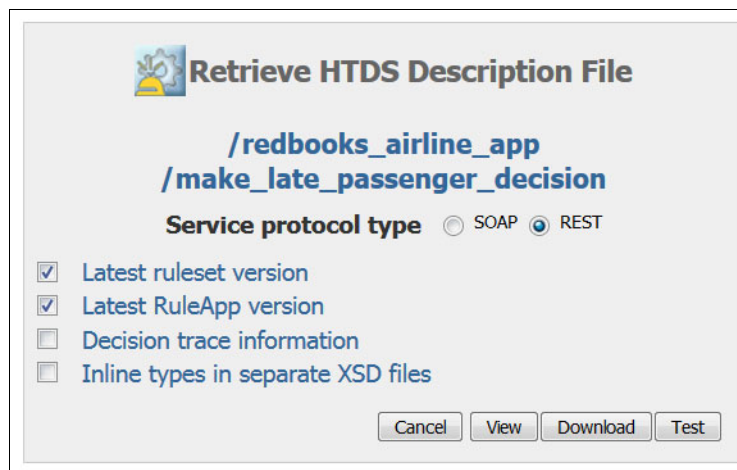


Figure 8-69 Settings to retrieve HTDS containing the URL for the decision service

The URL under the `<resources base="http://...">` tag is the correct one (see Example 8-28).

Example 8-28 Example HTDS description with decision service URL location highlighted

```
<application xsi:schemaLocation="http://wadl.dev.java.net/2009/02 wadl.xsd">
  <doc title="make_late_passenger_decision"/>
  <grammars>
    <include
href="rdbooks_airline_appMake_late_passenger_decisionParameters.xsd"/>
    </grammars>
    <resources
base="http://localhost:9080/DecisionService/rest/v1/rdbook_airline_app/make_late_p
assenger_decision">
      <resource path="">
        <method name="POST">
          <doc title="execute"/>
          <request>
...

```

The URL is set statically, so if your deployment configuration changes, you must rebuild and redeploy the whole solution. You can optionally set the URL as a property in your `server.xml`, which allows you to update your configuration at run time. The following section explains how to set up the `resURL` as a property.

Setting the `resURL` as a property in `server.xml`

The example uses a JNDI entry lookup to obtain the `resURL` property from `server.xml`. First, set up your `server.xml`. If you are using the default development server, the file is in `${ODM Installation Directory}/runtime/wlp/usr/servers/cisDev`.

Open the appropriate `server.xml` file for your deployment and add the lines from Example 8-29.

Example 8-29 server.xml

```
...
<featureManager>
  <feature>jndi-1.0</feature>
  ...
</featureManager>

<jndiEntry
value="http://localhost:9080/DecisionService/rest/v1/rdbooks_airline_app/make_late
_passenger_decision" jndiName="env/resURL"></jndiEntry>
...

```

The final task is to modify your service implementation class to add an initial context lookup for the URL, as shown in Example 8-30.

Example 8-30 RESInvokerImpl.java

```
...
import javax.annotation.InitialContext;
...
public class RESInvokerImpl implements RESInvoker {
  private String resURL;

```

```

private CloseableHttpClient client;

@Override
public String getActionCode(String loyaltyStatus, int latenessInMinutes) {
    // Look up the URL if necessary
    if (null == resURL) {
        try {
            resURL = (String) new InitialContext().lookup("env/resURL");
        } catch (NamingException e) {
            e.printStackTrace();
            return "ERROR: NAMING";
        }
    }
}
...

```

Adding the OSGi service to the solution

For the OSGi service to be exported with the solution, you must include the OSGi service as a referenced project. Right-click the **rdbook_airline solution** project and select **Properties** → **Project References**. Select the service project (**res_invoker_service**) in this list and click **OK**.

8.6.3 Mapping the OSGi service to a BOM entry

Now that your OSGi service is implemented, you need to map it to a BOM entry. Following the instructions in Appendix A.2, “XOM to BOM mapping” on page 239, create a new rule project, calling it **res_invoker_bom**, referencing the OSGi bundle project as the XOM.

Create the BOM Entry referring to the service interface, and perform the necessary configuration (OSGi.service property, make the `getActionCode` method static, change the Rule Engine to Decision Engine).

Finally, create the verbalization (Figure 8-70).

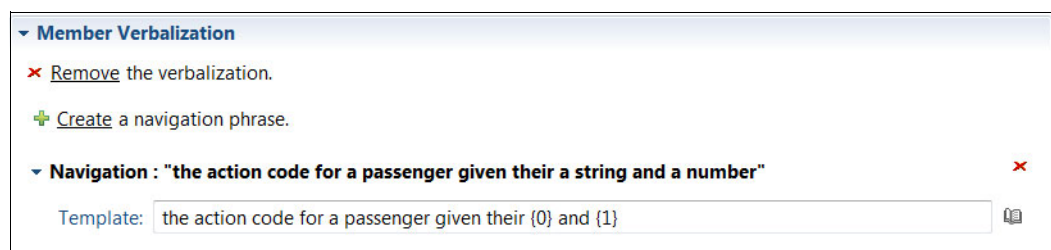


Figure 8-70 Verbalization for the OSGi service call

8.6.4 Starting the service in a rule agent

To use service verbalization, create a rule agent. Click **File** → **New** → **Rule Agent Project** and name the rule agent **late_passenger_agent**.

Complete the agent descriptor with the contents of Example 8-31. Use this agent to decide what to do with the late passengers by processing the late passenger events generated by the predictive scoring agent that you created in 8.5.4, “Predictive analytics agent implementation” on page 159.

Example 8-31 agent.adsc for the late_passenger_agent

```
'late_passenger_agent' is an agent related to a booking,
processing events :
- late passenger, where this booking comes from the booking of this late passenger
```

Right-click the rules folder under the **late_passenger_agent** project and select **New** → **Action Rule**. Place the rule in the package `com.ibm.rdbooks.airline.agent` and call it **make decision**.

Complete it with the contents of Example 8-32.

Example 8-32 make decision rule in late_passenger_agent

```
when a late passenger occurs
definitions
    set 'the passenger' to the passenger of 'the booking';
    set 'loyalty tier' to the loyalty tier of 'the passenger';
    set 'lateness' to the duration of the lateness of this late passenger in
minutes;
    set 'the action code' to the action code for a passenger given their 'loyalty
tier' and 'lateness';
then emit a new created exception where
    the booking is 'the booking',
    the action code is 'the action code';
```

In 8.7, “Taking action example” on page 180 you process the created exception events created here and start taking some appropriate actions.

8.7 Taking action example

This section extends the airline scenario to provide some concrete examples of how situations that are detected by Decision Server Insights can trigger an automatic action in an external system.

The scenario requires the following specific actions under different circumstances:

- ▶ Notify gate staff that a disabled passenger is running late and needs assistance.
- ▶ Automatically rebook the passenger onto a different flight and allow issue of the new boarding pass.
- ▶ Send a request to a service agent in the ticket office to call the passenger to arrange for rebooking manually.

From a technology perspective, assume that Rdbooks Airline has an existing notification service exposed on their ESB that delivers notifications to desktop computers at airport desks and mobile devices of their staff. Also assume that Rdbooks Airlines has an existing business process that is implemented in a business process management system (BPMS) to handle rebooking.

The following description uses implementations of the service and business process in the IBM Integration Bus and IBM Business Process Manager platforms.

8.7.1 Decision Server Insights outbound integration with IBM Integration Bus

IBM Integration Bus makes it easy for developers to connect applications together. It converts data formats and protocols, and can expose backend services and APIs to allow for innovation and reuse. Many hundreds of organizations rely on it for processing and moving business critical data quickly and reliably. For developers, it is quick to use yet flexible enough to fit in with existing skills and tools.

This example uses IBM Integration Bus to provide protocol and data transformations that are required to connect to a third-party notification service. This service cannot be directly started by DSI. The third-party notification service is defined by a WSDL and allows invocation by using a SOAP/HTTP request.

Download IBM Integration Bus: If you want to try the examples described in this book, you can download IBM Integration Bus V10 Developer Edition from the IBM Integration Bus product page on the IBM website:

<http://www.ibm.com/software/products/en/ibm-integration-bus>

To build this integration between DSI and IBM Integration Bus, complete these high-level steps:

1. DSI: Extend the connectivity definition of the solution.
2. DSI: Add the HTTP connectivity features to the server configuration.
3. DSI: Export the event types as an XML schema.
4. IBM Integration Bus: Build the message flow.
5. Deploy the IBM Integration Bus application.
6. Deploy the DSI solution.

The examples here assume that you already have installations of DSI and IBM Integration Bus, and that you have a DSI solution that you want to extend.

Decision Server Insights connectivity definition

The connectivity definition file (called `connectivity.cdef` in the Connectivity Definitions folder of the solution project) specifies the inbound and outbound bindings and endpoints that process incoming and outgoing events. You can create a connectivity definition file by right-clicking your solution project in Insight Designer and selecting **New** → **Connectivity Definition**. You might also need to create the Connectivity Definitions folder if you have not been following this chapter end-to-end.

Inside the connectivity definition file, provide the description of your connectivity as shown in Example 8-33.

Example 8-33 Sample connectivity definition for outbound HTTP events

```
define outbound binding 'notificationBinding'
  using
    message format application/xml ,
    protocol HTTP ,
    delivering events :
      - mobile alert .

define outbound HTTP endpoint 'iibNotificationEndpoint'
```

```
using
    binding 'notificationBinding' ,
    url "http://localhost:7080/soi/notification".
```

Notice in particular the list of events that are associated with the binding. In this case only mobile alert events are sent. You must also replace the host name and port number within the endpoint definition with that of your IBM Integration Bus server. Note also the URI used within the endpoint definition (/soi/notification). You will specify this path within IBM Integration Bus in a subsequent step.

Adding connectivity features to your DSI server configuration

DSI packages its connectivity functionality as the WebSphere Liberty Profile features, which can be enabled or disabled as necessary. The definition of features is made in the <DSIInstallDir>/runtime/wlp/usr/servers/<server_name>/server.xml file.

The airline scenario does not require the inbound HTTP feature (as all of the inbound events come in through JMS). However, Example 8-34 lists both options for your reference. The features are added within a <features> element of server.xml, alongside the other features that are already enabled.

Example 8-34 Additional features that are required for HTTP inbound and outbound connectivity

```
<feature>ia:iaConnectivityInboundHTTP-8.7.1</feature>
<feature>ia:iaConnectivityOutboundHTTP-8.7.1</feature>
```

Export the DSI event types as an XML schema

Given that you have already defined your events within DSI, you can export those definitions from Insight Designer and then import them directly into the IBM Integration Bus Integration Toolkit (the IBM Integration Bus developer tools). This helps ensure that the data model used in IBM Integration Bus interfaces exactly matches the event messages that come from DSI.

To export the DSI event types from Insight Designer, right-click your solution project and choose **Export**. Then select **Insight Designer** → **Event Types to XML Schema** → **Next** before specifying a location to store them and clicking **Finish**.

Build the message flow in IBM Integration Bus

Complete the following steps to build the message flow in IBM Integration Bus:

1. From within IBM Integration Bus Integration Toolkit, create a integration application by clicking **New** → **Application**.
2. Import your exported schema by clicking **File** → **Import** → **General** → **File System**.
3. Navigate to the directory where you have saved your DSI events schema and check the check box in the right pane next to the file. Click **Finish** to complete.

At this stage, import your web service's WSDL file.

- Now create a message flow for the integration application (**New** → **Message Flow**).

Using the nodes in the palette of the flow editor, build the simple message flow as shown in Figure 8-71.

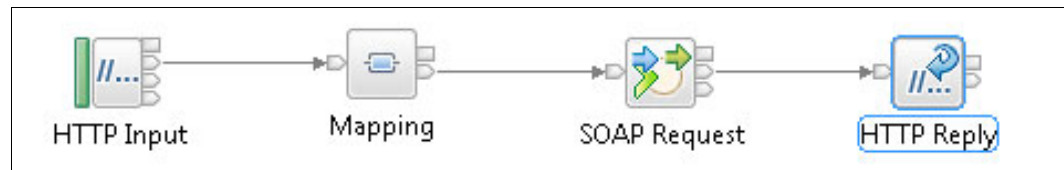


Figure 8-71 A simple message flow for an HTTP service.

- Configure each of the nodes in turn.

The HTTP input node defines the external HTTP endpoint that can be called by DSI. In the Properties tab, Basic section (Figure 8-72), specify the URI to match the definition that you made in “Decision Server Insights connectivity definition” on page 181 in the HTTP Input node properties window.

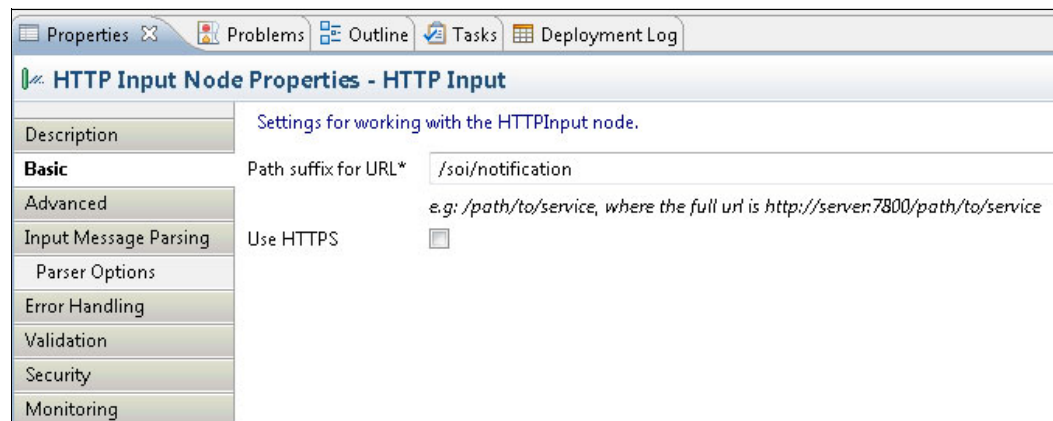


Figure 8-72 Specifying the path suffix for an HTTP Input Node in IBM Integration Bus

In the Input Message Parsing section (Figure 8-73), change the message domain to XMLNSC to specify that the incoming message is in an XML format.

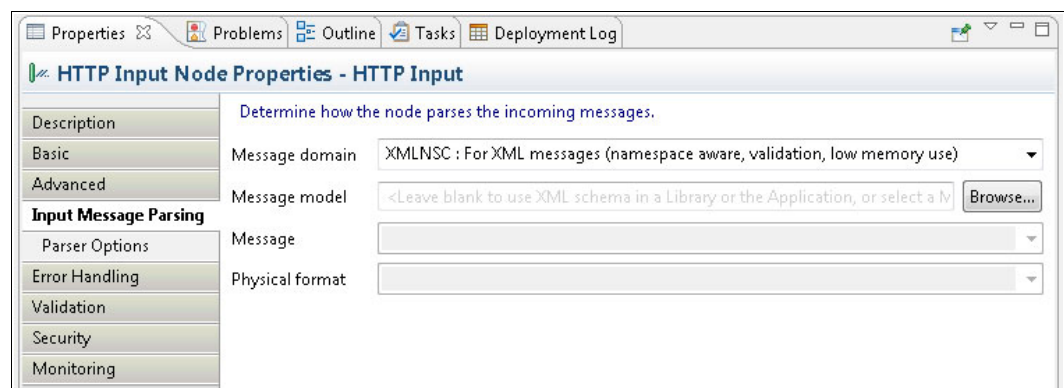


Figure 8-73 Specifying the message domain for an HTTP Input Node in IBM Integration Bus

- Double-clicking the mapping node starts the New Message Map wizard that guides you through the initial configuration for a graphical mapping tool. This allows you to move data from the input message (from DSI) to the format that your existing SOAP web-service requires.

7. In the second step shown in Figure 8-74, select the input and output data models. In this example, the models are Mobile Alert (coming from DSI) on the left, and message (going to the notification service) on the right.

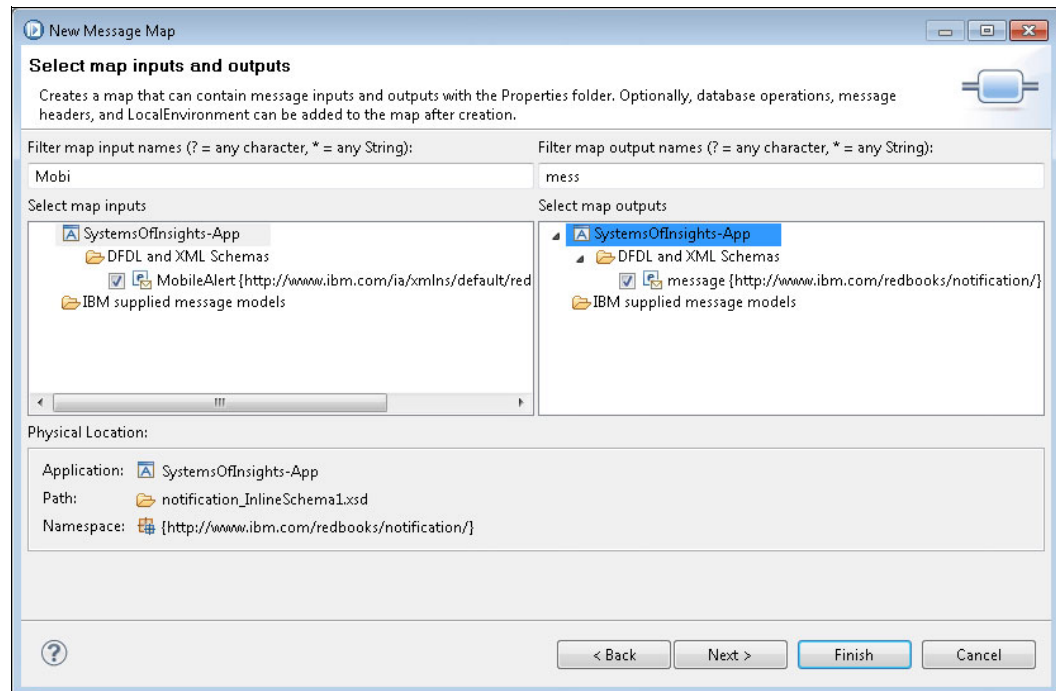


Figure 8-74 Selecting the map inputs and outputs

When inside the map, drag from the yellow handle next to messageContent onto the value attribute of message to perform the assignment as shown in Figure 8-75.

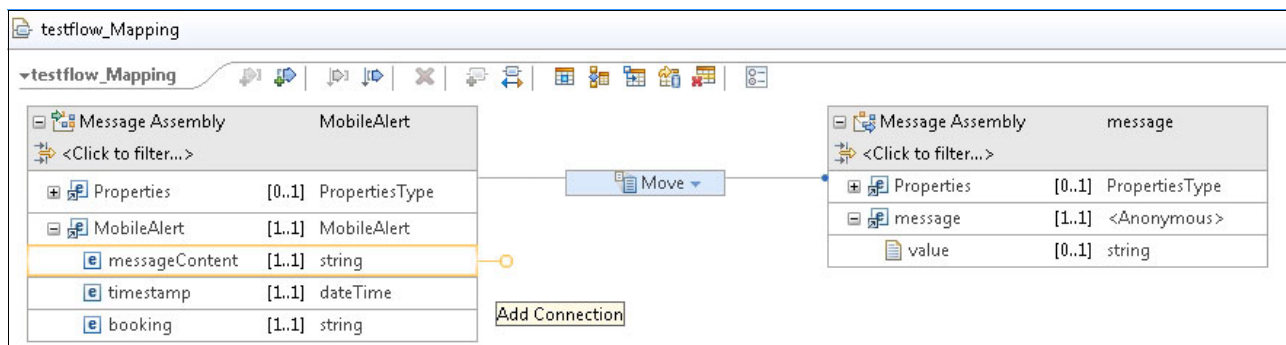


Figure 8-75 Mapping from input to output data structures.

The complete move is shown in Figure 8-76.

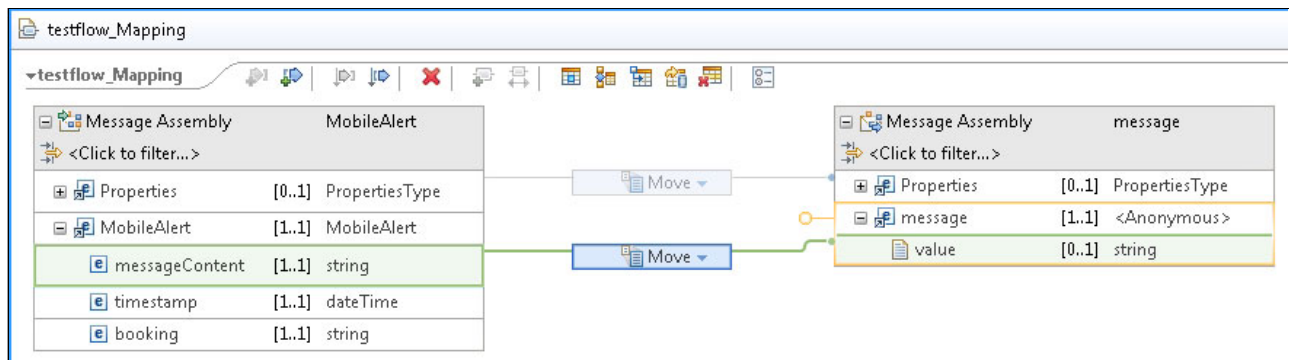


Figure 8-76 Completed mapping from input to output data structures

Deploy the IBM Integration Bus application

To deploy the finished IBM Integration Bus application, drag the Integration Application from the Application Development view and drop it onto the Integration Server (called default in Figure 8-77) in the Integration Node view.

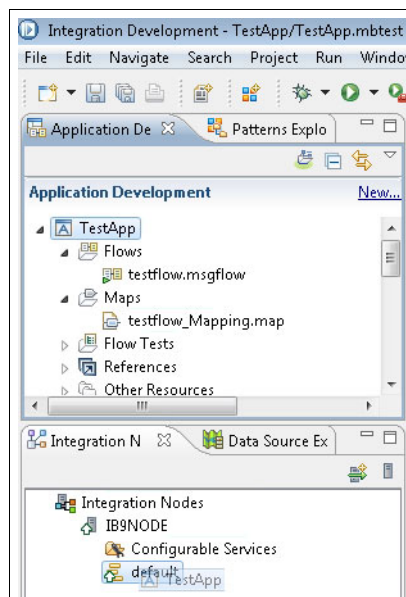


Figure 8-77 Deploying an Integration Application

Deploy the DSI solution

Deploy your solution according to the instructions in the IBM Knowledge Center at the following web address:

http://www.ibm.com/support/knowledgecenter/SSQP76_8.7.1/com.ibm.odm.itoa.deploy/to pics/odm_itoa_deploy.html?lang=en

You do not need to specify a URL override for the `iibNotificationEndpoint` because you have already provided an appropriate value in the solution's connectivity definition file.

8.7.2 Decision Server Insights outbound integration with IBM Business Process Manager (IBM BPM)

IBM BPM is a comprehensive business-process management platform that provides visibility and insight into the execution of business processes. IBM BPM handles the design, execution, monitoring, and optimization of business processes, based on different styles of work: Integration-centric, human-centric, prescriptive, and investigative-type work patterns.

In the airline scenario, you are required to start a business process that realizes the change booking process. To build this scenario, take an existing IBM BPM Process Application that has a manual start (triggered by an agent at the airline's ticket desk) and modify it to be triggered automatically under the situation detected by DSI. IBM BPM provides you with several alternative approaches for starting a process based on external stimuli.

This example uses asynchronous point-to-point messaging with XML over JMS. However, you are encouraged to evaluate each option's applicability to your own scenario, in particular, if you are using event sources and destinations other than IBM operational decision management (ODM) and IBM BPM.

The following list outlines some inbound integration options for IBM BPM:

- ▶ XML over JMS: Send a message directly to the IBM BPM event manager for asynchronous processing.
- ▶ Web Service (REST): The IBM BPM REST API provides two options for starting a process:
 - POST (start) requires an HTTP request Content-Type of `application/x-www-form-urlencoded` and a URL-encoded JSON object. For more information about POST (start) method, see the IBM Knowledge Center at:
http://www.ibm.com/support/knowledgecenter/SSFPJS_8.5.6/com.ibm.wbpm.ref.doc/rest/bpmrest/rest_bpm_wle_v1_process_post_start.htm
 - POST (sendMessage) requires a URL-encoded XML parameter. This is the same message format as used in the XML over JMS example. For more information about POST (sendMessage) method, see the IBM Knowledge Center at:
http://www.ibm.com/support/knowledgecenter/SSFPJS_8.5.6/com.ibm.wbpm.ref.doc/rest/bpmrest/rest_bpm_wle_v1_process_post_sendmessage.htm
- ▶ Web Service (SOAP): Binding a web-service artifact to an undercover agent (UCA) in Process Designer allows you to send SOAP-compliant HTTP requests.
- ▶ BPM Advanced Adapters: The advanced integration capabilities allow you to interact with IBM BPM directly using various protocols. This also allows further message manipulation before starting a business process. This option is only available in IBM Business Process Manager Advanced.
- ▶ External mediation: Finally, you have the option of using some intermediate mediation step to perform protocol and message transformations. Commonly, this can be in IBM Integration Bus or message-driven bean (MDB).

The following developerWorks article gives an example that uses IBM Business Process Manager as the destination for events from ODM DSI with a separate mediation layer (first using an MDB and then with the IBM BPM SCA implementation).

<https://developer.ibm.com/odm/docs/solutions/odm-and-ibm-business-process-manager/odm-advanced-decision-server-insights-bpm/>

For more information about generic integration patterns for ODM DSI, see Chapter 9.

This section details the steps that are required to start a process in IBM BPM on receipt of an event sent from DSI. In summary, you take the following steps:

1. Configure an IBM BPM process application for event-based interactions.
2. Extend a DSI Solution's connectivity definition with outbound JMS bindings and endpoints.
3. Apply a transformation to the DSI event.

Configuring an IBM BPM process application

This section describes the steps to configure an IBM Business Process Manager application.

IBM Business Process Manager on Cloud trial: IBM Business Process Manager on Cloud provides a full lifecycle IBM BPM environment including development, test, and production. It also provides tools and run time for process design, execution, monitoring, and optimization. IBM BPM on Cloud is available for you to try at no cost from the following webpage:

<http://www.ibm.com/software/products/en/business-process-manager-cloud>

Figure 8-78 introduces the business process in a Business Process Model and Notation (BPMN) diagram. This diagram includes activities that define tasks that are assigned to user roles by their placement in a swimlane. The diagram also includes several system tasks that define calls to update systems of record or start enterprise services.

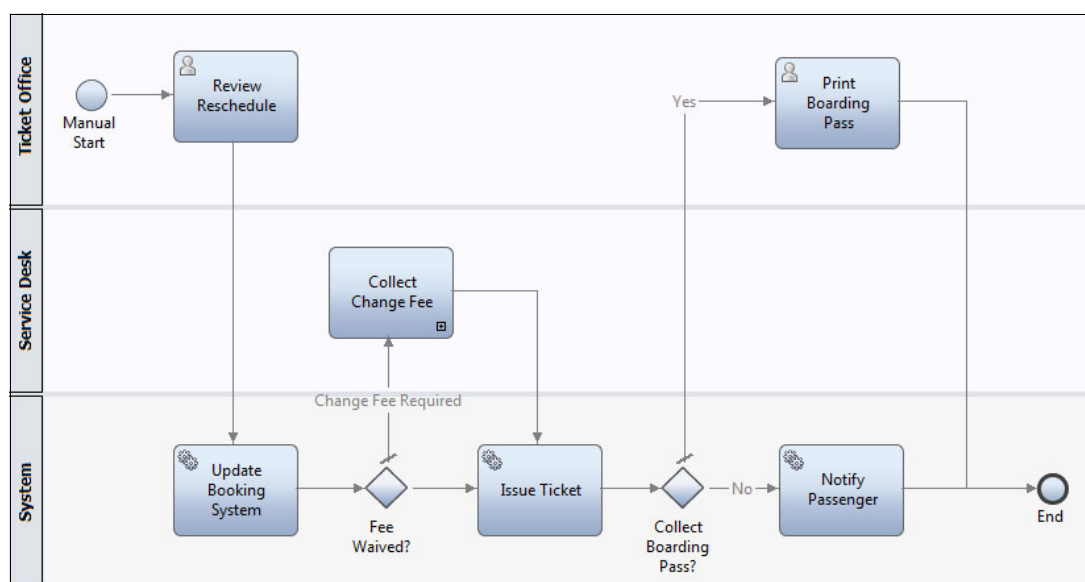


Figure 8-78 The manually triggered ticket rebooking process.

Using the Process Designer tool, modify the process diagram to reflect the changes. The new diagram follows the Process Start pattern defined in “Feedback from business processes” on page 115. In addition to changing the standard manual Start Event to a Message Start Event, also add a Message Start in the system swimlane. This configuration allows you to reuse the same process definition in different situations. Specifically, when the system of insight has enough context to skip the manual review by the ticket office and potentially allows the process to be completely automated (that is, to become a straight-through process). Furthermore, by changing the manual start to an automated start, you now push work to the ticket office instead of relying on them to detect the need for reschedule and initiate the process themselves.

Figure 8-79 shows the modified process diagram.

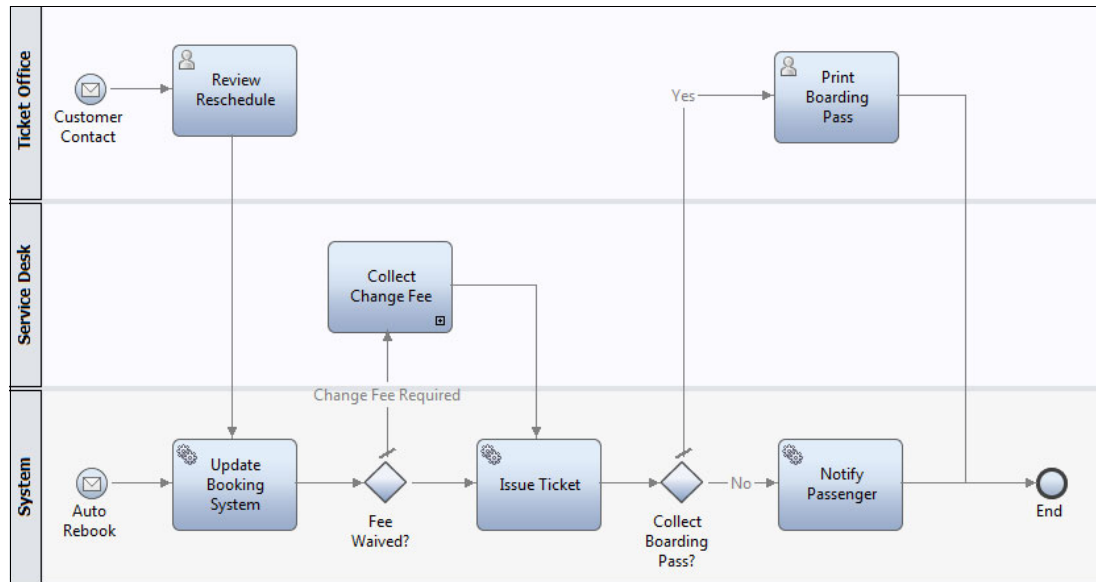


Figure 8-79 The message triggered ticket rebooking process

The following steps guide you through the process of creating and configuring a Message Start Event with a UCA to start a service:

1. To insert a Message Start Event, select the start item from the palette on the right side and place it on the canvas. Then, modify the start event type by using the drop-down in the Implementation tab in the Properties pane below the canvas (Figure 8-80).

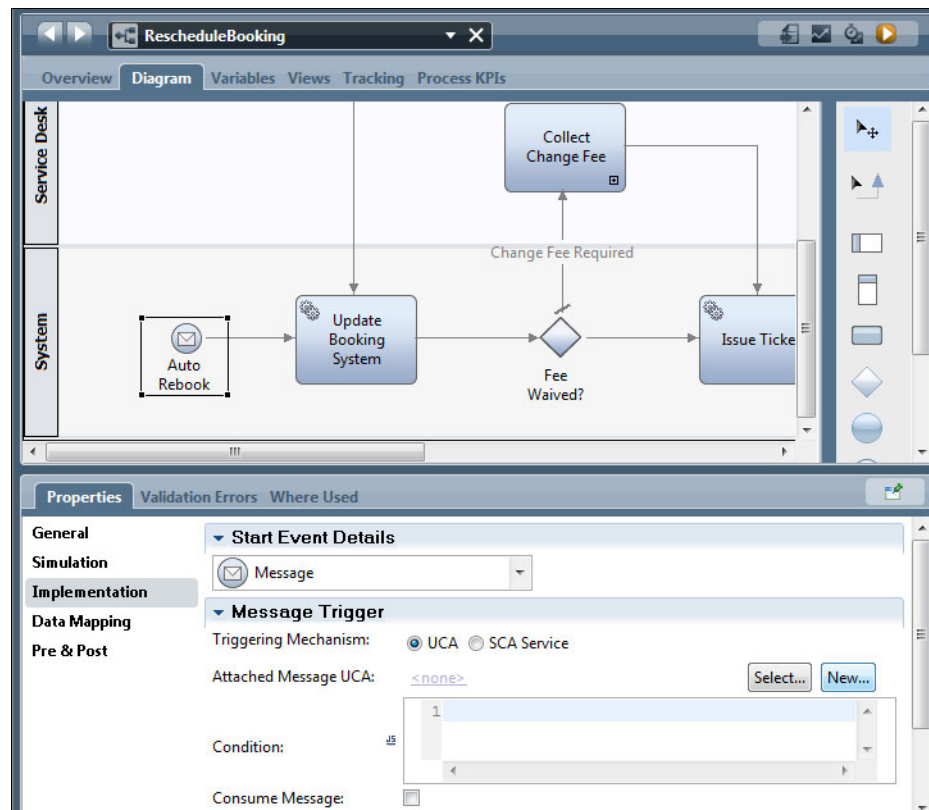


Figure 8-80 Creating a Message Start Event

2. To trigger this new Message Start Event, add a UCA. A UCA provides a triggering mechanism for the business-process diagram and is associated with a system-service that is used to handle message manipulation before data reaches the process.
3. Clicking **New** in the Start Event implementation tab opens the UCA window shown in Figure 8-81. UCAs can also be triggered based on timers, but in this scenario you want it to be triggered on receipt of an event.

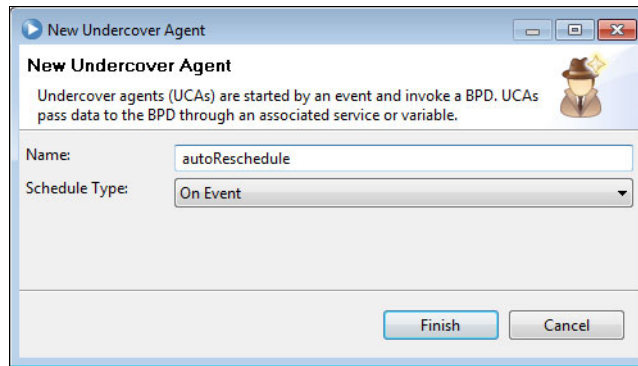


Figure 8-81 Creating an Undercover Agent

4. After creating the UCA, open it from the **Implementation** menu and then add a service definition that specifies the service parameters (Figure 8-82).

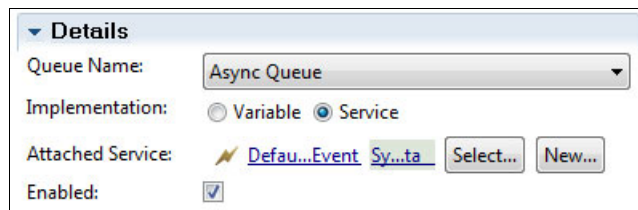


Figure 8-82 Attaching a service to a UCA.

5. In this simple scenario, the service is not required to do any processing and so you simply connect the start and end markers directly (Figure 8-83).

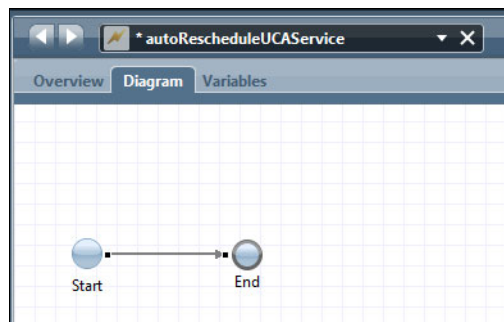


Figure 8-83 A simple service for use with a UCA

6. Next, define the data objects that are passed to the service. Here, input and output parameters match exactly as this service acts as a simple pass-through service (Figure 8-84).

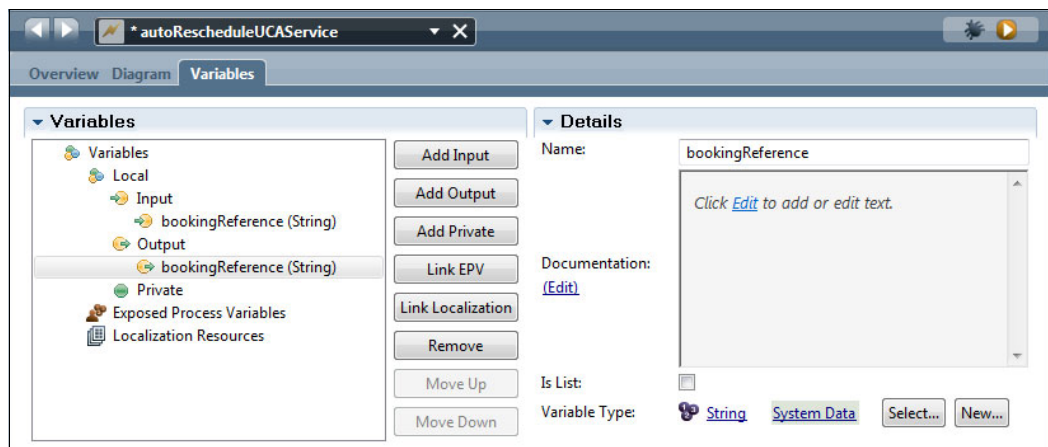


Figure 8-84 Defining the input and output interface for the UCA service.

The final important step in the UCA editor is to note the event message identifier shown at the bottom of Figure 8-85. This ID is passed as part of the source message so that the platform knows which event service to route the message to (see Example 8-36 on page 192).

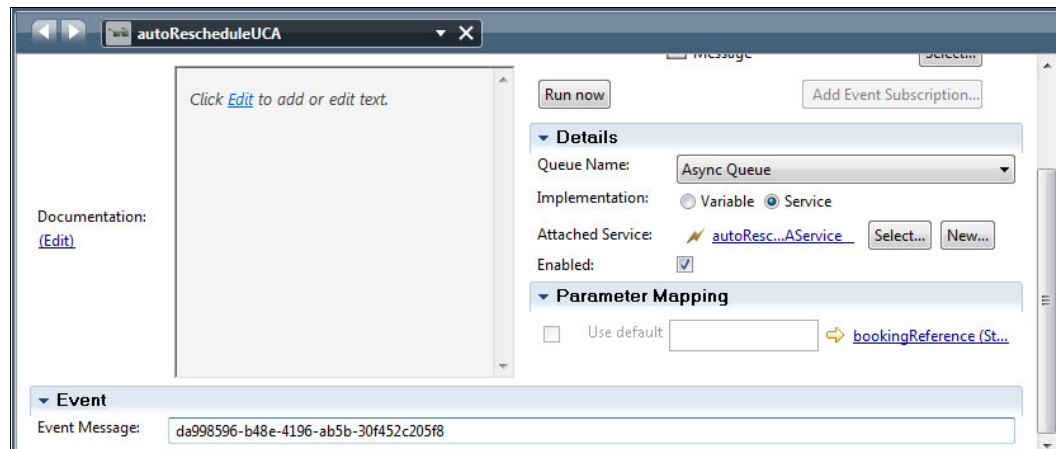


Figure 8-85 Looking up the Event Message Identifier

- Finally, you need to store the data that will be sent through the service to a local process variable. To accomplish this, return to your business-process diagram to map the output of your UCA service to a variable as shown in Figure 8-86.

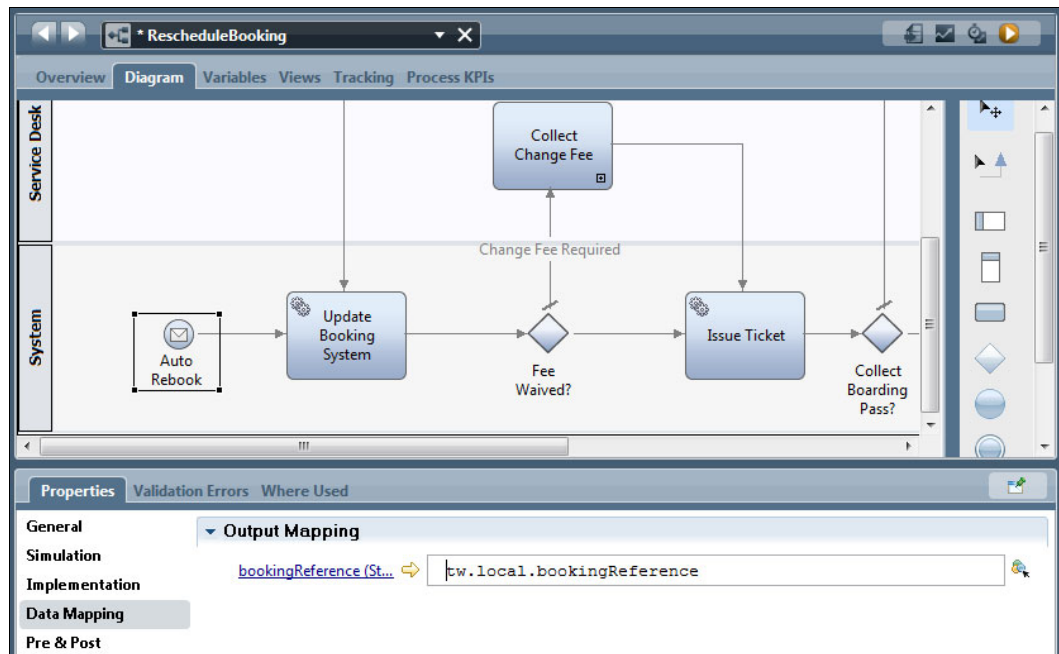


Figure 8-86 Mapping the output of the UCA service.

Configure Decision Server Insights

To connect DSI to IBM BPM, you need several attributes about the event configuration from the IBM BPM internal integration bus. This bus is provided by the IBM BPM WebSphere Application Server and is typically referred to as the WebSphere Application Server Service Integration Bus (SIBus). Complete these high-level steps:

- Extend the connectivity definition of the DSI solution.
- Add the JMS connectivity features to the DSI server configuration.
- Create a deployment configuration that includes the correct connectivity parameters for the IBM BPM server.

Extend the connectivity definition

The following steps define the tasks that are required to extend the connectivity definition:

- The existing connectivity definitions file (connectivity.cdef in the Connectivity Definitions folder of the DSI solution) is extended to include the definitions shown in Example 8-35. For the HTTP connectivity definitions, we add a binding and an endpoint.

This example has a single endpoint that provides a connection factory and a destination name that are defined in the next section.

Example 8-35 Connectivity Definitions for Integration with IBM BPM

```
define outbound binding 'processBinding'
using
  message format application/xml,
  protocol JMS ,
  delivering events :
    - customer contact invocation, transformed using "bpmManual.xml"
    - reschedule process invocation, transformed using "bpmAuto.xml".
```

```
define outbound JMS endpoint 'bpmEndpoint'
  using binding 'processBinding' ,
  connection factory "jms/bpmConnectionFactory",
  destination "bpmEventQueue".
```

2. The binding specifies the JMS protocol and lists a number of different events from which you want to trigger an action in IBM BPM. Each event type is annotated with a transformation that gets applied before the event reaches IBM BPM. This transformation changes the format of the XML message from the DSI schema to a structure that is recognized by IBM BPM.

Example 8-36 shows a sample XSL transformation for events sent to IBM BPM from DSI. This simple scenario only requires you to send a booking reference on to IBM BPM because it is assumed that IBM BPM will look up any further information directly from the system of record.

Example 8-36 Outbound transformation for use with IBM BPM, bpmAuto.xsl

```
<?xml version="1.0"?>
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
  xmlns:event="http://www.ibm.com/ia/xmlns/default/rdbook_airline_bom/model"
  exclude-result-prefixes="event" version="1.0">
  <xsl:template match="event:RescheduleProcessInvocation">
    <eventmsg>
      <event processApp="S0I">da998596-b48e-4196-ab5b-30f452c205f8</event>
      <parameters>
        <parameter>
          <key>bookingReference</key>
          <value>
            <xsl:value-of select="event:booking" />
          </value>
        </parameter>
      </parameters>
    </eventmsg>
  </xsl:template>
</xsl:stylesheet>
```

Table 8-2 highlights the key attributes of the transformation in Example 8-36.

Table 8-2 Transformation attributes for outbound connectivity with IBM BPM

Attribute	Description
BOM namespace	Found under the targetNamespace attribute in your exported event types schema.
	For example: http://www.ibm.com/ia/xmlns/default/rdbook_airline_bom/model
Event name	The XML name of the event that this transformation applies to.
	For example, RescheduleProcessInvocation
Process application	The acronym for your IBM BPM Process Application. Found in the Process Center repository (Figure 8-87 on page 193)
	For example, S0I

Attribute	Description
Event message ID	The identifier of your target UCA (Figure 8-85 on page 190). In this scenario, this is the only value that differs between bpmAuto.xml and bpmManual.xml.
	For example, da998596-b48e-4196-ab5b-30f452c205f8
Parameter key	The parameter name from the definition of your UCA service (Figure 8-84 on page 190). You will need multiple parameter elements in your schema if you need to map multiple values.
	For example, bookingReference
Parameter value	The element from your DSI event XML that contains the value that you want to map to the input parameter key.
	For example, booking.

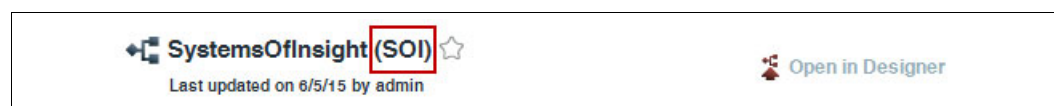


Figure 8-87 The Process Application acronym shown in the Process Center in IBM BPM

Add JMS connectivity features

As in “Adding connectivity features to your DSI server configuration” on page 182, extend your server.xml file to add extra features that enable the use of JMS connectivity by DSI. Again, provide both inbound and outbound features for completeness (Example 8-37).

Example 8-37 Extra WebSphere Liberty Profile features required for JMS connectivity in DSI

```
<feature>ia:iaConnectivityInboundJMS-8.7.1</feature>
<feature>ia:iaConnectivityOutboundJMS-8.7.1</feature>
<feature>wasJmsClient-1.1</feature>
```

Create a connectivity configuration

Follow the next steps to create a connectivity configuration in DSI, which will allow output events to be placed in the IBM BPM JMS queue.

1. In addition to creating the connectivity configuration (Example 8-35 on page 191), you also need to create a connectivity configuration that tells DSI how to connect to your JMS destination.
2. Right-click the solution project in Insight Designer and select **Deploy** → **Configure & Deploy**. A wizard opens that guides you through most of the required configuration steps. The wizard does not include a step to automatically include authentication information, so this step is added manually step at the end.

Table 8-3 and Table 8-4 list the values that are needed to configure the outbound JMS connection to the IBM BPM application server with a basic unsecured connection. Refer to the product documentation for WebSphere Application Server for further details about JMS messaging:

https://www.ibm.com/support/knowledgecenter/SSEQTP_8.5.5/com.ibm.websphere.wlp.doc/ae/cwlp_messaging.html

Table 8-3 Outbound JMS Connection Factory

Attribute	Description
Provider type	WebSphere Application Server
Bus name	The name of the SIBus in WebSphere Application Server For example, BPM.ProcessCenter.Bus or BPM.ProcessServer.Bus
Remote server address	An address string that includes the host name and SIB port number. The port number can be found in the WebSphere administrative console under Servers → Server Types → WebSphere application servers → Communications → Ports → SIB_ENDPOINT_ADDRESS . For example, localhost:7276:BootstrapBasicMessaging
Target transport chain	InboundBasicMessaging

Table 8-4 lists the values need to configure the outbound JAM destination.

Table 8-4 Outbound JMS destination

Attribute	Description
Provider type	WebSphere Application Server
Destination type	javax.jms.Queue
Destination name	This is found in the WebSphere administrative console under Resources → JMS → Queues → eventqueue → Connection → Queue name . (Figure 8-88). Note that this is not the JNDI name of the queue. For example, eventqueueDestination.Node1.server1

The screenshot shows the 'Connection' configuration page in the WebSphere administrative console. It contains three dropdown menus: 'Bus name' with the value 'BPM.ProcessCenter.Bus', 'Queue name' with the value 'eventqueueDestination.Node1.server1', and 'Delivery mode' with the value 'Application'. The 'Queue name' dropdown is preceded by an asterisk, indicating it is a required field.

Figure 8-88 Finding the Bus name and Queue name

3. After completing the connectivity configuration wizard, your configuration file contains entries similar those shown in Example 8-38. The final step is to add the `userName` and `password` attributes to the `properties.wasJms` element. Example 8-38 shows the password in plaintext, but you should hide the password and use an authentication alias:

http://www.ibm.com/support/knowledgecenter/SSEQTP_8.5.5/com.ibm.websphere.wlp.doc/ae/twlp_sec_basic_registry.html

To authenticate with the bus, the user must be in the bus connector role with the appropriate permissions. See *Securing service integration in the IBM WebSphere Application Server Knowledge Center* at:

http://www.ibm.com/support/knowledgecenter/SSEQTP_8.5.5/com.ibm.websphere.base.doc/ae/tjr9999_.html

Example 8-38 Outbound Connectivity configuration for IBM BPM JMS event queue

```
<jmsConnectionFactory jndiName="jms/bpmConnectionFactory">
  <properties.wasJms busName="BPM.ProcessCenter.Bus"
    remoteServerAddress="localhost:7276:BootstrapBasicMessaging"
    targetTransportChain="InboundBasicMessaging"
    userName="admin" password="passw0rd"/>
</jmsConnectionFactory>

<jmsQueue id="bpmEventQueue" jndiName="bpmEventQueue">
  <properties.wasJms queueName="eventqueueDestination.Node1.server1"/>
</jmsQueue>

<ia_outboundJmsEndpoint endpoint="rdbook_airline/bpmEndpoint"
  jms.persistent.delivery="true"/>
```



Building systems of insight with ODM Advanced

This chapter provides supporting information for organizations that choose to build a system of insight with IBM Operational Decision Manager Advanced (ODM Advanced). It provides guidance for solution architects, project managers, and developers on the following topics:

- ▶ The process of adopting the systems of insight paradigm in your business, and how to derive business value at each stage of this process
- ▶ Development cycles for IBM Operational Decision Manager Decision Server Insights (DSI) and ODM Decision Server Rules (DSR)
- ▶ Considerations when planning your DSI solution, and tips about how to design your solution for performance
- ▶ Reference topologies for DSI deployments
- ▶ How to maintain transactionality of your DSI-centric system of insight

This chapter covers the following topics:

- ▶ Adoption
- ▶ Request-driven decisions development cycle
- ▶ Situation-driven decisions development cycle
- ▶ Considerations for planning and designing DSI deployments
- ▶ ODM Decision Server Insights topologies
- ▶ Transactionality and rollback

9.1 Adoption

This section describes the adoption process for IBM Operational Decision Manager Decision Server Insights (ODM DSI) specifically, but some of the general points can apply to other systems of insight applications.

9.1.1 Identify use case

If you are at the stage where you have no system of insight, then the first task is to identify where a system of insight can provide business value.

You can use these two methods to identify a use case for a system of insight:

► *Action driven*

This approach involves potentially automating existing actions that your business takes, or adding value for your customers by identifying new actions that can be taken. Therefore, thinking about what these actions are or could be determines what a system of insight can do for your business.

Ask yourself:

- What situations do I want to detect?
- What customer “magic moments” do I want to capture? (for example, airline customers being rebooked and notified before they even realize that they have missed their flight)
- How can I differentiate my response to events from those of my competitors?

► *Data driven*

This approach involves looking at what information your business has access to that it is not currently getting full value from. An example of this would be some data that you collect in real time, but do not process in real time. The data loses value with time, so by moving to a real-time system of insight, you can maximize the value of your data and potentially discover things that you were not aware of.

Ask yourself:

- What data source does my business have that could be used in different or time-sensitive ways?
- Are there known data patterns that I could respond to if I identified them in time?

9.1.2 Maturity model

Figure 9-1 presents the maturity of a system of insight.

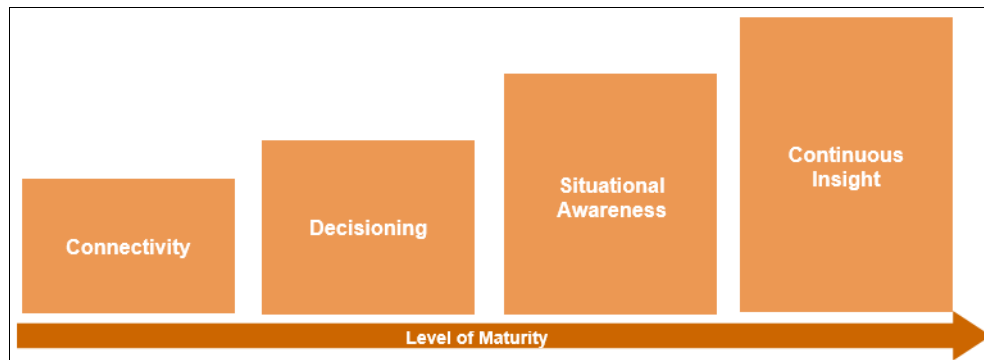


Figure 9-1 Maturity model for a system of insight

- **Connectivity**

At this stage, you have some coordination and collocation of data, but without any reasoning. This process can include bringing together of data from existing systems. Some examples of potential business value at this stage might include dashboarding, monitoring, and data enrichment. Existing manual decision-making processes can be made better informed, and therefore more reliable.

- **Decisioning**

This step introduces the concept of automated decision making, perhaps through some business rules processing, with these rules linked back to and enforced by business policy. The value offered here is in automating and centralizing your business logic to institutionalize best practice in every customer interaction. This is the stage corresponding to the value added by ODM DSR.

- **Situational awareness**

This stage is where the situational detection capabilities of DSI are seen. Analytics and complex event filtering combine to offer real-time situation detection, which is combined with the decision-making capabilities already in place. At this stage, you can start to make decisions in real time when situations that require business logic are detected. The business value at this stage is in being able to react to these situations in a consistent and informed manner at the time they happen, rather than, for example, waiting for an overnight batch process to complete.

- **Continuous insight**

This is the stage where stream-based analytics and temporal event history correlation are brought into the picture. Predictive analytics are integrated with the rule processing. DSI offers a lot of power at this level. Here you can predict and identify risks before they affect you business, improve customer relationships, and help to identify new markets for new or existing products and services. Proactive decision making helps to avoid negative situations and encourage positive ones to occur. This is where the customer “magic moments”, introduced in 9.1.1, “Identify use case” on page 198, can be identified and cultivated regularly.

Example solution roadmap

After you identified which stage of the maturity process that you are at, the question becomes one of how to move your solution forward to higher levels of maturity to gain more value from your system of insight.

Figure 9-2 includes an example solution roadmap for an event-driven system of insight.

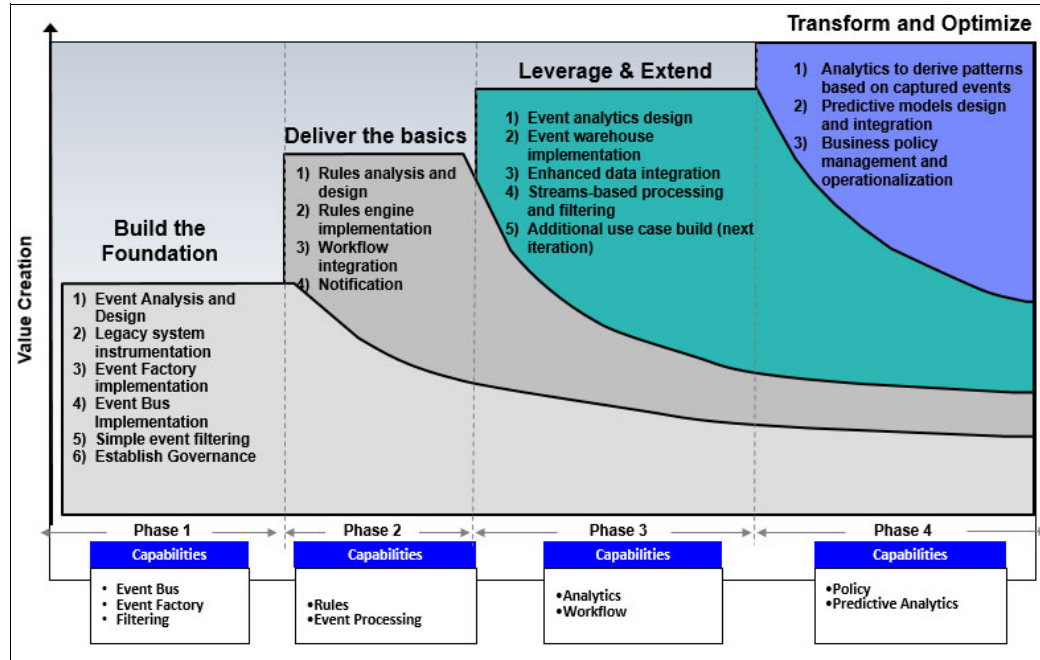


Figure 9-2 Example solution roadmap

The roadmap emphasizes the approach of adding real business value with every step up in capability, which was introduced in 9.1.2, “Maturity model” on page 199.

9.2 Request-driven decisions development cycle

This section focuses on the rule application development cycle as shown in Figure 9-3, with DSR as the example platform.

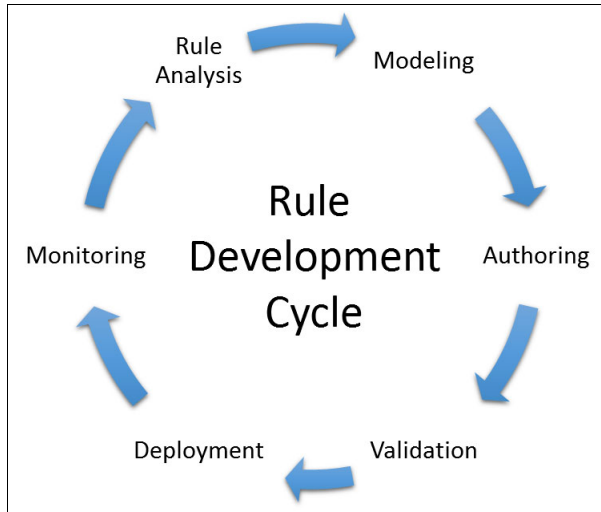


Figure 9-3 Rule development lifecycle

9.2.1 Rule analysis

It is important to understand that analysis and design are the starting point for every rule application implementation. Having the analysis and design first provides a clear understanding about what needs to be implemented. This remark is important because, in general, the ODM rules implementation approach is bottom up, but this does not imply a lack of prior analysis.

Figure 9-4 presents a simple example of the top down process of analysis and design phase in preparation for the bottom up implementation phase. For example, the business object model and associated vocabulary relevant to the rules must be defined before the technical model can be implemented. This means that business rules are identified, with their corresponding business terms and phrases, before these rules can be created.

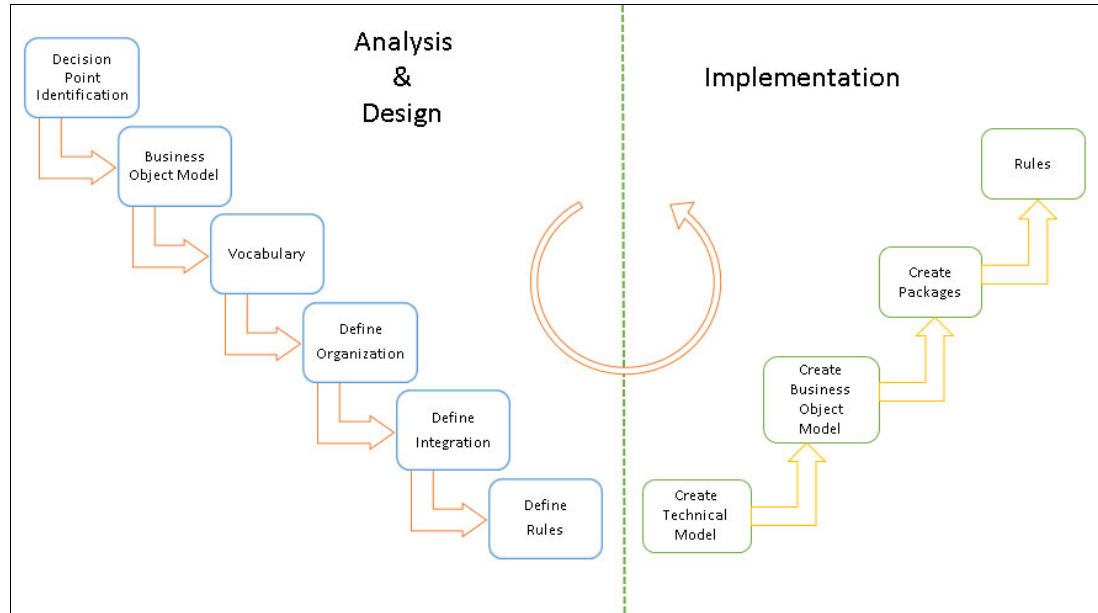


Figure 9-4 Top-down approach example for rule analysis

The DSR solution implementation approach followed by IBM Software Services is based on a proprietary methodology that uses the Agile Business Rules Development (ABRD) approach as a base. For more information about ABRD and documentation templates to use during a rule project implementation, see the following web address:

http://epf.eclipse.org/wikis/epfpractices/practice.tech.abrd.base/guidances/practices/abrd_697B2958.html

9.2.2 Modeling

Before authoring any rule artifact, you must define and build the execution layer of the business rules application. This step is the modeling phase of development. Rule Designer is the tool provided by Decision Server Rules where you set up the framework for the rule authoring. Rule Designer is considered the main tool that is used by Rule Developers to perform the authoring, testing, and management of rule assets.

9.2.3 Authoring

This phase can be considered the core for a business rule solution. It contains all the logic that drives decisions based on input data. For more information about the available options for rule authoring, see 6.2, “Request-driven decisions in ODM Decision Server Rules” on page 73.

9.2.4 Validation

Rule validation is a key task in the development of rule applications. Rule authors need to know whether rules are written correctly, and make sure that the rules will be triggered correctly when appropriate data is passed to the system.

ODM provides validation capabilities for rule developers using Rule Designer, and in Decision Center Enterprise Console and Business Console for business users.

This rule validation capability is also integrated with Microsoft Office where validation scenarios are created using Microsoft Excel files. Figure 9-5 shows the content of these files where a list of scenarios is created and data required for the decision is provided.

		the borrower					the loan	
Scenario ID	description	first name	last name	b	S	city	start date	nu amount
Very low risk	Very low risk loan accepted	Sam	Adams	41	98		8/1/2009	# 100000
Low risk	Low risk loan accepted	Bob	Schwartz	41	94		8/7/2010	# 500000
Average risk	Average risk loan accepted	John	Johnson	41	88		9/1/2010	# 900000
Amount too high	Loan rejected when amount too high	Mary	Doe	41	32		8/15/2010	# 1100000

Scenarios Expected Results HELP

Figure 9-5 Test suite file in Excel

ODM generates an empty Excel file template with all BOM elements needed as input and output for the specified rule project. In Figure 9-5, columns with yellow colored header (the borrower column and the loan column) include the verbalized representation of the required attribute term and phrase. In this case, these are the borrower and the loan.

Users are responsible for populating the Scenarios tab in the Excel file, and also the Expected Results tab, which includes the expected values of the response.

Figure 9-6 shows the execution results after a test suite was run in Business Console. This summary report shows a consolidated number of successful and failed scenarios. The report also provides a detailed summary of every scenario, showing whether the expected result defined in the Excel file was returned or not. The expected result for Scenario 2 in Figure 9-6 on page 204 was the loan report is approved equals... true, but the returned value was false.

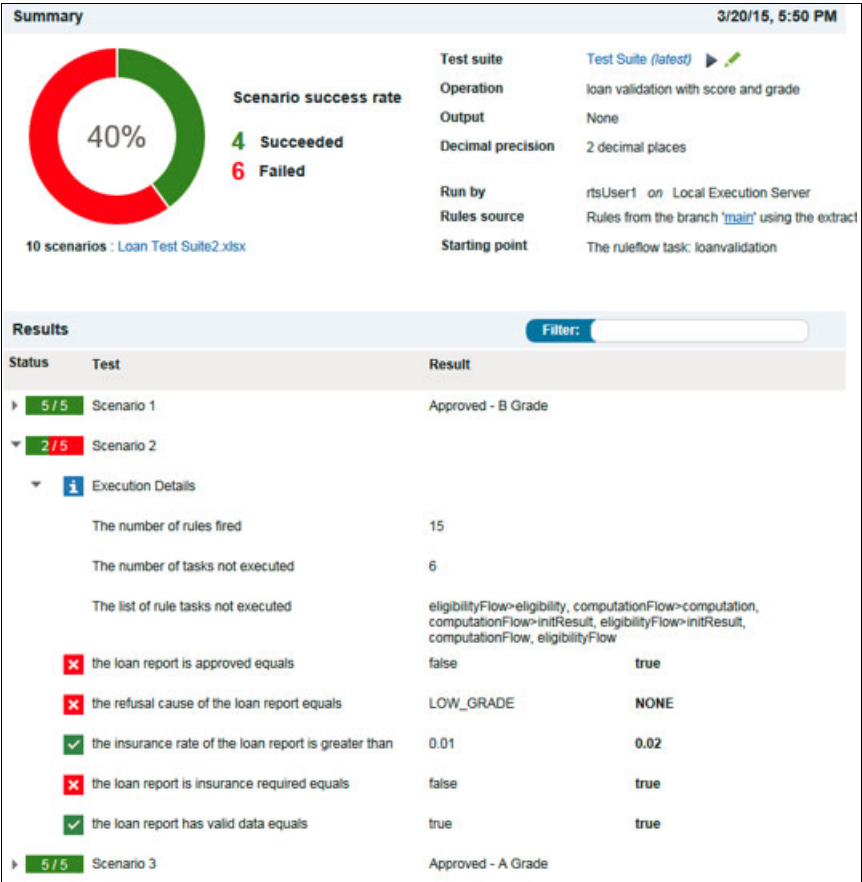


Figure 9-6 Test suite report from Business Console

For more information about Excel scenario files, see the IBM Knowledge Center at:
http://www.ibm.com/support/knowledgecenter/SSQP76_8.7.1/com.ibm.odm.dcenter.bu.bconsole/shared_testsim_topics/con_di_excel.html?lang=en

In addition to validating rules for one scenario, ODM can validate multiple scenarios as a unit against predefined key performance indicators (KPI) and metrics. This task is called simulation, and requires multiple steps to be defined to generate an outcome.

Figure 9-7 shows a graphical representation of the steps that are required to create a simulation in ODM.

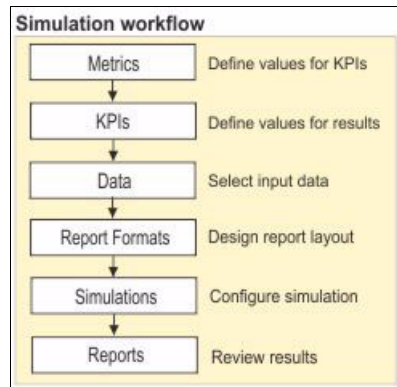


Figure 9-7 Steps to create a simulation in Business Console

- ▶ **Metrics:** Define the values that will be used in KPIs.
- ▶ **KPIs:** Shows the results of defined metrics. It can be a scalar value or a grouped set of values.
- ▶ **Data:** Corresponds to input scenarios used during the simulation. ODM has existing functionality for Microsoft Excel integration that can be customized to use a different data provider.
- ▶ **Report Format:** Defines the output format for a report, such as pie chart, line chart, or bar chart.
- ▶ **Simulations:** Defines the list of simulations configured with required KPIs, Data, and Report Formats.
- ▶ **Reports:** Provides historical access to all reports that you have run.

Example 9-1 shows the definition of metrics and KPIs needed to create a report.

Example 9-1 Simulation metrics and KPIs examples

```

Metric: Granted loan overall
    'the loan report' is approved
KPI: ratio of granted loans overall
    ratio of 'Granted loan - overall'

Metric: granted loan with scoring
    'the loan report' is approved when 'the grade' is not null
KPI: ratio of granted loans with scoring
    ratio of granted loans with scoring

Metric: Scored loan
    'the grade' is not null
KPI: ratio of scored loans
    ratio of 'Scored loan'

Metric 1: Scored loan
    'the grade' is not null
Metric 2: Loan grad
    'the grade'

```

KPI: Scored loans by grade

number of 'Scored loan' grouped by 'Loan grade'

Metric 1: Granted loan with scoring

'the loan report' is approved when 'the grade' is not null

Metric 2: Loan amount

the amount of 'the loan'

KPI:ratio of granted loans by amount

ratio of 'Granted loan - with scoring' grouped by 'Loan amount' with a 100000 interval

Figure 9-8 shows the report outcome of a simulation using the metrics and KPIs defined in Example 9-1. The first three KPIs refer to a single scalar value, whereas the last two KPIs refer to a grouped set of values.

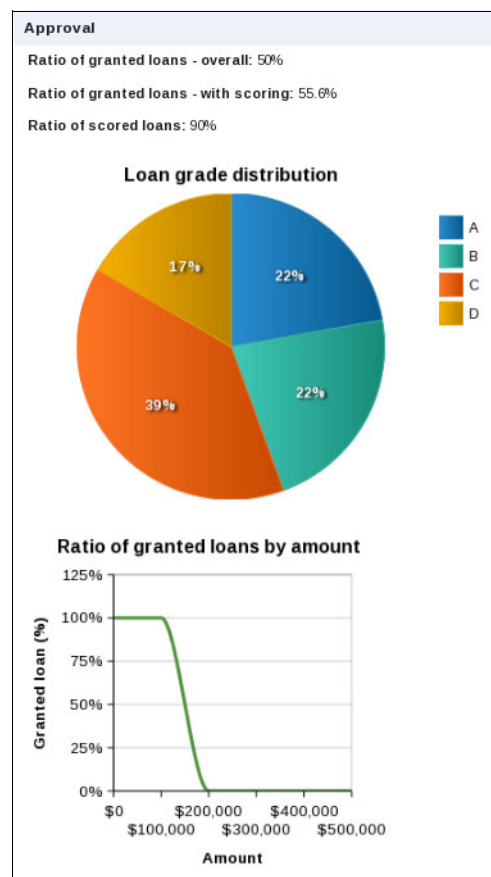


Figure 9-8 Simulation report output from Business Console

For more information about how to build a simulation, see the IBM Knowledge Center at:

http://www.ibm.com/support/knowledgecenter/SSQP76_8.7.1/com.ibm.odm.dcenter.bu.bconsole/simulation/con_bc_sim_interface.html?lang=en

9.2.5 Deployment

In the deployment phase, you make the authored artifacts available for execution. In preparation for deployment, a Decision Service rule project requires two components: Decision operations and deployment configurations.

The final output from the deployment phase is a RuleApp that contains a set of one or more rulesets.

9.2.6 Monitoring

Rule execution monitoring is available through different components. The Rule Execution Server (RES) console provides a view with the following ruleset statistics:

- ▶ Count: Total number of times a ruleset was run during the session.
- ▶ Total time: Total duration of ruleset execution in milliseconds.
- ▶ Average time: Average ruleset execution time based on Total time and Count.
- ▶ Max / Min: Longest and shortest ruleset execution time.
- ▶ First / Last Execution Dates.
- ▶ Last Execution duration.

Another useful monitoring component is Decision Warehouse, which is used for auditing decisions. Decision Warehouse is accessible through the RES console and provides tracing information that includes:

- ▶ Summary of execution details, such as decision ID, date, executed ruleset path, processing time, number of executed rules, and number of executed tasks.
- ▶ Decision trace with a tree of ruleflow tasks.
- ▶ Input parameters.
- ▶ Output parameters.
- ▶ Execution output.

For more information about Decision Warehouse, see the IBM Knowledge Center at:

http://www.ibm.com/support/knowledgecenter/SSQP76_8.7.1/com.ibm.odm.dserver.rules.res.console/topics/tpc_rescons_dw_intro.html?lang=en

Rule execution monitoring outside the Rule Execution Server console involves gathering information from logs, monitoring rule execution, and receiving notification of rule engine events.

Rule execution metrics are available through an MBean in the Java Management Extensions (JMX) infrastructure of the Java EE server. Monitoring engine events are also available through MBeans.

Additional information about monitoring RES events is available in the IBM Knowledge Center at:

http://www.ibm.com/support/knowledgecenter/SSQP76_8.7.1/com.ibm.odm.dserver.rules.res.managing/topics/tpc_res_monitor_intro.html?lang=en

9.2.7 Execution patterns

The modular architecture of the RES is flexible in meeting your needs for running rule applications. Java SE is applicable for stand-alone applications that requires rulesets to be close to the client. This approach can be referred as *embedded mode* or *embedded pattern*. When client applications are in the context of Java EE or web services, you need a more robust enterprise solution where multiple rulesets can be run at the same time, and there are continuous ruleset updates with multiple deployments. These requirements are managed by the RES architecture, and so it is referred as the *managed execution pattern*.

When organizations have an architecture based on services (a service-oriented architecture or SOA), the managed execution pattern becomes the recommended pattern.

When considering high throughput scenarios, Java SE can be useful for batch processing applications. Other batch processing scenarios can be implemented with a managed environment that includes Java Message Service (JMS) messaging and message-driven beans (MDBs).

Embedded execution in Java SE

Figure 9-9 depicts the Java SE execution components of RES. For this execution type, all subcomponents are located in the same Java virtual machine (JVM) as the client application. A Java SE rule session is used to run the ruleset. The client application is responsible for loading the ruleset before execution.

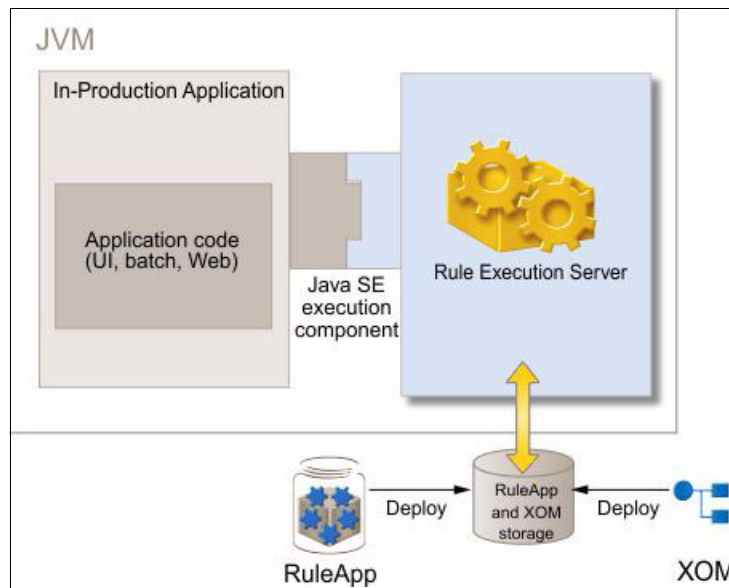


Figure 9-9 Java SE execution components

Managed execution in Java EE

Figure 9-10 depicts the Java EE execution components of a RES. This is considered a more flexible and robust pattern for organizations.

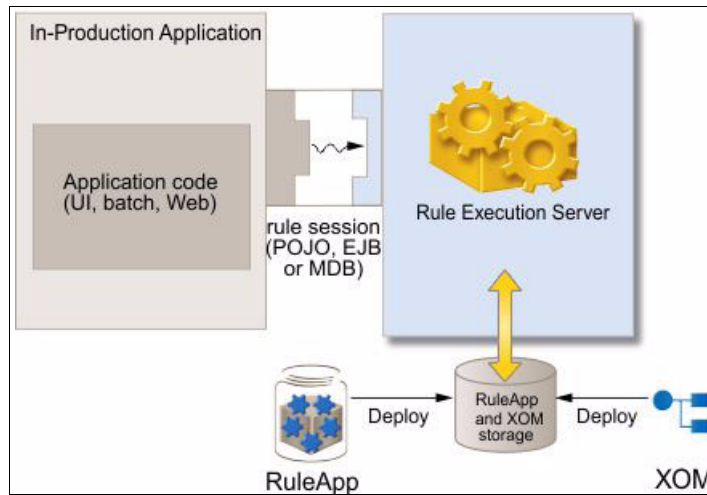


Figure 9-10 Managed execution in Java EE execution components

When using Java EE execution components, synchronous calls to a decision service from a remote client use stateful or stateless Enterprise JavaBeans (EJBs). Synchronous calls to a decision service from a local client use the local interfaces of stateful or stateless EJBs, or plain old Java objects (POJOs):

- ▶ EJBs are useful for remote client access capabilities, declarative transactions, and security descriptors.
- ▶ POJOs are useful for simpler packaging and deployment, and for use outside the EJB container.

Applications that require asynchronous calls to decision services can use the MDB, which provides a scalable means to call rulesets where high-latency or high peak-load is expected.

Managed execution with web service

Figure 9-11 shows the transparent decision services pattern that applications can use to access the rule engine. In this pattern, the client applications are usually remote. The client applications access the RuleApps through HTTP or HTTPs to the Rule Execution Server.

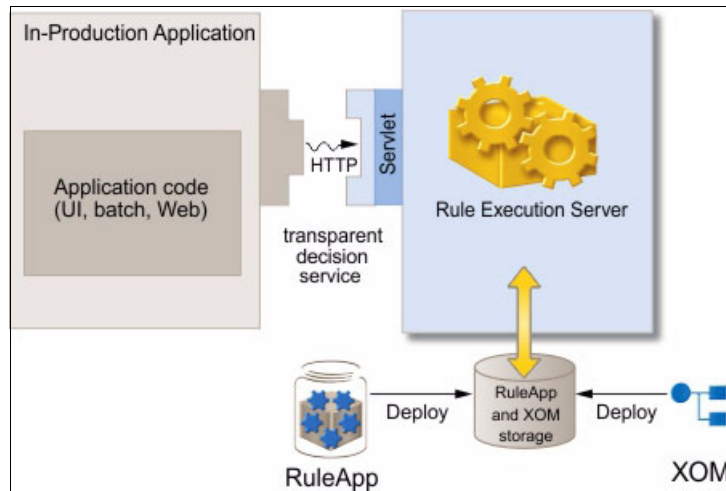


Figure 9-11 Managed execution with web services integration pattern execution components

The benefit of this pattern is that a web service is automatically available when a ruleset is deployed (including rulesets with Java XOMs). Rule Execution Server console also keeps access statistics for rulesets when the hosted transparent decision service is used.

9.2.8 Concept of operations

In the rule development process, the request-driven decision operations (operations) start with designing and creating RuleApps and end at the production server administration as illustrated in Figure 9-12.

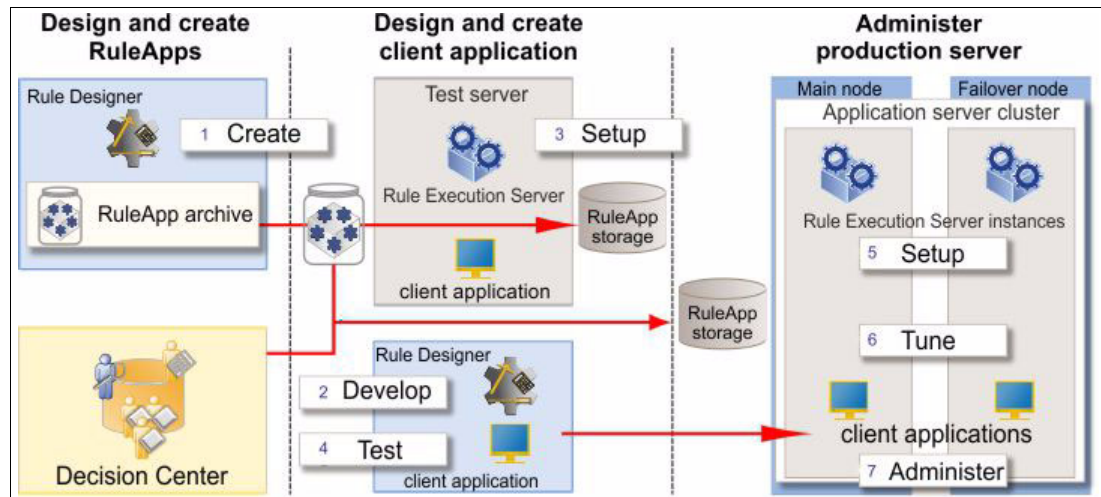


Figure 9-12 Request-driven decision operations

► Design and create RuleApps

In this phase, rule applications are created with Rule Designer and Decision Center.

Through Rule Designer, developers design, author, and test rule applications. Rule projects are synchronized between Rule Designer and Decision Center.

Through Decision Center, users author, manage, and validate rules.

As shown in Figure 9-12 on page 210, a RuleApp is generated and deployed in this phase for testing from Rule Designer or Decision Center. Whether deploying ruleapps will be allowed or not is defined as part of the Governance policy for each organization.

- Design and create client application

Depending on the type of application, in this phase, a client application can be built using Rule Designer to start rules deployed in a Rule Execution Server. The Rule Execution Server can be in a test or production environment. Figure 9-12 on page 210 shows develop, test, and setup activities under design and create client application to highlight the execution of testing rules deployed in a test environment.

- Administer production server

The production server administration includes setting up and tuning the infrastructure. Monitoring, auditing, and administering all rule applications deployed in the production environment are on-going activities.

9.2.9 Applicability in a service-oriented architecture (SOA)

An SOA can be described as a design pattern in which applications externalize components through web services or other forms. SOA is considered a loosely coupled architecture that allows the creation of collaborative applications.

This description clearly indicates that applications must externalize their capabilities as web services to be considered a SOA architecture. As described in 9.2.7, “Execution patterns” on page 208, DSR can externalize rule applications (RuleApps and rulesets) as web services automatically as part of the Rule Execution Service architecture. DSR can also be integrated with a Java EE architecture to support asynchronous requirements through MDBs. All these capabilities make it clear that Decision Server Rules is perfectly applicable to an SOA architecture.

Within the layers of the SOA reference architecture in Figure 9-13, decision services are part of two layers: The business process and services layers.

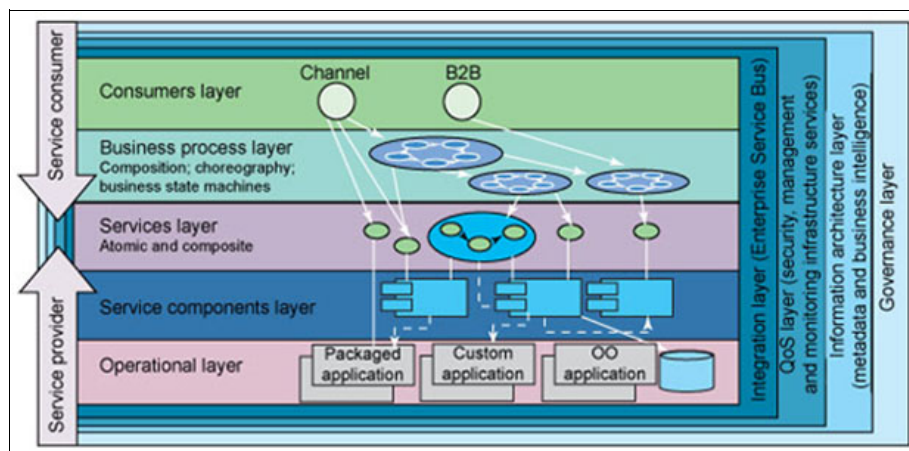


Figure 9-13 Layers of the SOA reference architecture¹

¹ developerWorks article: *Design an SOA solution using a reference architecture* at <http://www.ibm.com/developerworks/library/ar-archtemp/index.html>

From the middleware view of the SOA reference architecture in Figure 9-14, there is not an explicit notation of decision services in the diagram. However, you might need to have these decision services as a category. Decision services are now an important capability that can be used by all services described in the diagram. For example, process automation requires decision automation that makes sense to be externalized as a service, and not built into the logic of every process definition.

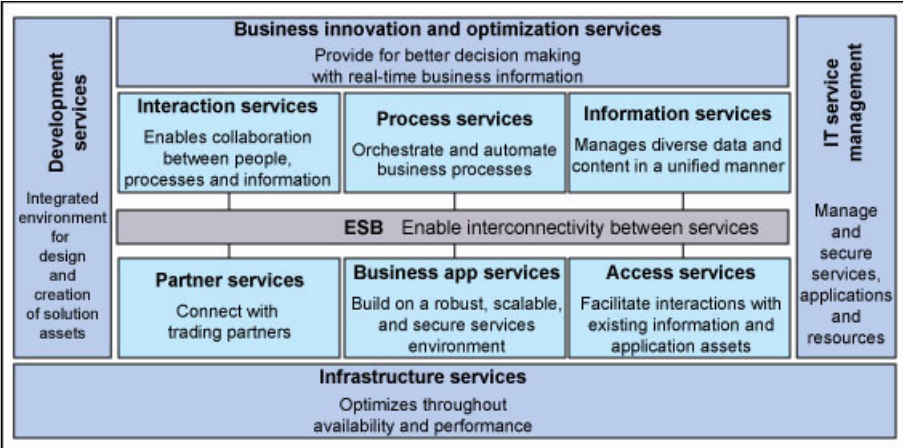


Figure 9-14 SOA reference architecture

In summary, DSR is not only applicable for an SOA architecture, but also plays an important role in this definition to complement other services.

9.3 Situation-driven decisions development cycle

This section describes the DSI solution development cycle that is depicted in Figure 9-15.

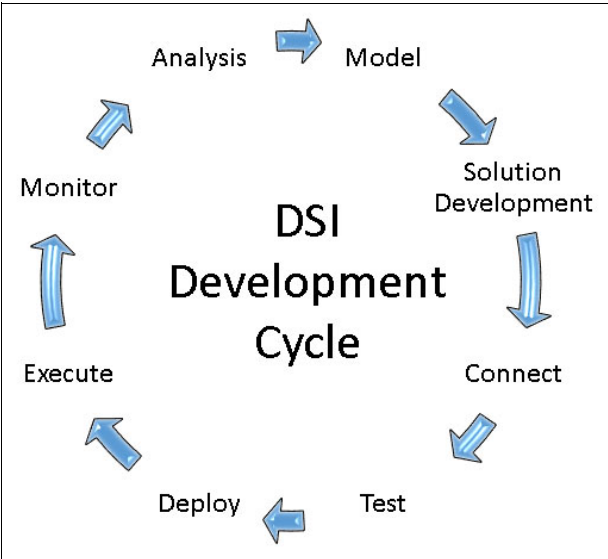


Figure 9-15 DSI development cycle

9.3.1 Analysis

The analysis phase for a DSI solution is similar to a business rules application. It follows the same top down approach for the analysis and then bottom up for the implementation (see 9.2.1, “Rule analysis” on page 201). In DSR, the focus is on the identification and implementation of decision points, whereas in DSI the focus is on the situation identification and the implementation needed to detect these situations.

Before starting the analysis of a situation-driven application, you must identify what situations are of interest. This involves understanding why a situation happens and how you want to process or react to this situation.

After situations are identified, analysis can take place to identify different actors involved in these situations. At some point, these actors might become entities in the system.

Another important task in the analysis phase is to identify the sources of information: Where the events come from, in what format, and how frequent they are. This information can be used to get an idea about the type of infrastructure required for the solution.

In summary, during analysis and design phase, consider the following concerns:

- ▶ Situation identification
- ▶ Entity identification
- ▶ Entity initialization
- ▶ Entity enrichment
- ▶ Event identification
- ▶ Event delivery
- ▶ Agent definition and justification
- ▶ Relationship between agents and entities
- ▶ Event processing and logic
- ▶ Geographic location data requirements
- ▶ Time requirements
- ▶ Event sources
- ▶ Type of connectivity with sources

For more information about design considerations, see the following webpages:

- ▶ IBM Knowledge Center at:

http://www.ibm.com/support/knowledgecenter/SSQP76_8.7.1/com.ibm.odm.itoa.develop/topics/odm_itoa_integrating.html

- ▶ The developerWorks article *Situation-driven Design with ODM Advanced - Decision Server Insights* at:

<https://developer.ibm.com/odm/2015/03/25/situation-driven-design-odm-decision-server-insights/>

9.3.2 Model

The modeling phase focuses on creating the business object model definition (BMD) file that includes entities and events definitions as described in 6.3.2, “Business model definition” on page 92. The business model definition file also contains information about initialization and enrichment of entities when needed.

9.3.3 Solution development

After the business object model is defined, you can define the logic that will be associated with the event processing through agent artifacts. For more information about agents, see 6.3.3, “Agents” on page 94.

9.3.4 Connect

DSI provides two connectivity layers for sending and receiving events. One layer is used for inbound events and the other for outbound events. These capabilities are described in 6.3.8, “Integration and connectivity” on page 103.

9.3.5 Test

Solution testing requires multiple events to be sent to an Insight Server. Figure 9-16 shows two available options for solution testing:

- ▶ Test client project: A Java client application that is used to test events logic. More details are included in “Test client project”.
- ▶ Event sequence file: Test option for more business users to create testing scripts using a business language. More details are included in “Event sequence file” on page 215.

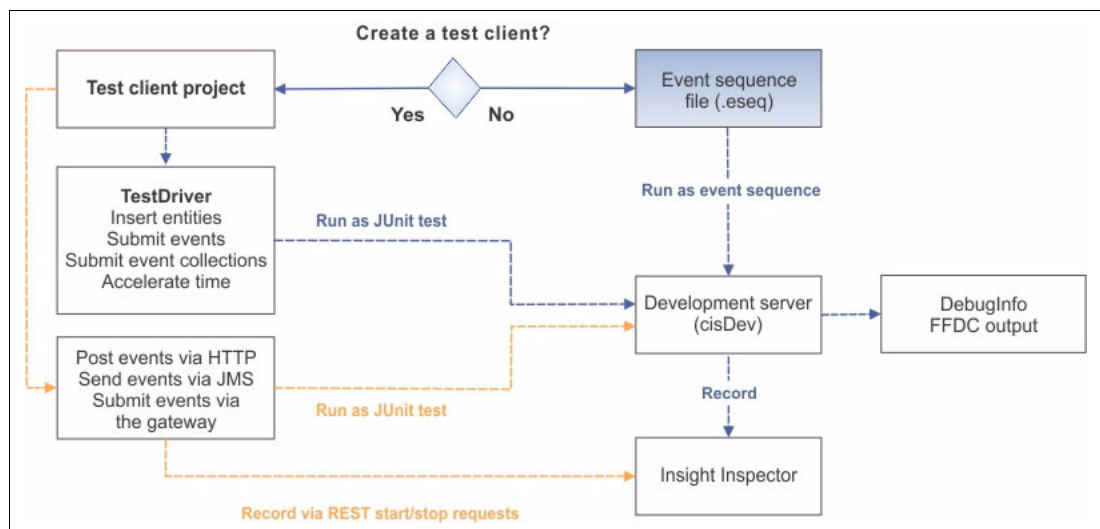


Figure 9-16 Solution testing options flow

Test client project

You can use the test client project in DSI to generate a Java class to emit events for a solution. The Java class just needs to be updated with the logic related to creating objects for the solution. As shown in Figure 9-16, when using the test client, all testing tasks must be coded. This includes entity creation, entity insertion in memory, and submission of events.

A step-by-step guide about creating a test client project is available in the IBM Knowledge Center at the following web address:

http://www.ibm.com/support/knowledgecenter/SSQP76_8.7.1/com.ibm.odm.dserver.rules.res.developing/topics/tsk_res_client_tds_proj.html?lang=en

Event sequence file

An event sequence file is a DSI artifact that is used to provide event details by using a similar business language as the one used for the business model definition.

Example 9-2 shows the syntax to create references to two entities: One for a customer and another one for a purchase.

Example 9-2 Creation of entities for an event sequence file

```
define 'the customer' as the customer identified by "John";

define 'purchase1' as a new purchase where
    the customer is 'the customer';
```

Example 9-3 shows the syntax to generate an event for a customer registration and a second event with a time stamp to start a sequence of a purchase five minutes after a customer registration. Note that this second event is not actually emitted five minutes later. The time stamp is adjusted instead.

Example 9-3 Creation of an event in an event sequence file

```
emit ( a new customer registration where
    the customer is 'the customer' ,
    the first name is "John" ,
    the last name is "Watson" ) ,
    time-stamped 3/25/2015 ;

emit purchase1 ,
    time-stamped 5 minutes later;
```

For more information about testing with event sequence files, see the IBM Knowledge Center at the following web address:

http://www.ibm.com/support/knowledgecenter/SSQP76_8.7.1/com.ibm.odm.itoa.test/topics/con_test_submiteseq.html?lang=en

In addition to the test client project and event sequence file, there are other ways to submit events to the grid to test solutions, such as:

- ▶ Use the gateway API to submit events
- ▶ Process events through HTTP (requires HTTP inbound connectivity)
- ▶ Process events through JMS (requires JMS inbound connectivity)

During testing, event sequences and entity information can be recorded and presented in a web format through Insight Inspector. Information about this component is available in “Insight Inspector” on page 90.

A complete section for testing is available in the IBM Knowledge Center at the following web address:

http://www.ibm.com/support/knowledgecenter/SSQP76_8.7.1/com.ibm.odm.itoa.test/topics/odm_itoa_test.html

9.3.6 Deploy

A DSI solution deployment is to make the solution artifacts, solution file, and connectivity files available for execution.

Deployment tasks are available through Insights Designer as shown in Figure 9-17.

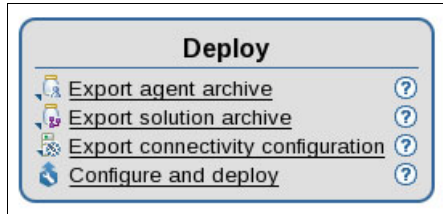


Figure 9-17 Deploy options from Insights Designer

You can export the solution and connectivity configuration files to manually deploy the application to the other environments. DSI provides two command-line scripts that allow this manual deployment process.

The configure and deploy wizard provides a simple step-by-step guide to select the server that the solution is to be deployed on and specify the deployment configuration name. You can also use the wizard to select the connectivity definitions that must be included as part of the deployment. The configure and deploy wizard is typically used during development rather than for production.

When deployment from Eclipse is not possible or desirable, deployment must be performed, by using the `solutionManager` script with the **deploy** command. This script provides multiple configuration options, including local or remote deployment.

More in-depth information about the `solutionManager` script options is available in the IBM Knowledge Center at the following web addresses:

- ▶ Deploying solution and agent archives:

http://www.ibm.com/support/knowledgecenter/SSQP76_8.7.1/com.ibm.odm.itoa.deploy/topics/tsk_deploy_using_scripts.html?lang=en

- ▶ **deploy** command:

http://www.ibm.com/support/knowledgecenter/SSQP76_8.7.1/com.ibm.odm.itoa.ref/topics/ref_itoa_sm_install.html?lang=en

Similar to the `solutionManager` script, for connectivity files, DSI provides a script called `connectivityManager` to deploy the connectivity solution by using the **deploy** command.

For more information about `connectivityManager` script, see the IBM Knowledge Center at the following web addresses:

- ▶ Deploying a solution connectivity application:

http://www.ibm.com/support/knowledgecenter/SSQP76_8.7.1/com.ibm.odm.itoa.deploy/topics/tsk_deploy_connect_config.html?lang=en

- ▶ **deploy** command:

http://www.ibm.com/support/knowledgecenter/SSQP76_8.7.1/com.ibm.odm.itoa.ref/topics/ref_itoa_cm_deploy.html?lang=en

9.3.7 Execute

When a solution is running, situations are identified and actions are triggered. Unlike Decision Server Rules, a DSI solution is not static when in execution. It continuously processes events and dispatches events to agents.

The execution phase in DSI comprises these steps:

- ▶ Processing of inbound events
- ▶ Creating, updating, or deleting entities
- ▶ Updating aggregates
- ▶ Evaluating and firing action rules
- ▶ Submission of events
- ▶ Processing of outbound events

9.3.8 Monitor

DSI provides monitoring capabilities through an MBean implementation to gather information about agents and connectivity:

- ▶ The AgentStatsMXBean API provides information about agents and events in the solution. For example, it can provide the amount of time to process all agent calls, the number of times events have been triggered, and the number of times agents have processed an event.
- ▶ The InboundEndpointMonitorMXBean API provides information about connection activity, such as the number of times a message is not processed due to a lack of available resources or the number of times an endpoint is called.
- ▶ SmartCloud Analytics provides capabilities to search for DSI event types and logs. It requires the SmartCloud Analytics console to have access to event types and logs through a web tool.

Information about monitoring solutions is available in the IBM Knowledge Center at the following web addresses:

- ▶ Interface AgentStatsMXBean:

http://www.ibm.com/support/knowledgecenter/SSQP76_8.7.1/com.ibm.odm.itoa.ref/html/api/html/com/ibm/ia/runtime/management/AgentStatsMXBean.html?lang=en

- ▶ Interface InboundEndpointMonitorMXBean:

http://www.ibm.com/support/knowledgecenter/SSQP76_8.7.1/com.ibm.odm.itoa.ref/html/api/html/com/ibm/ia/connectivity/management/InboundEndpointMonitorMXBean.html?lang=en

- ▶ Searching SmartCloud Analytics logs:

http://www.ibm.com/support/knowledgecenter/SSQP76_8.7.1/com.ibm.odm.itoa.admin/topics/tsk_search_2020.html?lang=en

9.3.9 Concept of operations

Figure 9-18 shows the operations flow for building a DSI solution. The process starts from Insight Designer at the upper left corner with a developer, an architect, and a business modeler working together on the analysis and design of the application.

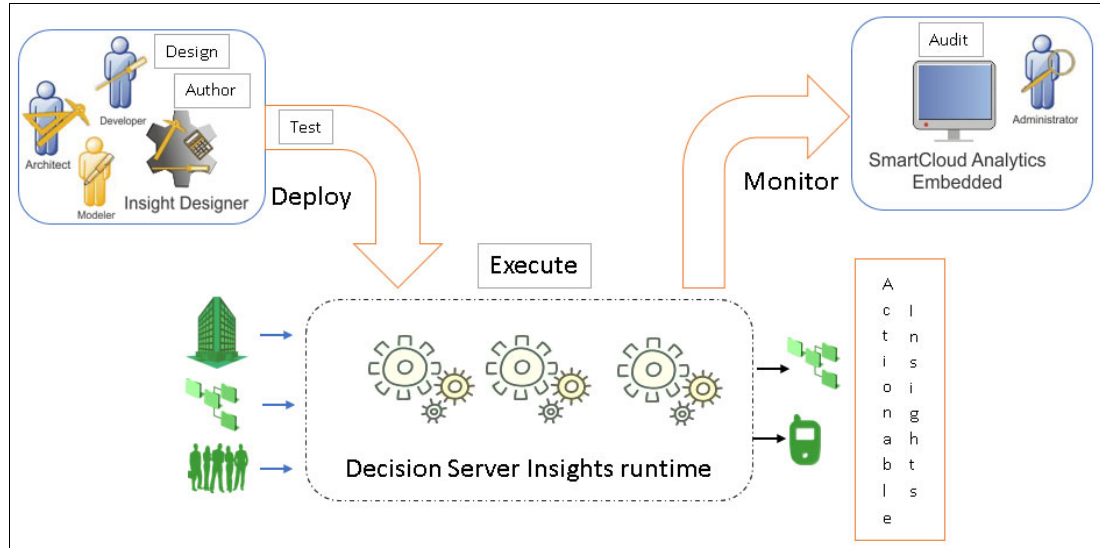


Figure 9-18 Decision Server Insights operations

A developer works on the authoring and testing of DSI solutions to make sure that the application works as specified by the requirements. The execution phase takes place with multiple event sources submitting events to the run time.

The DSI runtime server triggers required actions or events that are dispatched through the outbound connectivity layer to trigger the actionable insights on the receiver applications.

An administrator audits and monitors the system through SmartCloud analytics component or available MBean resources.

9.3.10 Applicability in an event-driven architecture

Event driven architecture (EDA) is an architecture pattern where changes in state of the system are represented and processed as events. As such, an EDA handles detection of, production of, consumption of, and reaction to events.

Dan Selman and Andy Ritchie propose a model for an EDA-based system of insight in their developerWorks article *Event Driven Architecture and Decision Management* at the following web address:

<https://developer.ibm.com/odm/docs/odm-capabilities/odm-advanced-decision-server-insights/event-driven-architecture-and-decision-management/>

Figure 9-19 shows a diagram of their model.

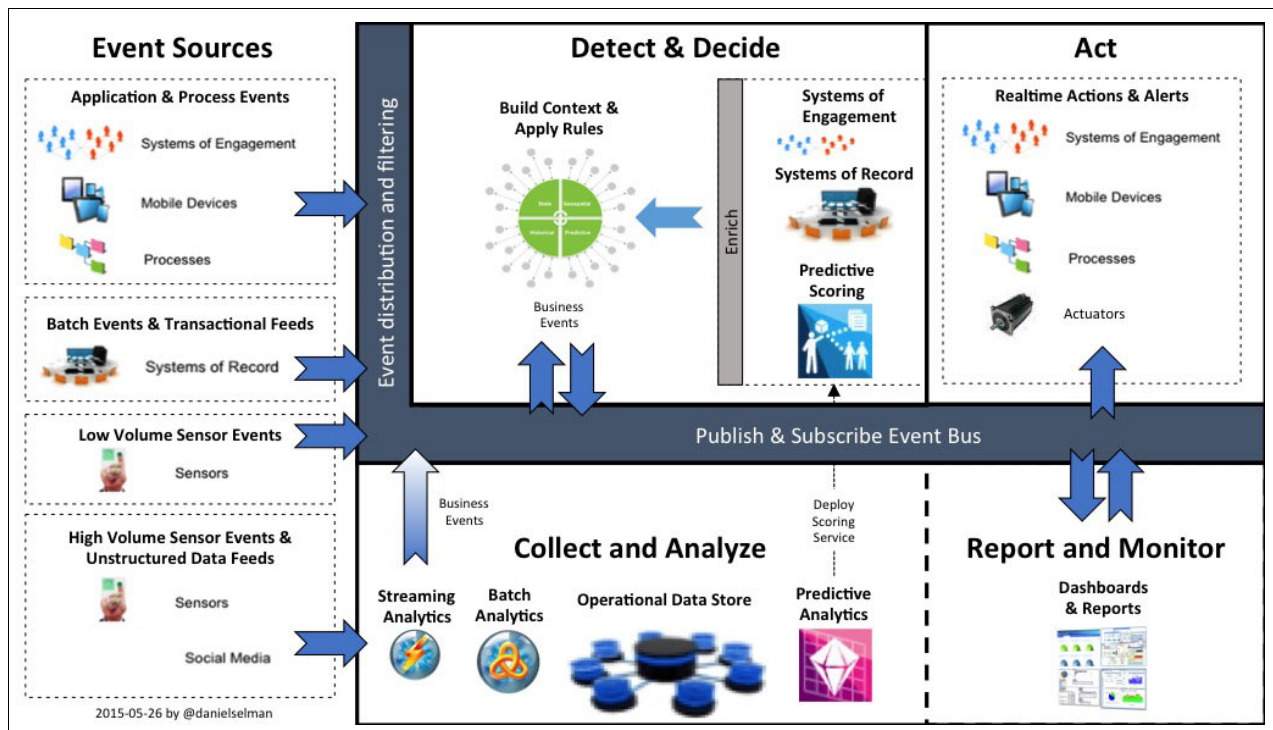


Figure 9-19 A model for a system of insight based on EDA

Dan Selman and Andy Ritchie consider six capabilities for an event driven architecture. The following list maps the systems of insight model to these capabilities:

- ▶ Event sourcing (*Consume*)
- ▶ Event distribution and filtering (*Consume*)
- ▶ Event collection and analysis (*Collect, Analyze*)
- ▶ Detection of situations and decision making (*Detect, Decide*)
- ▶ Taking action (*Action Execution*)
- ▶ Reporting and monitoring (*Analyze, Report*)

Note that the model in this book considers taking action (Act) as an interface whereas Dan Selman and Andy Ritchie consider taking action as part of the system

DSI fits into this EDA model well. DSI offers these capabilities:

- ▶ Event distribution and filtering (Consume)
- ▶ Detection of situations and decision making (Detect, Decide)
- ▶ Taking action (Act)

For more information about the systems of insights model, see Chapter 3.

9.4 Considerations for planning and designing DSI deployments

Decision Server Insights adheres to a compute-grid architecture (Figure 9-20) to provide high-performance and scalability. In this configuration, the computation of business logic is distributed across a number of nodes in a grid. By default, this grid also contains a local in-memory data store required to perform computations. As described in Chapter 6, this context contains the event history and entity state. Incoming events are routed to the correct place in the grid for processing according to the data distribution strategy and agent descriptors.

As is common in big data scenarios, DSI aims to minimize the movement of data to allow for maximum performance. In contrast to execution in most Decision Server Rules solutions, the processing is moved to where the data is, instead of moving the data to the logic.

This architecture highlights the need to carefully consider both the data storage requirements and computation requirements when designing such a solution. These requirements are tightly coupled with the business scenario that a solution is designed to address.

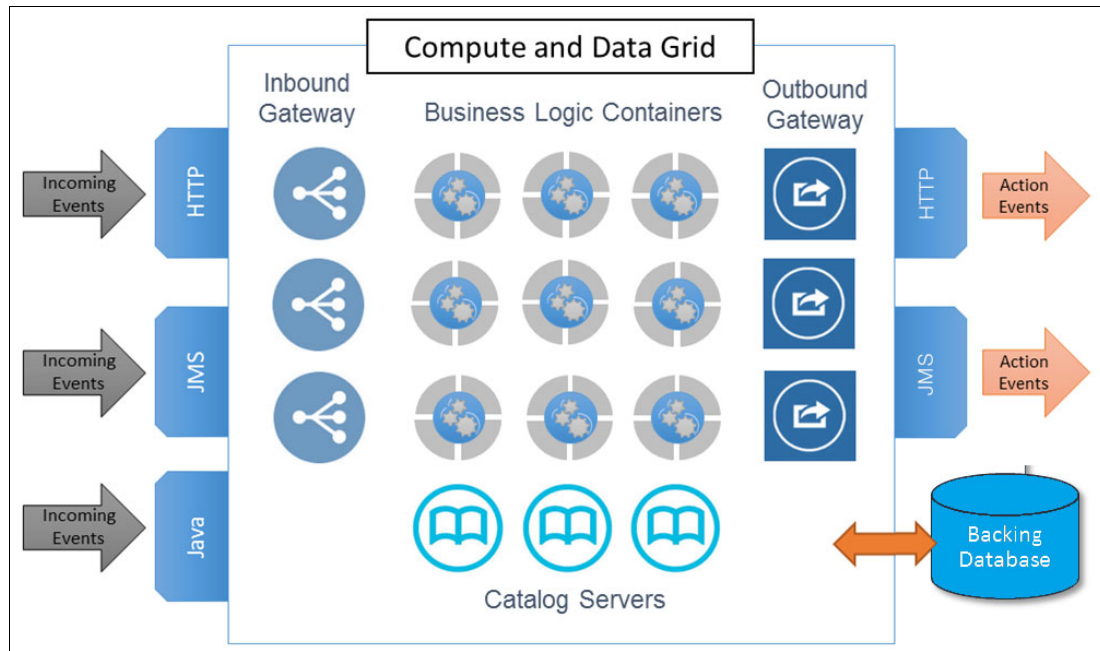


Figure 9-20 Decision Server Insights architecture,

As discussed above, it is important to have an understanding of what sorts of things are going to affect the performance (and therefore sizing) of your solution. This is especially important because some of the most significant factors are decided when the solution is being developed rather than at run time. This section discusses what these factors are, and why they are important.

9.4.1 Designing for performance

Some choices can be made at a solution developer level that influence the performance of a DSI solution. Therefore, this section includes some general tips for designing your solution for performance.

Store only what you need

Much of the potential excess computing cost (that is, processing above and beyond rule execution) in DSI comes from storing, accessing, and updating information in the grid. This potential cost is only increased when replication is taken into account. Therefore, it is important to only store data that is absolutely required to have enough context to make your decisions. These are some examples of what can be done:

- ▶ Set your time horizon appropriately: This can be done at the solution level, agent level, and per agent type per event level.
- ▶ If some events are not required in your event history based on a simple condition, set up a filter in the agent descriptor to discard them.
- ▶ If some events are not required in your event history based on a complex condition, set up a stateless Java Agent to filter them.
- ▶ If there are some extraneous fields in your inbound events, remove these, whether at the client side or with a transformation in the inbound connectivity.
- ▶ Store only the required fields in an entity.

Use Java agents for simple logic

If you are doing a simple operation with an agent, for example updating the fields on an entity, where the business logic is reasonably static over time, you will get improved performance when it is implemented as a Java agent. However, it is important to carefully consider this, because the benefits of the rule agent, including auditability by business users and ease of updating the business logic, are lost.

Prefer more rules over more agents bound to the same entity

Having multiple rules on one agent rather than splitting those rules over multiple agents with the same bound entity eliminates some of the costs associated with a transaction from happening multiple times (specifically, loading event history, engines, and entities from the grid). Therefore, wherever possible, consolidate your rules into one agent per entity type and use packaging to keep them distinct for governance.

Take advantage of performance optimizations

There are a couple of paths in DSI that are optimized for good performance, so use these in your solution unless they affect functionality:

- ▶ Global event aggregates with no reference to time, or using “during the current minute/hour/day” rather than “during the last period of x minutes/hours/days”. These have been optimized to eliminate the requirement to store all the events during the referenced period.
- ▶ Referencing a list of multiple remote entities (that is, entities that are not the bound entity). If the operation is to add an entity to the list that does not already contain it, then this operation can be optimized by using the “has several different” construct in the BMD, and then not doing an explicit presence check in the agent. This approach compares IDs to see whether the entity is already in the list rather than resolving all the entities and then comparing.

9.4.2 Configuring for performance

This section describes general tips for configuring your DSI servers for performance.

Consider your high availability (HA) requirements carefully

Adding synchronous replicas adds a significant performance cost, so if replicas are not required, you can expect a significant performance improvement.

Event cache duration tuning

If you have a low throughput but a long time horizon, or long lived but rarely referenced objects, use the in memory grid as a cache for the backing database can dramatically decrease the memory sizing requirements of a DSI deployment. This option involves evicting events from the in-memory grid after a specified time period, but retaining them in the backing database. This means that if an operation occurs that requires these events, they are reloaded from the backing database. After the time horizon, they are deleted as normal.

For more information about configuring the event cache, see the IBM Knowledge Center at:

http://www.ibm.com/support/knowledgecenter/SSQP76_8.7.1/com.ibm.odm.itoa.admin/topics/tsk_modify_event_cache.html

JVM monitoring

The volume of data that passes through the memory in DSI means that the load on the garbage collection (GC) of the JVM can be significant. Therefore, it is useful to capture the verbose GC logs (JVM option `-verbose:gc`) and monitor them. Overall heap utilization should also be monitored because a heap that is under pressure can cause issues.

9.5 ODM Decision Server Insights topologies

Designing a production topology for your Decision Server Insights platform requires careful consideration of the necessary qualities of service and a fair understanding of your target solution. To assist with your planning, this section describes the building blocks of a Decision Server Insights topology, a minimum configuration, and a suggested topology to provide high availability. It concludes by describing a process for selecting and sizing your solution.

Note: This section is an introductory guide only. Always consult an ODM specialist before deciding on a deployment configuration.

9.5.1 Component definitions

At a high-level, a DSI platform is divided into four different components that each provide different capabilities.

Inbound connectivity

This component handles requests from source systems, submits inbound events to the compute grid, and handles initial filtering and routing of events according to the agent descriptors.

Outbound connectivity

Outbound connectivity is responsible for delivering outbound events to event destinations as specified in the connectivity definition file.

Runtime server

The runtime server is where the computation happens. Agents in the runtime server use their available context to respond to incoming events and apply their rules. For the purposes of storing context in the data grid, the runtime server uses the following additional concepts:

- ▶ **Catalog service:** This tracks and manages the distribution of data and data replication in the grid.
- ▶ **Container:** This stores entities and event history, and runs agents and analytics jobs.

Backing database

This database instance is responsible for persisting state from the grid. The persisted state is used for multiple purposes that include:

- ▶ Restoring grid state after a system shut-down (intentional or not)
- ▶ Offloading old events for longer-term storage

The backing database is sometimes referred to as the disaster recovery database.

When combining these components (excluding the database, which this assumes is provided by an independent database platform), you have a choice of several containerization options.

Figure 9-21 illustrates the two most common options: Combined node and separate nodes.

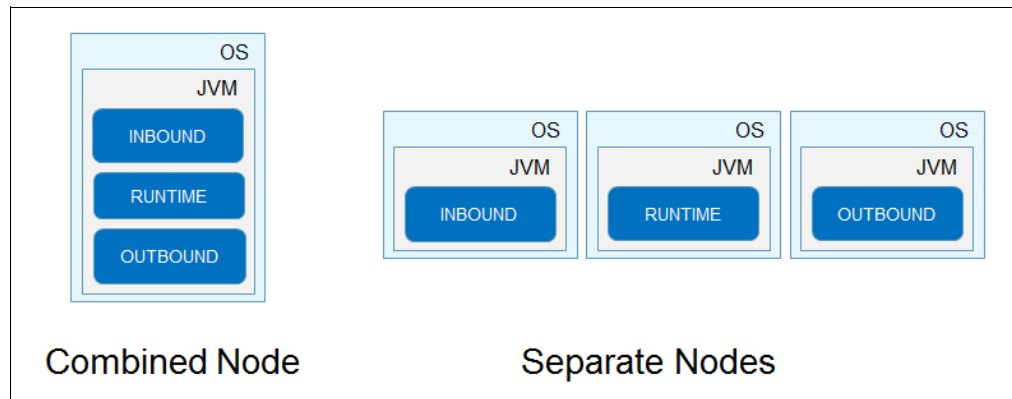


Figure 9-21 Comparing combination options for DSI topology components.

In a combined node, the inbound, runtime, and connectivity features are all installed in the same JVM: In fact, a single WebSphere Liberty Profile application server. This has the advantage of simplicity of management and scaling, as each server is self-contained and identical.

When using separate nodes, each of the components is hosted in separate JVMs and OS containers. This allows much more control because each capability can scale independently of the others (just adding more runtime servers, for example).

The descriptions that follow do not consider HTTP servers, load balancers, or messaging configurations that are external to DSI.

Note: When building your production topology for DSI, the combined node is not identical to the *cisDev* template. Although both contain the same components, they have a different configuration. You should build your combined node by adding the inbound and outbound connectivity and catalog components to a *cisContainer* template instead.

The separate node combination allows you to use the *cisContainer*, *cisCatalog*, *cisInbound*, and *cisOutbound* server templates directly. For more information about co-locating container and catalog servers, see “Step 2: Placement of connectivity servers” on page 226.

9.5.2 Minimum configuration

Figure 9-22 shows the minimum configuration for a DSI deployment.

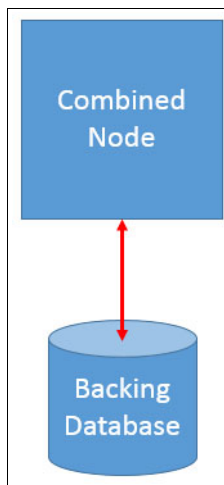


Figure 9-22 DSI minimum deployment configuration

Because you can have all components of the DSI run time configured within one combined node, you can have a fully functional DSI deployment with just one combined node with a backing database.

However, although this deployment is fully functional, if you lose the server to maintenance or some hardware or software failure, then you lose your DSI solution until it is restarted. The following section details the supported DSI configuration for high availability and continuous availability (HA and CA).

9.5.3 HA and CA configuration

A typical HA and CA configuration is shown in Figure 9-23.

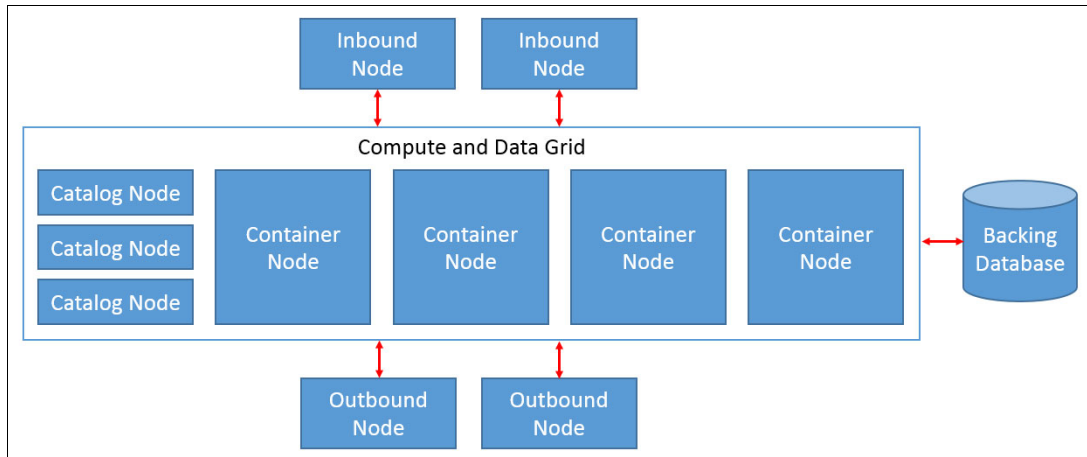


Figure 9-23 Typical HA and CA cluster

The explanation, justification, and options for modification are included in the following node descriptions:

- 3x catalog nodes

Three catalog nodes are included because during a failure on a switch, having three prevents a “split brain” scenario where your cluster splits into two independently functional clumps. It is possible to reduce to two if your machines are in the same room if this scenario is not a concern.

Note: To achieve a HA topology, the catalog service must be separated physically from the runtime nodes. Therefore, in Figure 9-23, the runtime nodes are referred to as *container nodes*, because they consist of only a container. There can be issues if a catalog and container are lost simultaneously, so splitting them up mitigates this risk.

- 4x runtime nodes

Four runtime nodes are included because this configuration has three replicas of the data: The primary, a synchronous replica, and an asynchronous replica. Therefore, it requires a minimum of three runtime nodes. The additional runtime node is a spare to allow for rolling maintenance.

It is possible to reduce the number runtime nodes to three, but every time you want to perform maintenance, you pay a performance or availability cost related to the elimination of replicas. When you have four servers, if one is shut down, you still have all three replicas maintained. If you have three servers only, you only have enough containers available for two replicas when one is shut down.

- 2x inbound and outbound nodes

Having two of each means that you can shut one down for maintenance, or be resilient during a single failure. The inbound and outbound nodes can optionally be collocated with the runtime or catalog nodes if you are hardware limited.

- Backing database

The backing database is required in all DSI configurations because the state is not persisted across grid sessions without it.

For more information about the DSI reference topology, see the developerWorks article *Install and configure a Decision Server Insights reference topology* at the following web link:

http://www.ibm.com/developerworks/bpm/bpmjournal/1503_defreitas1/1503_defreitas1.html

9.5.4 Selecting your topology

Given the definitions above, use the following decision flow to guide your design of a DSI topology.

Step 1: Quality of service

Your required quality of service is determined by which patterns described in 9.5.3, “HA and CA configuration” on page 225 that you use as your base configuration. In particular, you must answer these questions:

- ▶ Is continuous or high-availability required?
- ▶ Cost of additional hardware and software for replicas.
- ▶ What is the recovery-time objective?
- ▶ Impact of losing replicas after taking down a single node?

Although you might want to build a highly available system, in a virtualized environment you must ensure that replicas and HA pairs have sufficient isolation from each other. Often this is achieved by having host machines in different racks at different ends of the room, or in different rooms in the same building. Where you cannot ensure physical and logical isolation in a virtualized environment, choose a simpler topology (such as the combined node) and rely on redundancy elsewhere in the stack.

Step 2: Placement of connectivity servers

The following list describes scenarios that indicate that it might not be appropriate to install your connectivity servers on the same node as your runtime servers.

- ▶ You want to scale your connectivity servers independently of your runtime servers.
- ▶ You want to have dedicated connectivity servers for different source or destination systems (separate JMS and HTTP inbound servers, for example).
- ▶ You have a high throughput for inbound events, such that processing the events requires dedicated resources.

If you do choose to separate either or both types of connectivity servers, consider the following points:

- ▶ When placing connectivity servers on separate physical nodes, different licensing considerations can apply.
- ▶ Outbound servers typically have a smaller load than inbound servers.
- ▶ Sometimes you might want to co-locate your inbound HTTP servers with a web-server to reduce the server footprint without compromising the availability of your DSI grid.

Step 3: Placement of catalog servers

Collocating catalog servers with container servers allows for a more simple topology that makes server management easier. However, in scenarios where the solution provides a business critical function, separating components allows for greater resilience during node failure, as in the HA topology. Specifically, recovery is more difficult if you lose a container and a catalog server simultaneously. However, you can co-locate a catalog server with a connectivity server (typically an outbound server) to reduce the server footprint.

Step 4: Selecting “write-through” or “write-behind” for your database

Depending on your quality of service requirements, you have the choice of ‘write-through’ or ‘write-behind’ configurations for your backing database. These options correspond to synchronous and asynchronous writes to the database. This is configured at installation time. However, it is highlighted here because your choice can affect performance, or might result in data loss. To be clear, synchronous writes to the backing database on state changes reduce performance, and asynchronous writes allow for faster processing but uncommitted data will be lost during an unmanaged shut down of the grid.

Step 5: Scaling for throughput and storage requirements

You should now have your base topology defined, in terms of numbers of nodes and placement of servers among those nodes. Next, scale the number of nodes according to your solution’s throughput and storage requirements.

For example, you need to ensure that the memory requirements are sufficient to accommodate all events and entities (along with their replicas) for the appropriate amount of time. Additionally, the grid needs enough computational power to handle the event throughput volumes.

(Optional) Step 6: Accommodate scenarios with large memory requirements

If your scenario has a large memory requirement, the default configurations might not give optimum performance. This is because at Java heap sizes greater than around 64 GB per node, garbage collection can limit performance. To overcome this limitation, you have two choices:

- Use IBM eXtremeMemory (XM). This is a feature of WebSphere eXtreme Scale that allows Java applications to use a system’s native memory instead of the Java heap on some operating systems. By moving objects off the Java heap, you can avoid garbage collection pauses. This feature is not supported for use with the write-behind backing database or Windows operating systems at the time of writing.
- Alternatively you can use a topology that deploys multiple runtime servers per node (Figure 9-24). This topology allows you to continue scaling to larger memory systems where each combined node instance has up to a 64-GB Java heap.

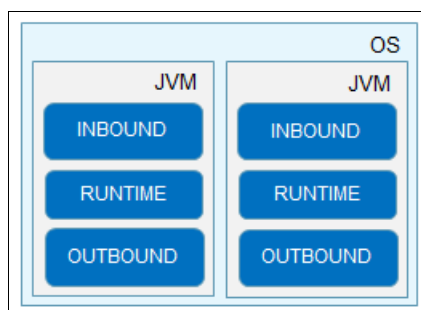


Figure 9-24 Multiple runtime servers per node

9.6 Transactionality and rollback

Integration of ODM DSI within a system of insight has some considerations that need to be taken into account when you design your deployment.

Consider the scenario where DSI accepts an event. This event is processed by a rule agent that makes a service call through an OSGi bundle, triggering an outbound event that goes to an external system such as described in Appendix A, “Integration of ODM Decision Server Insights with custom services” on page 231.

This scenario involves three separate transactions:

1. The event is received through inbound connectivity and added to the InboundQueue map (Figure 9-25).

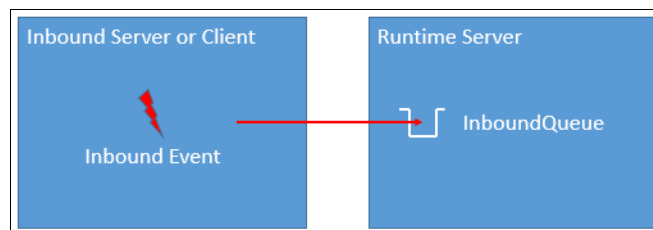


Figure 9-25 First transaction in the example

2. The event is processed by the agent and, if an outbound event needs to be sent, the outbound event is added to the OutboundQueue map (Figure 9-26).

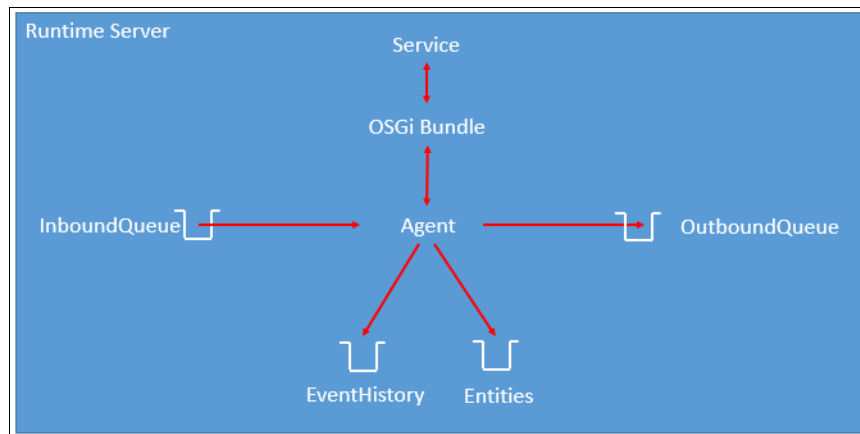


Figure 9-26 Second transaction in the example

3. The outbound event is delivered and removed from the OutboundQueue map (Figure 9-27).

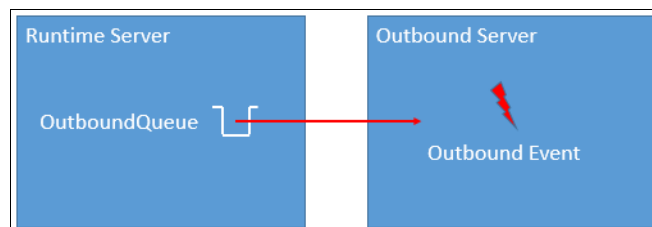


Figure 9-27 Third transaction in the example

All three of these transactions have integration points. In transactions 1 and 3, the rollback is taken care of by the in-built rollback mechanism within DSI. So, for example, if there is some problem putting the inbound event to the InboundQueue map, the transaction is rolled back and retried.

Similarly, if there is a problem submitting the outbound event, it rolls back the transaction and retried, retaining the event on the OutboundQueue.

However, in transaction 2, some solution developer implemented service calls introduce some concerns around transactionality. It is difficult to handle rollback in an external system through DSI because there is no way of determining at what stage of the transaction that it occurred.

Therefore, any external service call should have these characteristics:

- ▶ Be synchronous
- ▶ Have no side-effects (that is, it should be read-only)

This should avoid any issues over rollback and maintain transactionality of the system.

For any integration points where an external system would be affected (that is, anything that would come under the *Action Execution* category in a system of insight), the outbound connectivity layer is the preferred option.



Integration of ODM Decision Server Insights with custom services

ODM Decision Server Insights (DSI) runs on the WebSphere Liberty Profile platform. Therefore, you can take advantage of OSGi services to extend the core functionality of the product.

Consider the following examples of methods of integrating OSGi services with DSI:

- ▶ Extending DSI agent functionality
- ▶ API calls to other local runtimes
- ▶ Web services calls to external systems

This section describes implementation and deployment of an OSGi service as part of a DSI solution. This knowledge is used to perform a web services call to ODM Decision Server Rules to decide how best to deal with an airline customer who is going to miss their flight in 8.6, “Detect and decide example” on page 165.

This appendix is informed by an excellent developerWorks article by Jose De Freitas and Lewis Evans that is available at the following web address:

http://www.ibm.com/developerworks/bpm/bpmjournal/1503_defreitas2/1503_defreitas2.html

This appendix covers the following topics:

- ▶ OSGi service implementation
- ▶ XOM to BOM mapping

A.1 OSGi service implementation

This section describes a method to create the OSGi service using the following steps:

1. Setting up
2. Creating the OSGi bundle project
3. Coding the service
4. Configuring the manifest
5. Configuring the blueprint

A.1.1 Setting up

To create an OSGi bundle project in Insights Designer (or any Eclipse installation), you need the IBM WebSphere Application Server Developer Tools feature installed. You can obtain the feature through the IBM Installation Manager for DSI by clicking **Modify**, then selecting **WebSphere Application Server Developer Tools**. Follow the instructions in the IBM Knowledge Center to complete the installation:

http://www.ibm.com/support/knowledgecenter/SSQP76_8.7.1/com.ibm.odm.itoa.install/topics/tsk_install_adding_wdt.html

A.1.2 Creating the OSGi bundle project

After you complete the setup, create your OSGi bundle by completing the following steps:

1. From Insights Designer, click **File** → **New** → **Other**, and search for **OSGi Bundle Project** in the list of available project wizards. See Figure A-1.

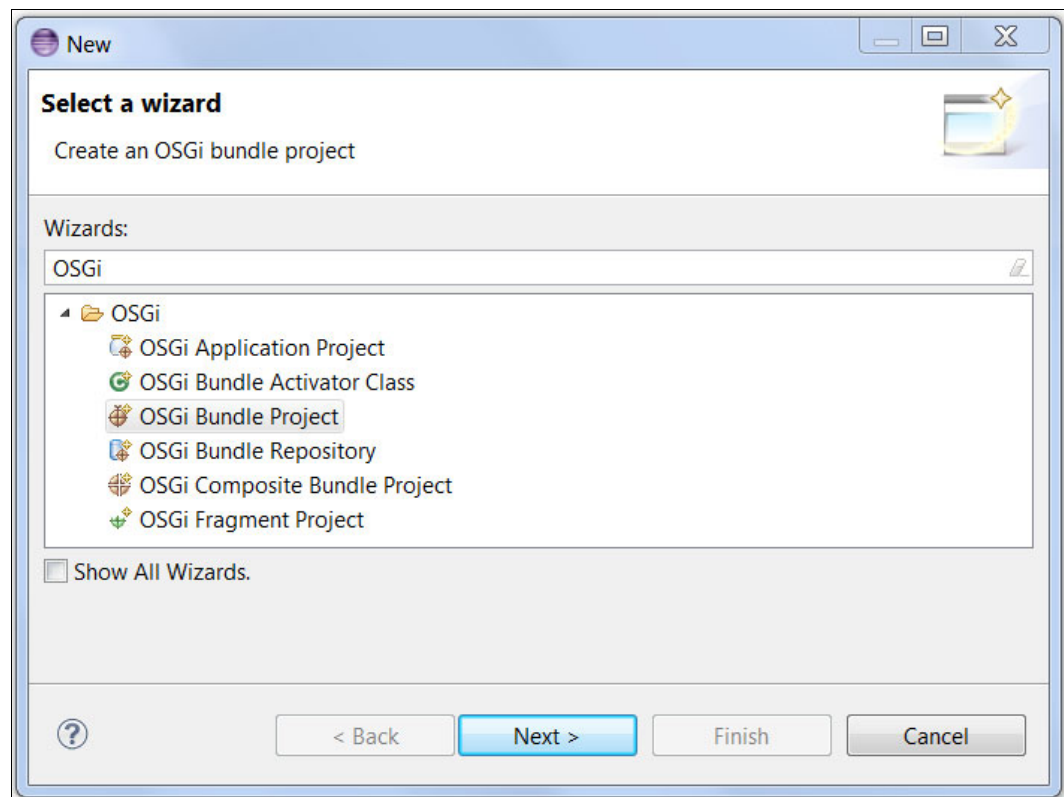


Figure A-1 Wizard selection dialog

2. On the first page of the wizard, give your project a name, such as “example_service”, select **Generate blueprint file**, and clear **Add bundle to application** as shown in Figure A-2.

The screenshot shows the 'New OSGi Bundle Project' dialog box. The title bar says 'New OSGi Bundle Project'. The main heading is 'OSGi Bundle Project' with a subtitle 'Create a standalone OSGi bundle project or add it to a new or existing application.' and a small icon of a bundle. The 'Project name' field contains 'example_service'. The 'Project location' section has 'Use default location' checked, and the 'Location' field shows 'C:\Users\IBM_ADMIN\workspaces\Redbook\example_service' with a 'Browse...' button. The 'Target runtime' is set to '<None>' with a 'New Runtime...' button. The 'Configuration' section has four unchecked checkboxes: 'Add Web support' (with 'Web 2.5' selected), 'Add persistence support' (with 'JPA 2.0' selected), 'Add EJB support' (with 'EJB 3.1' selected), and 'Custom' (with 'Advanced...' selected). The 'Generate blueprint file' checkbox is checked. The 'Application membership' section has 'Add bundle to application' unchecked, and the 'Application project' field shows 'example_project.app' with a 'New Application...' button. The 'Working sets' section has 'Add project to working sets' unchecked, and the 'Working sets' field is empty with a 'Select...' button. At the bottom are buttons for '< Back', 'Next >', 'Finish' (highlighted in blue), and 'Cancel'.

Figure A-2 OSGi bundle project wizard first page with correct options selected

3. On the final page of the wizard, make sure that the **Bundle root** is blank, so that your META-INF and OSGI-INF folders are placed at the project root as shown in Figure A-3.

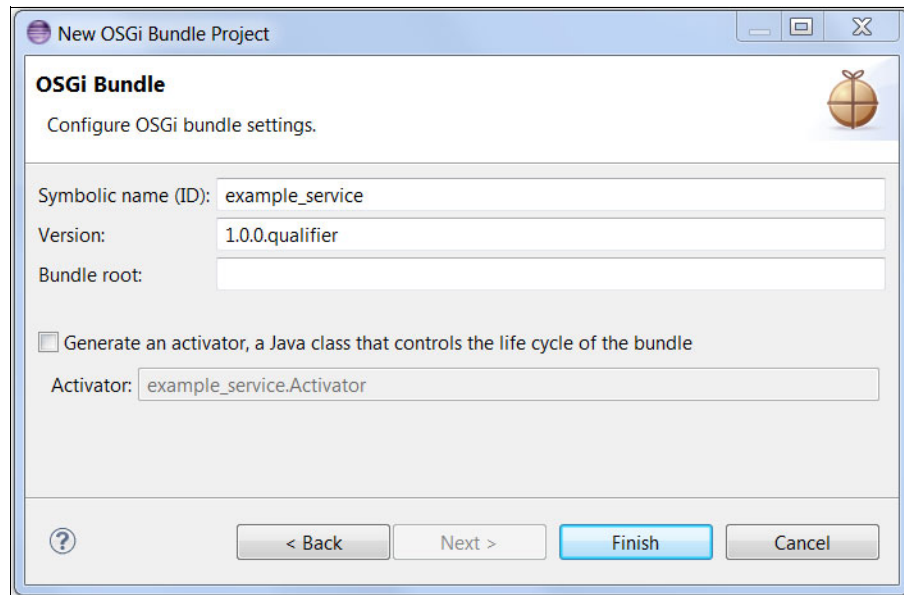


Figure A-3 Final page of the OSGi bundle project wizard with correct options selected

4. Complete the wizard by selecting **Finish**.

A.1.3 Coding the service

Create the Java packages for the service interfaces and implementations, such as `com.ibm.rdbook.service.example.api` for the interface, and `com.ibm.rdbook.service.example.impl` for the implementation.

Create the interface (Example A-1) and implementation (Example A-2) classes in their respective packages. In this example service, you call out to a black box function that returns the promotion code for a set of parameters.

Example A-1 An example service interface

```
package com.ibm.rdbook.service.example.api;

public interface ExampleService {

    public String getPromotionCode(String[] params);

}
```

Example A-2 shows the implementation class.

Example A-2 An example service implementation

```
package com.ibm.rdbook.service.example.impl;

import com.ibm.rdbook.service.example.api.ExampleService;
import com.ibm.rdbook.example.BlackBox;

public class ExampleServiceImpl implements ExampleService {
```

```

BlackBox bb = new BlackBox();

@Override
public String getPromotionCode(String[] params) {
    return bb.promote(params);
}
}

```

A.1.4 Configuring the manifest

To configure the manifest, complete the following steps:

1. Open **META-INF/MANIFEST.MF**.
2. Add any required dependencies to the Imported Packages (Figure A-4).

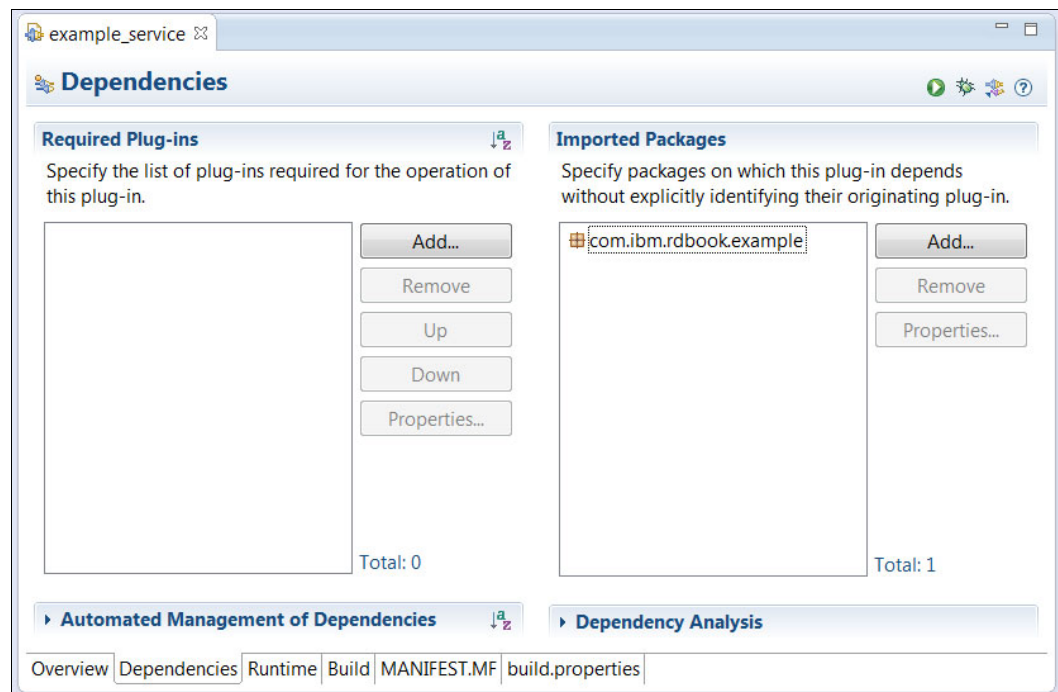


Figure A-4 Importing service dependencies with the manifest editor

3. Set up the build properties (Figure A-5).

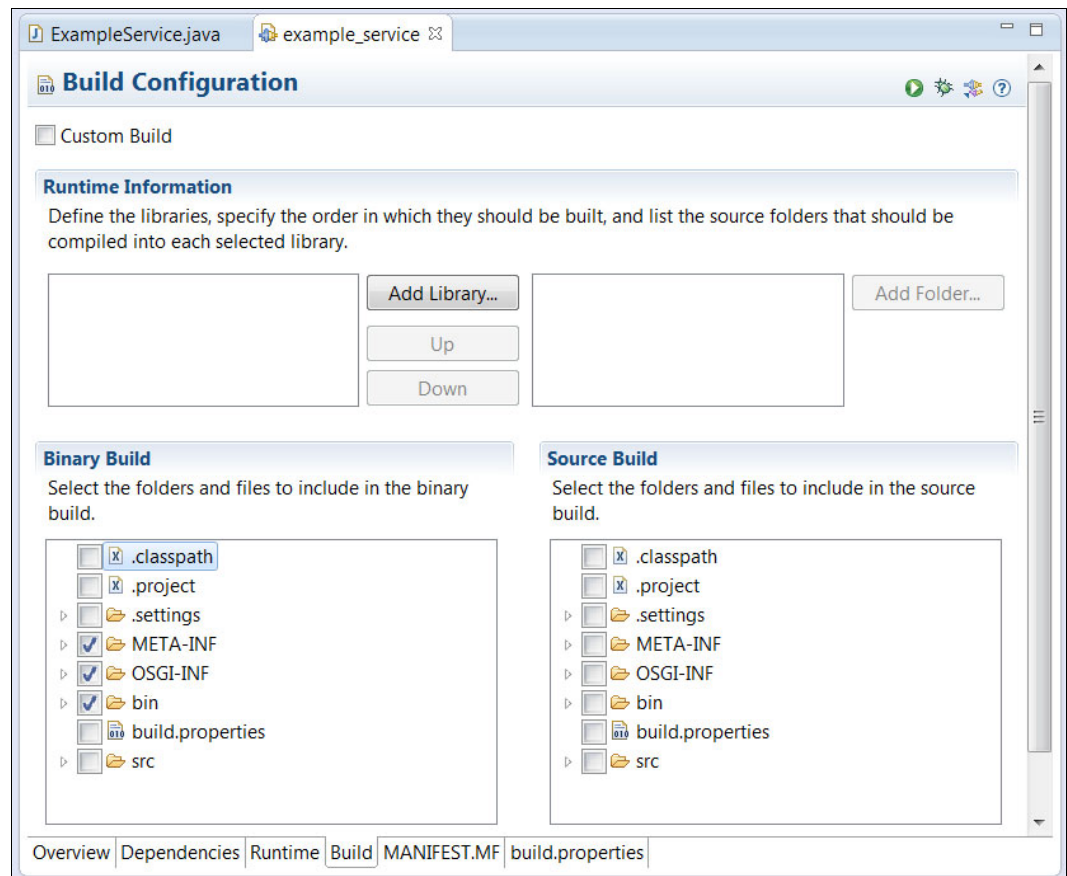


Figure A-5 Setting up build properties with the manifest editor

4. Export the service interface (Figure A-6).

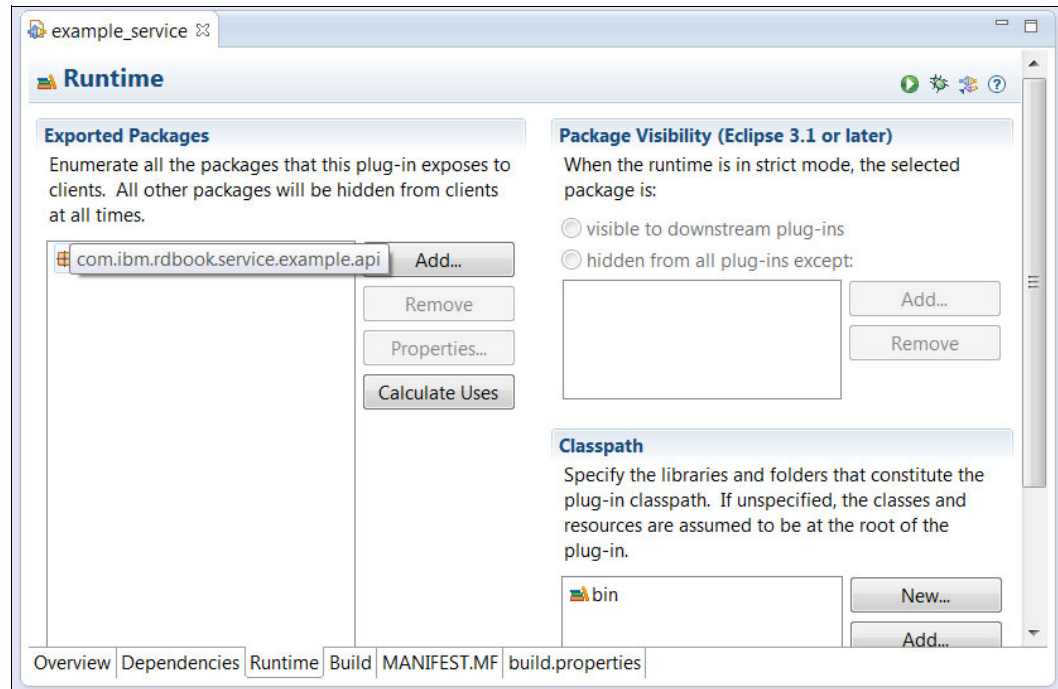


Figure A-6 Exporting the service interface with the manifest editor

A.1.5 Configuring the blueprint

To configure the blueprint, complete the following steps:

1. Open **OSGI-INF/blueprint/blueprint.xml** and, from the Design tab in the Eclipse Blueprint XML editor, click **Add** and select **Service** (Figure A-7).

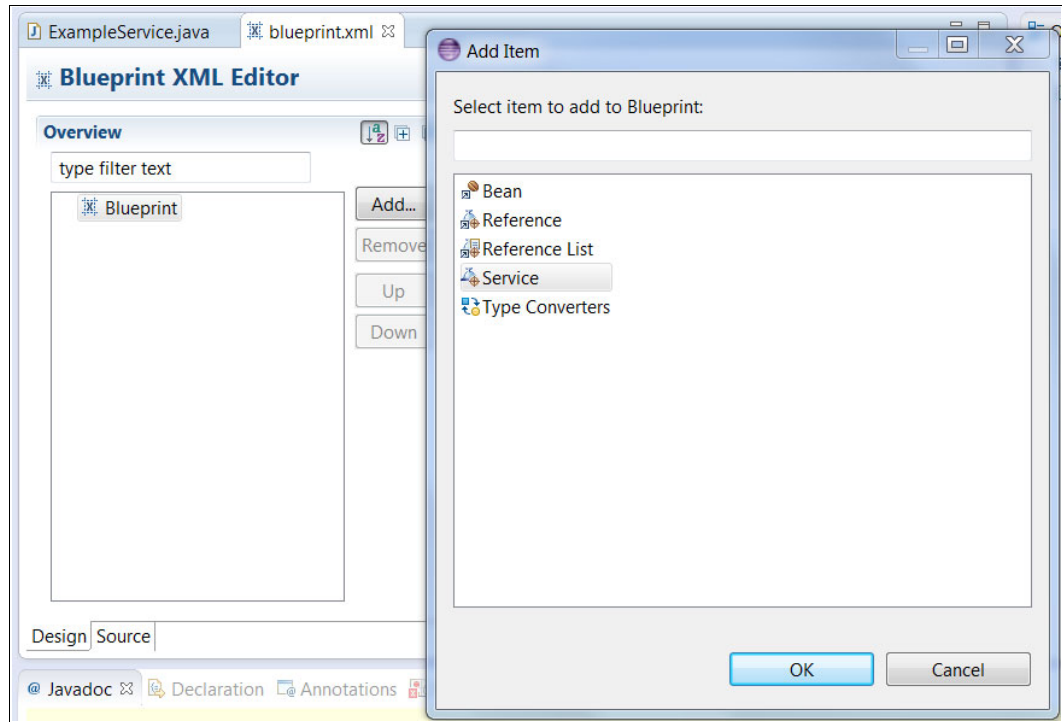


Figure A-7 Adding a service to the blueprint with the blueprint XML editor

2. Select the service interface, give it a sensible name, and create the implementation reference bean (Figure A-8).

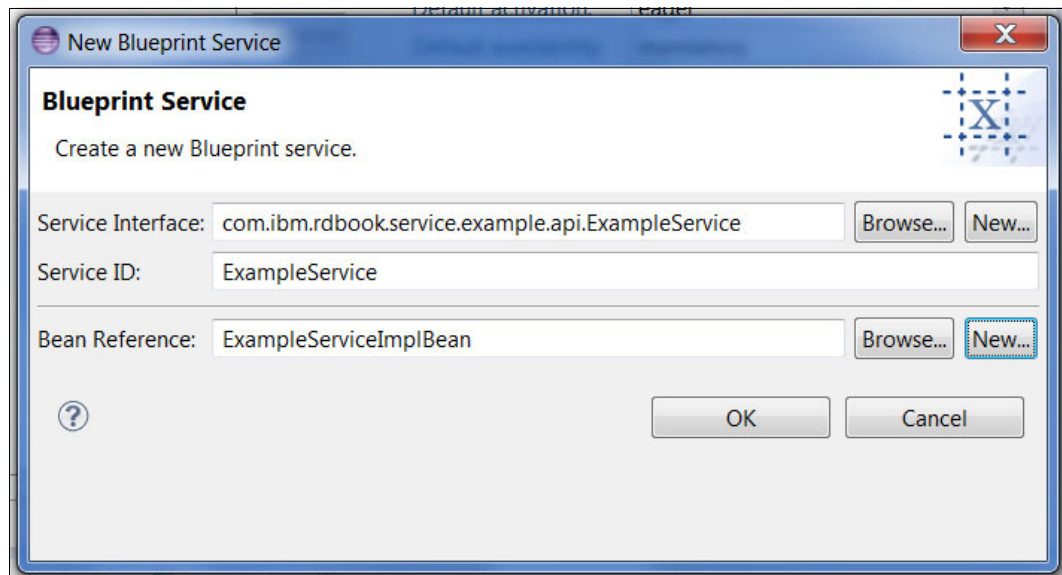


Figure A-8 Configuring the service with the blueprint XML editor

With that, you have completed defining the OSGi service.

A.2 XOM to BOM mapping

Now you need to provide a mapping between this OSGi service (which is treated as the execution object model (XOM)) and a business object model (BOM) that allows you to refer to the service in the rules language.

A.2.1 Creating a rule project

Complete the following steps to create a rule project:

1. In Insights Designer, click **File** → **New** → **Other**, and search for **Rule Project**. See Figure A-9.

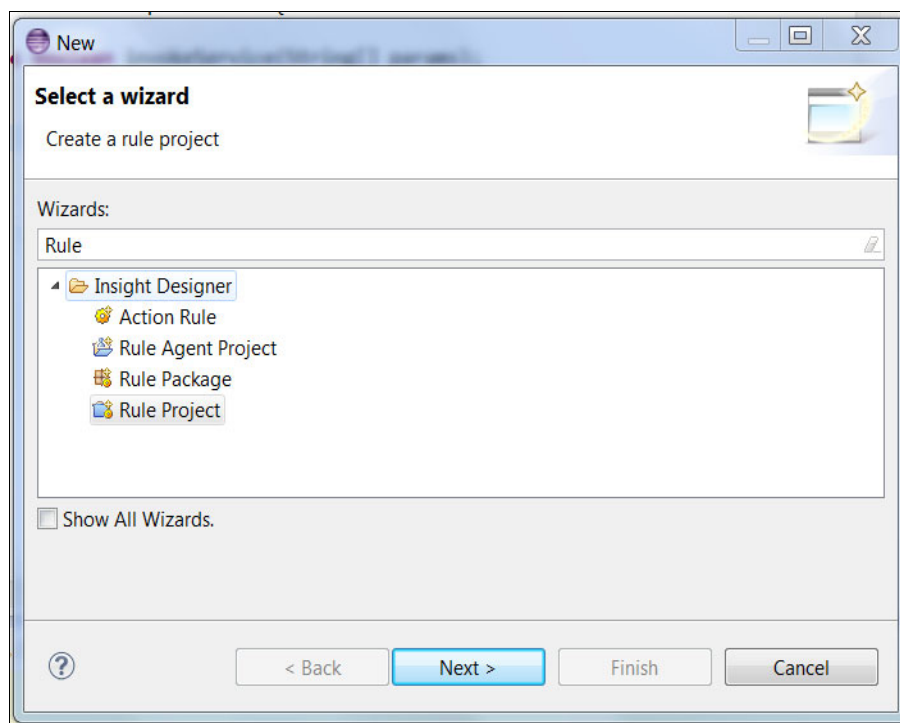


Figure A-9 Opening the Rule Project wizard

2. Give the rule project a sensible name, for example “example_service_bom”, and accept the defaults until you arrive at the Rule Project XOM Settings window. Select your OSGi service project as a required Java project (Figure A-10).

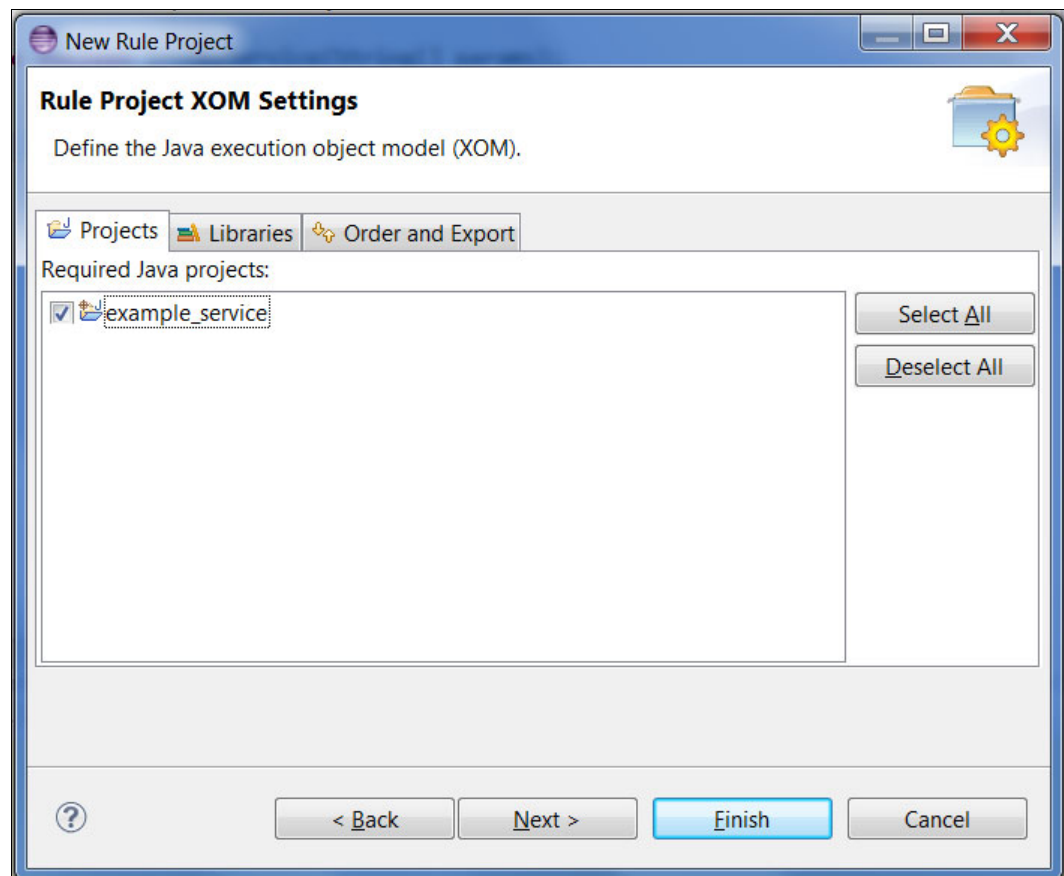


Figure A-10 Selecting the OSGi service project as a required Java project

3. Click **Finish** and your Rule Project is created with the correct configuration.

A.2.2 Creating a BOM entry

To create a BOM entry in the Rule Project, complete the following steps:

1. Click **File** → **New** → **Other** in Insights Designer and search for **BOM Entry** (Figure A-11).

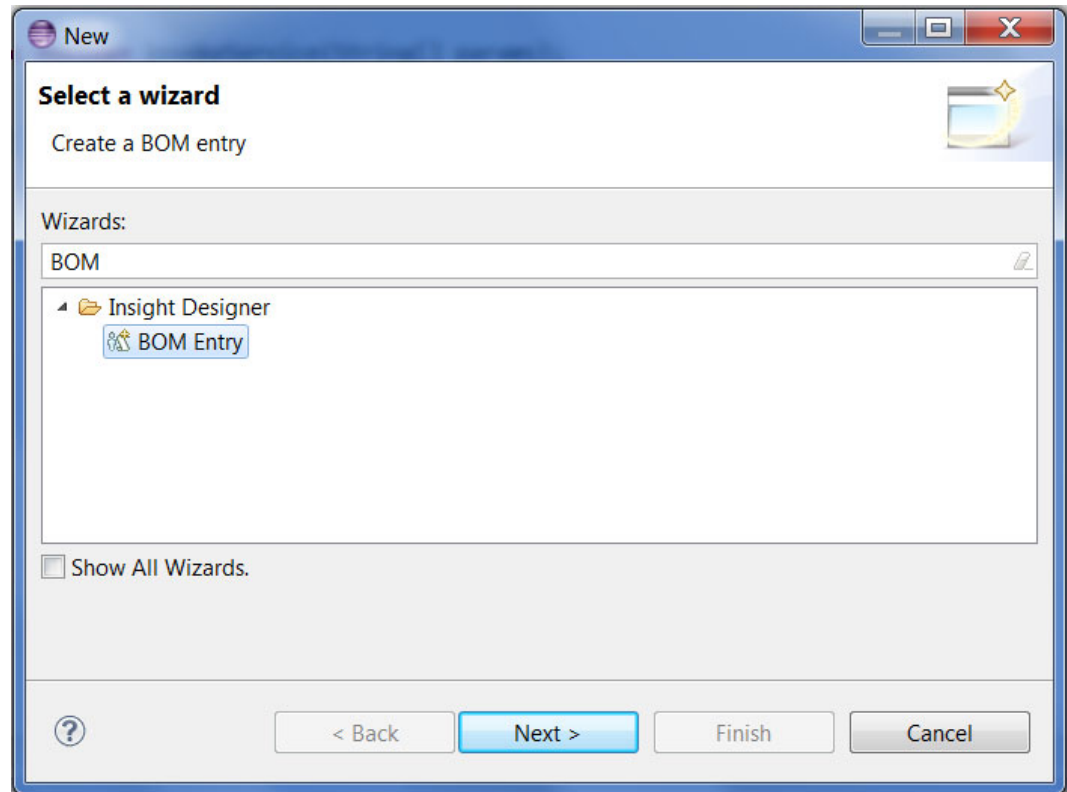


Figure A-11 Opening the BOM entry wizard

2. Browse XOM entries and select the OSGi service XOM, and then select the service interface (Figure A-12).

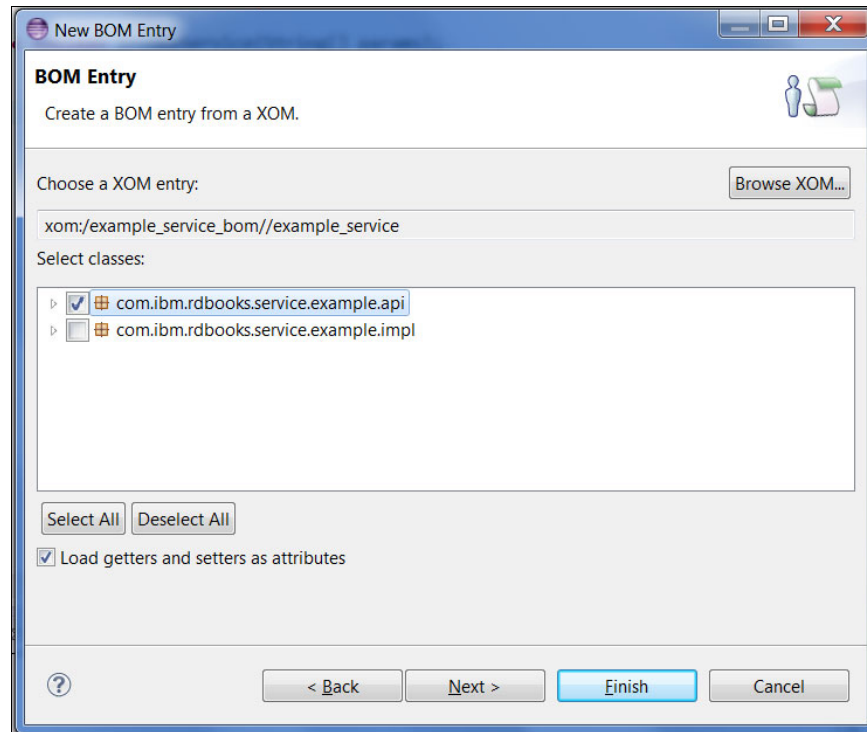


Figure A-12 Choosing the XOM entry to create the BOM entry from

3. Click **Finish** to complete the step.

A.2.3 Configuring the BOM entry

There are a couple of configuration steps that are required for the BOM entry to start the XOM correctly.

To start this process, complete the following steps:

1. Open the BOM file that you created in the previous section, and navigate down the package tree to your service implementation class on the Design tab. Double-click this class to open the Class tab.
2. On the Class tab, expand the **Custom Properties** section (you might have to scroll down to see it). Click **Add** and define a property called **OSGi.service** that points at your service interface (Figure A-13).

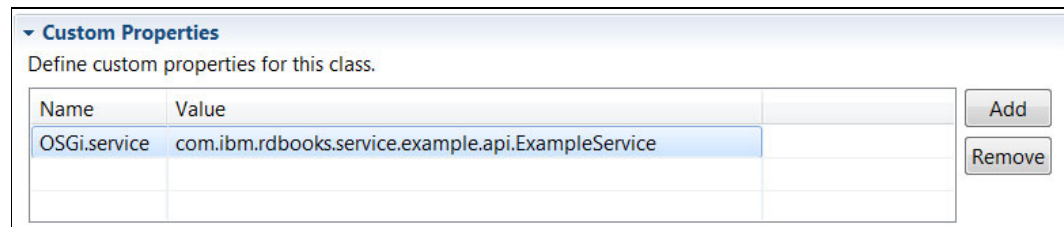


Figure A-13 Adding the OSGi.service custom property to the BOM entry

Important: The OSGi.service property and its value are case-sensitive.

3. Double-click your service method name on the left side of the Class tab. This opens the Member tab. From here, select **Static** (Figure A-14) to make the service method call static.

The screenshot shows the 'Member getPromotionCode (class: com.ibm.rdbooks.service.exan)' dialog. Under the 'General Information' section, the 'Name' field contains 'getPromotionCode', the 'Type' field contains 'java.lang.String', and the 'Class' field contains 'com.ibm.rdbooks.service.example.api.ExampleService'. There are 'Browse...' buttons next to the 'Type' and 'Class' fields. Below these fields, there are four checkboxes: 'Static' (checked), 'Final' (unchecked), 'Deprecated' (unchecked), and 'Update object state' (unchecked).

Figure A-14 Making the service method call static

A.2.4 Verbalizing the service call

To have your service fully available for use in DSI, you must provide a verbalization of the service call for use in Rule Agents. If you are still in the Member tab on the BOM Entry after the previous section, you are already in the correct place. Otherwise, return there.

Under the Member Verbalization section, click **Create**. Give the call a sensible verbalization, being careful to retain the necessary placeholders for function arguments (Figure A-15). It is useful to consider how your verbalization will read in a rule when determining what this should be. In the example, your rule could read something like “if the promotion code for a customer with “GOLD ACCOUNT” is “PROMOTION8” then”.

The screenshot shows the 'Member Verbalization' section. It has a title bar with a dropdown arrow and the text 'Member Verbalization'. Below the title bar, there are two links: 'Remove the verbalization.' (with a red 'x' icon) and 'Create a navigation phrase.' (with a green '+' icon). Below these links, there is a section titled 'Navigation : "the promotion code for a customer with strings"' (with a red 'x' icon). Under this section, there is a 'Template:' label followed by a text input field containing 'the promotion code for a customer with {0}'. There is a small icon to the right of the input field.

Figure A-15 The custom verbalization for the service call

A.2.5 Final setup

The final configuration step is to right-click the Rule Project, and select **Properties**. Click the Rule Engine tab and select **Decision engine** (Figure A-16).

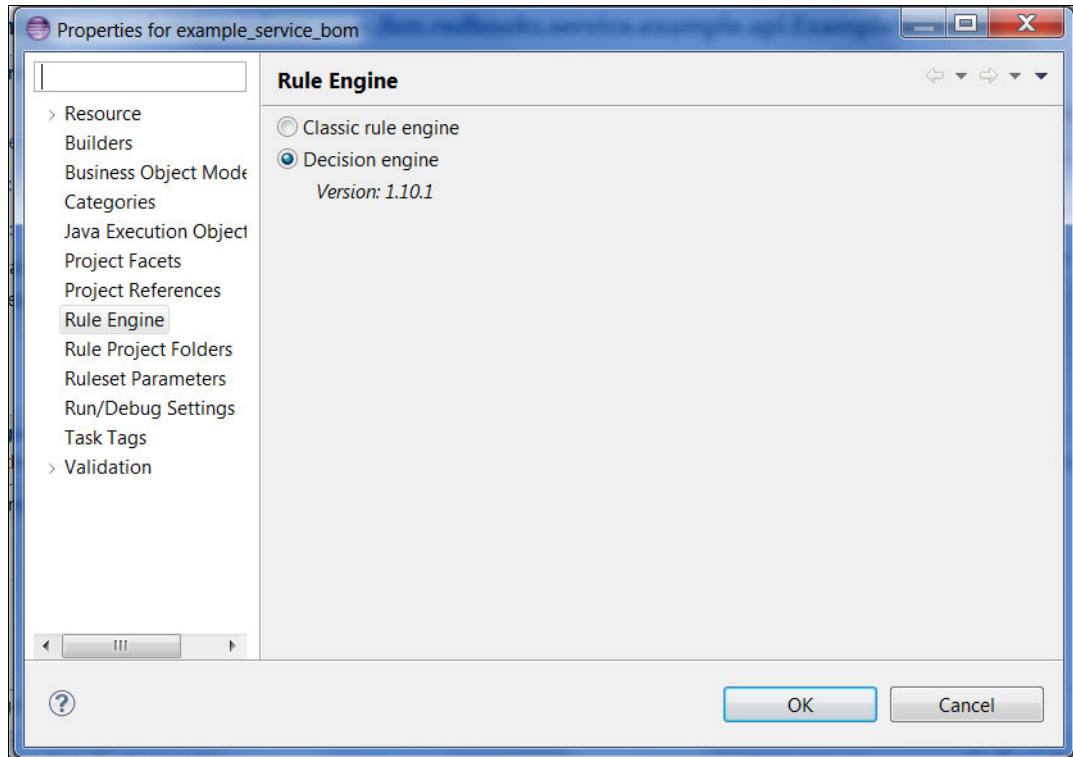


Figure A-16 Selection of the correct rule engine

Your OSGi service is fully set up and ready to use. If you are interested in seeing an example of a practical use for this capability, see 8.5, “Analyze and report example” on page 143 where ODM Decision Server Rules are started from a DSI Rule Agent.

Related publications

The publications listed in this section are considered particularly suitable for a more detailed discussion of the topics covered in this book.

IBM Redbooks

The following IBM Redbooks publications provide additional information about the topic in this document. Note that some publications referenced in this list might be available in softcopy only.

- ▶ *Flexible Decision Automation for Your zEnterprise with Business Rules and Events*, SG24-8014
- ▶ *IBM Operational Decision Management V8.0 Performance Tuning Guide*, REDP-4899
- ▶ *Using IBM Operational Decision Manager: IMS COBOL BMP, COBOL DLIBATCH, and COBOL MPP*, REDP-4997

You can search for, view, download or order these documents and other Redbooks, Redpapers, Web Docs, draft and additional materials, at the following website:

ibm.com/redbooks

Online resources

These websites are also relevant as further information sources:

- ▶ *Event Driven Architecture and Decision Management*
<https://developer.ibm.com/odm/docs/odm-capabilities/odm-advanced-decision-server-insights/event-driven-architecture-and-decision-management/>
- ▶ IBM Business Process Manager V8.5.6
http://www.ibm.com/support/knowledgecenter/SSFPJS_8.5.6/com.ibm.wbpm.ref.doc/rest/bpmrest/rest_bpm_wle_v1_process_post_sendmessage.htm
- ▶ IBM DataPower Gateway
<http://www.ibm.com/software/products/en/datapower-gateway>
- ▶ IBM Integration Bus
https://www.ibm.com/support/knowledgecenter/SSMKHH/mapfiles/product_welcome.html
- ▶ IBM Internet of Things Foundation
<https://internetofthings.ibmcloud.com/#/>
- ▶ IBM MessageSight
<http://www.ibm.com/software/products/en/messagesight>
- ▶ IBM Operational Decision Manager 8.7.1
http://www.ibm.com/support/knowledgecenter/SSQP76_8.7.1/com.ibm.odm.distrib/kc_welcome_odm_distrib.html?lang=en

- ▶ InfoSphere Streams
<http://www.ibm.com/software/products/en/infosphere-streams/>
- ▶ *Integrating Hadoop and IBM Operational Decision Manager*
http://www.ibm.com/developerworks/bpm/bpmjournal/1209_crowther/1209_crowther.html
- ▶ *Install and configure a Decision Server Insights reference topology*
http://www.ibm.com/developerworks/bpm/bpmjournal/1503_defreitas1/1503_defreitas1.html
- ▶ *Simplify complex code with OSGi services in Decision Server Insights rules*
http://www.ibm.com/developerworks/bpm/bpmjournal/1503_defreitas2/1503_defreitas2.html
- ▶ *Think big! Scale your business rules solutions up to the world of big data*
http://www.ibm.com/developerworks/bpm/library/techarticles/1411_crowther-bluemix/1411_crowther.html

Help from IBM

IBM Support and downloads

ibm.com/support

IBM Global Services

ibm.com/services

Redbooks

Systems of Insight for Digital Transformation

SG24-8293-00

ISBN 073844118X



(0.5" spine)

0.475" <-> 0.873"

250 <-> 459 pages



SG24-8293-00

ISBN 073844118X

Printed in U.S.A.

Get connected

