Redbooks
ibm.com/redbooks

Machine Learning with Business Rules on IBM Z: Acting on Your Insights

Chris Backhouse

Stéphane Faure

David Griffiths

Mike Johnson

Yann Kindelberger

Ke Wei Wei

Hao Zhang

Redpaper

Machine Learning with Business Rules on IBM Z: Acting on Your Insights

This IBM Redpaper™ introduces the concept of operational decision management based on machine learning (ML) and covers these topics:

Important changes since this document was written:

This document was written for an older release of Operational Decision Manager for z/OS (ODM for z/OS). ODM for z/OS 8.9.1 required the writing of custom Java code to access a Watson Machine Learning for z/OS Scoring Service (this can be seen in “Java implementation within ODM” on page 19). Since that time ODM for z/OS version 8.10.1 has been released and much improves the integration experience. Integrating the two products no longer requires custom Java code. Using ODM for z/OS 8.10.1 or later you can use an automated wizard in the ODM tooling to:

- Browse and select a model from Watson Machine Learning
- Import the Machine Learning data model into your rule project
- Automatically generate a template rule that integrates a call to the Watson Machine Learning scoring service

Should I read on?

Continue to read this document for:

- Individual introductions to ODM for z/OS and Machine learning
- Discussions on the benefits of using the two technologies together
- Information on integrating if you have not yet updated to ODM for z/OS 8.10.1

Follow this link to learn about the machine learning integration in ODM for z/OS 8.10.1

https://www.ibm.com/support/knowledgecenter/en/SSQP76_8.10.x/com.ibm.odm.zos.develop/topics/con_ml_integration_overview.html

Overview

Digital computing is rapidly changing our world and by 2020 analysts expect there to be 75 billion connected devices in the world. The users of these devices have ever more demanding expectations. These users want the companies to provide service that was unheard of 20 years ago. They expect to be able to consume services right now, wherever they are, and tailored to their personal needs. Mobile, social, analytics, cloud, and cognitive technologies are transforming the way that we do business.

The key to successfully accomplishing this transformation is knowing how to best leverage the assets you already have and are betting your business on. It is reported that 90% of the world’s data has been created in the last 2 years. It is further reported that 80% of the world's corporate data sits on, or originates from, the mainframe. Mainframes process 30 billion transactions a day and enables $6 trillion in card payments each year. Clearly then, you cannot talk about digital transformation without talking about the mainframe and how organizations can exploit the rich value in their existing IBM® Z® investments.

There is much talk about how to expose these valuable Z assets to the rest of the business through technologies such as IBM z/OS Connect Enterprise Edition as RESTful APIs. However, what good is an API if the application that is sitting behind that API is not business relevant or agile enough to meet the demands of its customer base?

The key to this problem is Business Decisions, which are the implementation or operationalization of the business policies of your company. How you make them, how you change them, who changes them and how fast you can change them is the key to a successful digital business. It is the Business Decisions within the applications that give these powers over the customer experience:

- To differentiate your company from others
- To differentiate one customer’s experiences from another
- To maximize the likelihood that a customer to chooses to do business with your company.

There are many ways to implement the decisions and business policies in a company. Historically these decisions have been coded into applications that are written in languages such as COBOL on the mainframe. Over the last 10 years, the use of Business Rule Management Systems, or more recently termed Decision Management Systems, have established themselves as a key technology in this arena. More recently, though, we have started to see the emergence of Artificial Intelligence (AI) based technologies. Machine Learning is key AI technology that has reached maturity and is finding its way into more and more business applications.

This leaves us with a difficult question to answer. Which of these seemingly competing technologies should you be choosing to implement your Business Decisions. Or should you be choosing at all?

This IBM Redpaper^{TM} introduces the concepts of a decision management system and IBM Operational Decision Manager for z/OS. Then, we introduce these topics:

- Key concepts of Machine Learning for real-time scoring in a z/OS environment
- IBM Machine Learning for z/OS as the key implementation of this technology on z/OS

We then compare and contrast the different approaches to decisions and show how — rather than being competing technologies — they are actually complimentary, each playing a key part in successful decisioning on z/OS. The article concludes with an example of how you can integrate the two technologies at run time to create truly responsive agile business decisions for z/OS applications.

The need for a solution

Every industry is struggling to keep pace with change, whether it is driven by regulatory, market, or customer forces.

For example, a bank might have a lending policy that states, “Customers whose credit rating is above average are entitled to a discounted rate on their loan.” The traditional development lifecycle to implement at change to this policy requires work by these staff members:

- A business analyst to document the detailed requirements and to design and develop this policy
- An architect to identify which applications on which platforms have implementations of that business policy
- One or more teams of developers take those requirements and interpret them into an application design
- One or more teams of developers to code and embed the decisions into the various applications.
- The application development is then followed by a lengthy testing process.
- Co-ordination to ensure that the new business policy rolls out uniformly across all platforms and applications.

Unfortunately, the decision logic is now hidden in the code of one or more applications potentially across different platforms and different development teams. Over time, as changes are added to the business policy, the code becomes more complex, making changes and auditability increasingly difficult.

Also, there is a challenging gap between the business team that owns the business policy behind the decisions and the final implementation of the decision in the application. It is difficult to verify with the policy owners whether the decision has been implemented correctly.

One solution to the problem is to externalize key parameter values of the decision. In the above example, this might be the level of discount that is offered to a customer. This approach can work in some cases, but its success is limited. Changes to the decision are restricted to the values that have been externalized by the application development team. Often a completely new application is required to manage the changes and lifecycle of those externalized values.

A much better solution is to use decision automation, externalizing the logic that implements the decision from the application. This solution allows for changes to the business policies to be made in a much more focused and agile way that is separate from traditional application changes. The approach that is shown Figure 1 has many advantages.

- When making a change to the decision, it is far easier to identify where to make the change because all the automation of the decision in concentrated in one place.
- By externalizing the decision, it can now be shared between multiple applications and potentially across multiple platforms. As a result, less development work is needed to implement changes to the decision.

- If the decision is separated from the application, it can be implemented in a way that allows subject matter experts to be directly involved in the management and maintenance of the decision.
- With the implementation of the decision external from the application, we can also reduce the amount of testing effort that would be required for a change to the decision. With the decision encompassed as part of the application any changes within the application would require extensive testing to the application. The tests must ensure that there have been no unwanted side effects to the application caused by the change of code. With the decision externalized the testing can be scoped down just to the changes to the decision. Then, the only testing that is required for the rest of the application is at the points where it calls the externalized decision.

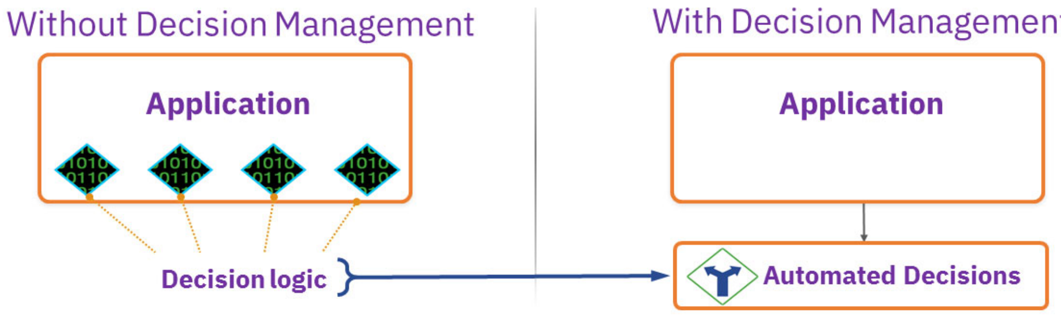


Figure 1 Decision Externalization

Operational Decision Manager for z/OS for Business Rules

This section introduces the concepts of Operational Decision Manager for z/OS (ODM) as a solution for writing, managing, and maintaining business rules for use in context of a business decision.

What is Operational Decision Manager for z/OS?

IBM Operational Decision Manager for z/OS is a comprehensive decision automation platform that helps you capture, analyze, govern, and automate rules-based business decisions with optimized runtime options for integration with z/OS applications.

With IBM Operational Decision Manager for z/OS, IBM offers a comprehensive and easy-to-use platform that allows users to capture, automate, and manage frequently occurring, repeatable business decisions as business rules. By externalizing decision logic from COBOL, PL/I and Java code this solution makes it possible for business users to collaborate with IT easily and consistently in the adaptation of the applications that are essential to business operations on z/OS.

Enabling rules-based, decision logic to be defined and maintained by nontechnical subject-matter experts, IBM Operational Decision Manager for z/OS brings these benefits:

- Single place of truth to access, share, and update business decisions
- Better alignment between the business and the IT teams
- Improved quality of decisions and the speed with which they can be changed
- Traceability on how decisions are made within the applications
- Tools for users, administrators, and developers to edit and manage rules
- Powerful decision server to execute business decisions
- Robust decision repository to centrally host the business decisions
- Sharing of business decisions across applications and platforms

ODM use cases

The following are sample use cases and solutions that can help you realize a faster return on investment with Operational Decision Manager for z/OS

- Risk and counter fraud detection across credit cards and banking
- Automation of underwriting decisions in the insurance industry
- Automation of claims processing in the insurance industry
- Intelligent operations and crew scheduling in the airline industry
- Optimized and personalized pricing and promotions in the retail industry

Why Operational Decision Manager for z/OS?

Operational Decision Manager for z/OS business rules applications bring these key benefits:

- Development and deployment of business-critical applications that can keep pace with market requirements
- Increased participation by the business in application development and updates
- Enablement of business policy updates by individual members of a team. These updates are managed by a single tool and deployed to multiple platforms
- For IT organizations, implementation of application-modernization projects by incrementally externalizing their business decisions from z/OS applications to ODM.
- Execution of business rules as part of your business decision that is closely coupled to the existing z/OS applications for the best performance and system management
- Native support for COBOL, PL/I, and Java-based applications
- High performance, low-latency runtime components to execute rule-based decision logic on mainframe systems with optimized access from CICS, IBM, and z/OS batch applications
- Multiple execution alternatives for closest integration with existing application architecture
- Support full interoperability and sharing of decisions across IBM Operational Decision Manager offerings on Public Cloud, Private Cloud, x86, Power, and Linux on Z
- Integration with IBM Z DevOps infrastructure
- Java-based rule execution, which lowers costs by offloading the heavy lifting of the business rules execution to IBM Z Integrated Information Processors (zIIP)

Using Operational Decision Manager for z/OS

The key to using operational decision management is the ability to express the business rules in a natural language format. That way, both the IT and the business teams can understand them. This vocabulary is fully customizable and comes from the Business Object Model (BOM) which is created by the tooling from the eXecution Object Model (XOM). The XOM describes the format of the data that will be passed in from the calling application. The key principle is that the BOM allows you to abstract away from the IT nature of the data and application. As a result, you can write rules that are understandable to the business. Figure 2 shows the relationship between a simple XOM, vocabulary, and business rule.

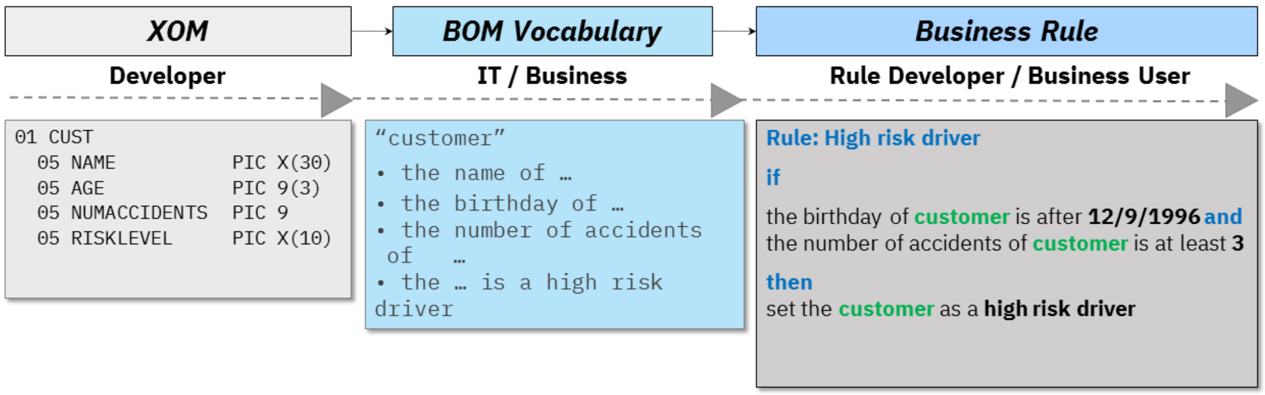

XOM
BOM Vocabulary
Business Rule
Developer
IT / Business
Rule Developer / Business User
01 CUST
05 NAME PIC X(30)
05 AGE PIC 9(3)
05 NUMACCIDENTS PIC 9
05 RISKLEVEL PIC X(10)
"customer"
• the name of …
• the birthday of …
• the number of accidents of …
• the … is a high risk driver
Rule: High risk driver
if
the birthday of customer is after 12/9/1996 and the number of accidents of customer is at least 3
then
set the customer as a high risk driver

Figure 2 Relationship of XOM, BOM Vocabulary, and Business Rules

The XOM can come from any of the following sources

- COBOL copybooks
- PL/I header files
- Java classes
- XML Schema (dynamic execution model)

These are imported to the ODM tooling to create the BOM and vocabulary. Rules are written using the vocabulary of the BOM. Multiple rules (often thousands of rules) are bundled together in a Decision Project. The Decision Project is deployed to a Rule Execution Server (RES) as a Decision Service. The RES handles the requests for a particular decision from the application based on runtime data. This runtime data is passed as input parameters to the RES. The RES can simultaneously execute multiple requests, from multiple clients, for multiple decisions, and even multiple versions of the same decision.

Using ODM allows the decisions to be separated from the client code. When the decisions need to be changed due to business requirements, it is only these decisions and not the application that need to be changed. New and changed rules are automatically redeployed to the rule servers. You don't have to change the application code.

The decision applications can be developed in any of these environments:

- An Eclipse-based environment
- Rule Designer, which is targeted towards IT rule developers
- Decision Center, a web-based development and rule-repository environment that is targeted towards both IT rule developers and business users.

Figure 3 shows the relationship between the different components.

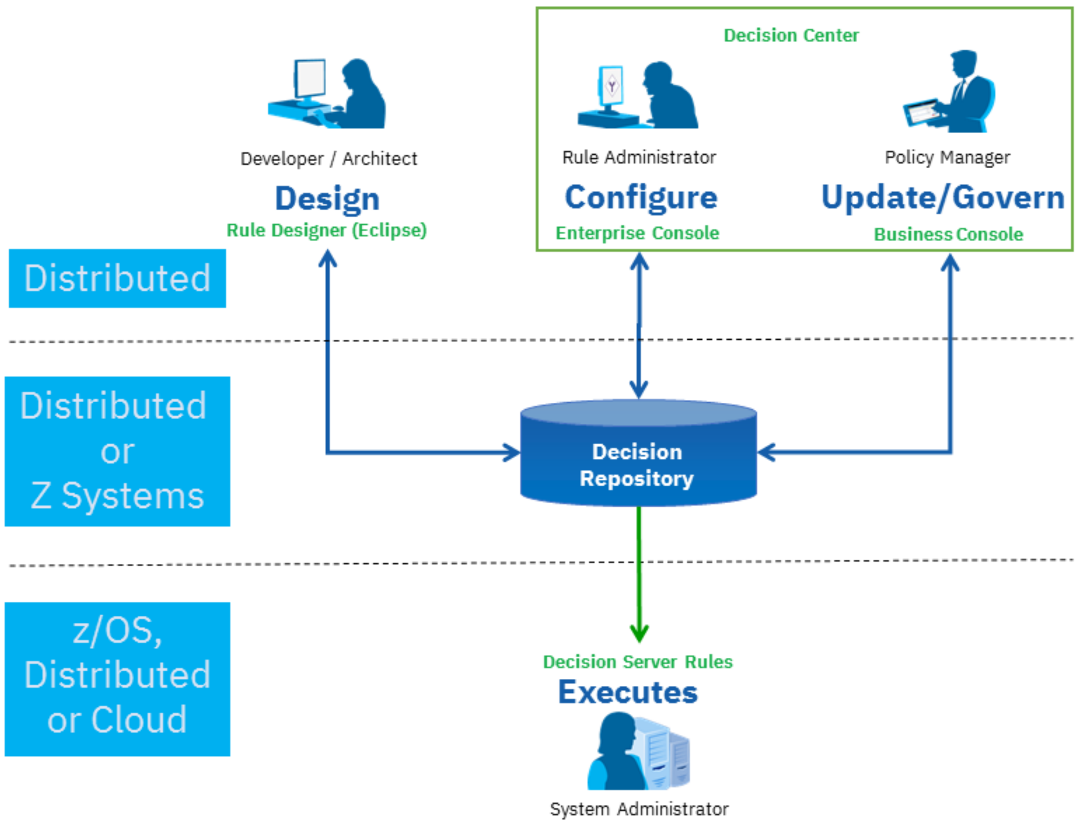


Figure 3 Operational Decision Manager Components

Rule Designer. The Eclipse-based development environment for IT Rule Developers. Here you can import the XOM and create the BOM and vocabulary. You can also write and test rules as part of a decision and deploy directly to Rule Execution Servers.

Decision Center – Enterprise Console. The web-based interface for administering the Decision Repository. Here, you see all the versions of rules and decision artifacts that are stored in the repository and manage the repository itself.

Decision Center – Business Console. The web-based interface for developing rules for both IT rule developers and business users. Here, you see all versions of rules and decision artifacts, author rules, test changes to rules, and deploy these rules to Rule Execution Servers.

Decision Center Repository. The database that stores the decision artifacts to allow sharing between rule authors and decisions.

Rule Execution Server. The rule runtime server that evaluates the decisions, based on the data passed in by the applications and the deployed business rules.

Running Decisions on z/OS

The business rules for the decision can run on any of these platforms. Whatever the platform, ODM uses the same core Rule Execution Server. This approach ensures that whichever platform runs your decision, you are assured that for the same input data you get the same output response. However, for z/OS we additionally support COBOL and PL/I languages and provide dedicated, optimized integration with other z/OS components. These z/OS optimized servers are referred to as the zRule Execution Server or zRES.

Figure 4 shows the optimized deployment options for z/OS.

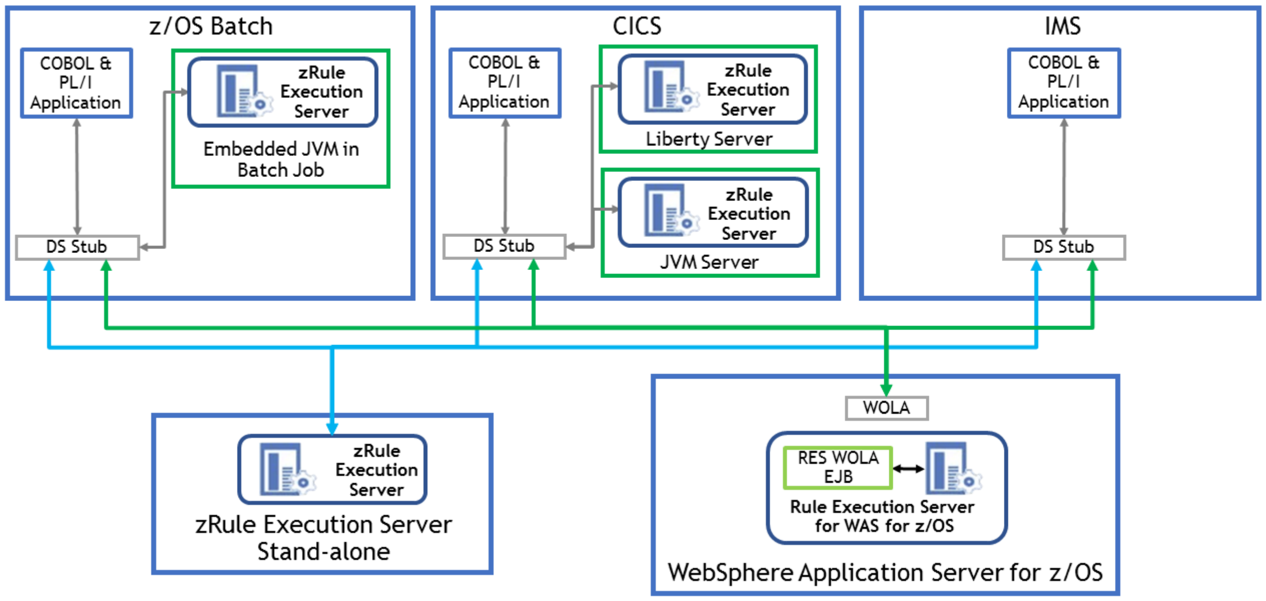


Figure 4 Deployment options for zRule Execution Servers

The client applications can be either z/OS Batch applications, CICS applications, or IMS Batch or Message Processing Region applications. (Java applications are also supported and execute in the same way as a Java distributed application that calls a standard ODM RES.) For each of the client types, ODM supplies a stub that is link edited into the calling application. The stub provides the simple API that the application can use to call to a zRES. The stub is responsible for routing the request to the configured server. All communication between the stubs and the server environments is optimized to be either in process or cross memory to ensure optimum performance.

The zRES servers can run in any of the following configurations:

- **zRES Stand-alone**: A dedicated started task address space for executing rules
- **WebSphere Application Server**: Deployment to both full and Liberty profile
- **CICS JVM/Liberty**: For CICS applications, the zRES can be hosted within the CICS Java environment
- **Embedded batch**: For long running z/OS batch jobs, ODM can start an instance of a zRES in process

More detailed descriptions of the deployment options are provided here:

- Redbooks - `http://www.redbooks.ibm.com/redbooks/pdfs/sg248014.pdf`
- IBM Knowledge Center - `https://www.ibm.com/support/knowledgecenter/en/SSQP76_8.9.2/com.ibm.odm.zos/kc_welcome_odm_zos.html`

Machine Learning for z/OS for real-time scoring

This section introduces the process of configuring IBM Machine Learning for z/OS for real-time scoring.

What is Machine Learning for z/OS

IBM Machine Learning for z/OS is an end-to-end, enterprise, machine-learning platform. It helps you create, train, deploy, and monitor machine-learning models to extract value from your mission-critical data on IBM Z, while keeping the data where it resides.

With IBM Machine Learning for z/OS, IBM offers a complete machine-learning solution that can extract hidden value from enterprise data. This solution helps organizations quickly ingest and transform data to build, deploy, and manage high-quality, self-learning, behavioral models using IBM Z data. Because the data is securely in place and processes in real time, you can more accurately anticipate customer and business needs.

Machine learning use cases

IBM Machine Learning for z/OS enables:

- Quick model development, tuning, and deployment
- Continuous monitoring and audit on model performance
- Proactive notification if model quality degrades, with the option to automatically retrain a model with feedback data
- Easy management of thousands of models
- The following are sample use cases and solutions that can help you realize a faster return on investment with Machine Learning for z/OS:
  - Enhanced credit card fraud detection for fast deployment using the most current data for modeling with flexibility to adjust as fraud patterns evolve
  - Real-time analytics to prevent improper claims payments before they are paid out
  - Automated loan decisions to decrease loan failure rates while growing revenue from more approvals
  - Optimized patient experience both at point of care with a healthcare provider and point of sale at the pharmacy
  - Reduce outages by monitoring IBM Z generated IT data and logs and predicting trends of key performance indicators

Why Machine Learning for z/OS

The key benefits of running machine-learning applications on z/OS are as follows:

- Gain new insight with machine-learning technology from the current and historical data on mainframe.
- Combine insight from structured and non-structured data from Z and non-Z data resources.
- Deliver in-transaction predictive analytics capability with real-time data.
- Minimize the latency, cost, and complexity of data movement.
- Maintain security and governance of data for analytics at lower cost.
- Leverage the strength IBM Z to provide incomparable available, reliable, scalable, and high-performance machine-learning services.
- Simplify development and deployment tasks significantly based on IBM Z DevOps infrastructure, so mainframe developers and administrators can leverage machine learning easily.
- Integrate closely with hardware and software in IBM Z stack, such as IBM Open Data Analytics for z/OS, WebSphere® Liberty Profile for z/OS, and CICS TS.
- Support interoperability across IBM Machine Learning / Data Science Experience offerings on Public Cloud, Private Cloud, x86, Power, and Linux on Z.
- Provide built-in solution templates for typical mainframe use cases.

- Lower cost by offloading the machine-learning workload to IBM Z Integrated Information Processors (zIIP).

Using Machine Learning online scoring services

IBM Machine Learning for z/OS (MLz) has a two-tier architecture, as shown in Figure 5.

- Some Machine Learning for z/OS components are containerized and orchestrated by Kubernetes, which runs on Linux on distributed platform or Linux on Z.
- Other components run on z/OS, because they need to be colocated with applications to meet high-availability requirements and service level agreements for real-time prediction.

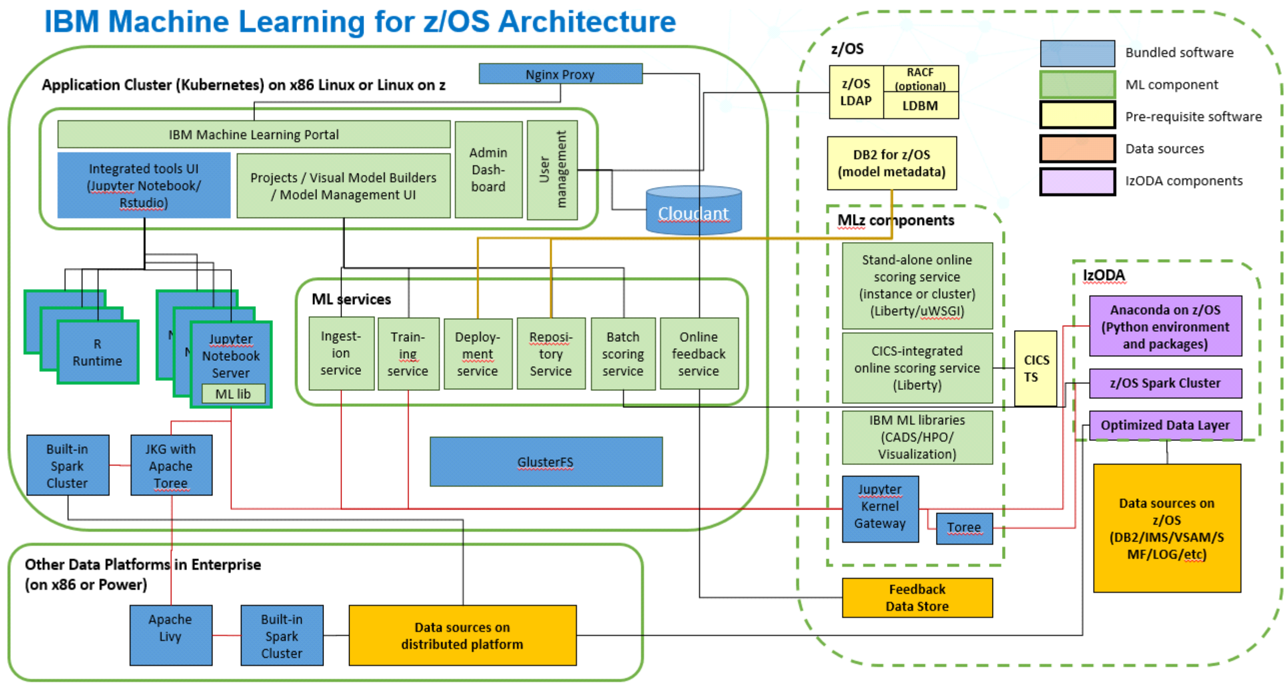

IBM Machine Learning for z/OS Architecture
Application Cluster (Kubernetes) on x86 Linux or Linux on z
Nginx Proxy
IBM Machine Learning Portal
Integrated tools UI (Jupyter Notebook/ Rstudio)
Projects / Visual Model Builders / Model Management UI
Admin Dash-board
User management
Cloudant
R Runtime
Jupyter Notebook Server
ML lib
ML services
Ingest-ion service
Train-ing service
Deploy-ment service
Reposi-tory Service
Batch scoring service
Online feedback service
Built-in Spark Cluster
JKG with Apache Toree
GlusterFS
Other Data Platforms in Enterprise (on x86 or Power)
Apache Livy
Built-in Spark Cluster
Data sources on distributed platform
z/OS
z/OS LDAP
RACF (optional)
LDBM
DB2 for z/OS (model metadata)
MLz components
Stand-alone online scoring service (instance or cluster) (Liberty/uWSGI)
CICS-integrated online scoring service (Liberty)
IBM ML libraries (CADS/HPO/ Visualization)
Jupyter Kernel Gateway
Toree
CICS TS
Feedback Data Store
IzODA
Anaconda on z/OS (Python environment and packages)
z/OS Spark Cluster
Optimized Data Layer
Data sources on z/OS (DB2/IMS/VSAM/SMF/LOG/etc)
Bundled software
ML component
Pre-requisite software
Data sources
IzODA components

Figure 5 IBM Machine Learning for z/OS architecture

Machine Learning for z/OS integrates tools, libraries, and runtimes for data scientists to develop, evaluate, and test machine-learning models. These integrations include proprietary tools such as these:

- Visual Model Builder for organizations that do not write code
- Cognitive Assists for Data Scientists (CADS) to accelerate model selection and hyper parameter tuning

Also, Machine Learning for z/OS integrates open source tools that data scientists are familiar with, such as Jupyter Notebook, R, or Apache Spark.

Besides an integrated development environment for data scientists, Machine Learning for z/OS offers RESTful services to manage the lifecycle of machine-learning models. These services extend from data ingestion and model training, to deployment and performance monitoring. You access and manage the services directly through Machine Learning portal UI.

Machine Learning for z/OS online scoring service

There are different approaches of using machine-learning models in applications. For example, a model that identifies fraud credit card transaction should be embedded in an online application. When the prediction result of the model shows that a transaction is suspicious, the transaction should be blocked. In this case, the model needs to be deployed to an online scoring service.

Machine Learning for z/OS online scoring service runs in Liberty for z/OS, which can be a stand-alone Liberty server or CICS Liberty JVM server.

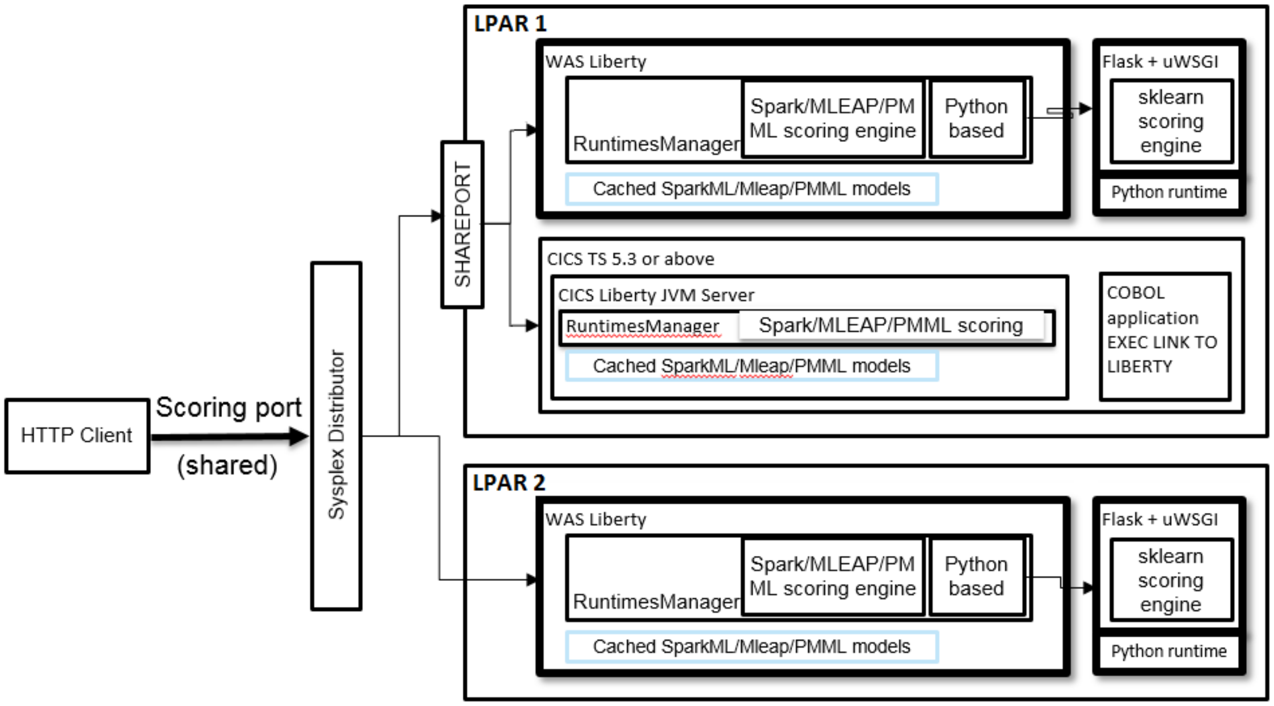


Figure 6 Flexible deployment options of Machine Learning for z/OS online scoring services

Applications can use models through online scoring REST APIs. COBOL applications can also use the EXEC LINK call to access the online scoring service in the same CICS region.

Bringing it together

If you compare “ODM use cases” on page 5 and “Machine learning use cases” on page 9, you see a large amount of overlap. With both capabilities having a role as decisioning technologies, it can be confusing to choose an optimal approach for a particular decision.

Business Rules

The approach taken by business rules to decisioning is very *prescriptive.* In the prescriptive approach, explicit rules are evaluated against the incoming data. Different algorithms might determine how exactly those rules are applied, but it is a very deterministic approach because someone who knows the rules can know what response the rules will return. This is both the strength and the weakness of business rules.

With legislation such as GDPR, businesses are being called on to describe what they are doing with end users' data, and how they are processing that data to arrive at a decision. With business rules, it is easy to show how the companies’ policies are being implemented at runtime against the user’s data. Also, where business rules are implementing policies for government regulatory requirements, having prescriptive, easily readable rules that implement those regulations gives a very easily, and auditable, way to show compliance.

However, a decision implemented by using business rules can only be as perceptive as the person who writes the rules. Consider the example of writing a decision regarding whether to lend someone money to buy a car. You might observe the following strange trend: *People who are buying yellow cars are far more likely to default on their loan*.
Unless you do research regarding that aspect of the loan applications details and the likelihood of loan defaults, the rules that you write cannot cope with that situation. This is where Machine Learning can enhance the business rules.

Machine Learning

Machine learning uses past data to take is a *predictive* approach to make a calculated guess about a potential future occurrence. There is great power in this approach when you have the historic data to use as a unique data source for future decisions. Returning to the example above of the car loan, if the historic data that you capture includes color of the car, the learning algorithm can easily detect the higher tendency for people who buy yellow cars to default on their loans.

However, historical factors alone cannot determine the entire decision. Again regarding the preceding example, what would happen if a 12-year-old applied for a loan to buy a car? We might unwisely base the entire decision on the historical data only. When we ask machine learning what the likelihood is of a 12-year-old defaulting on a loan, the prediction would likely show that there is no risk of a 12-year-old defaulting on the loan. The history shows that it never happened; no 12-year-old ever defaulted on a loan. Yet, this example is obviously nonsensical. Age of the applicant would not be the only historical factor that the machine learning considered. And other checks would be in place to stop the loan application from getting this far. Nonetheless, this example illustrates both the power and the limitations of historical data.

Also at the highest level, even if you have the best possible model the response from any machine learning predictive model can be accurate to only a certain percentage. You mitigate this limitation with a learning feedback loop, which detects when the accuracy of a model deteriorates. But even with that safeguard, you cannot know with 100% certainty that the response from machine learning is valid.

Strengths and weaknesses of Business Rules and Machine Learning

So, do we conclude that both approaches are flawed, and neither is suitable for use in critical business decisions? Well, obviously not. Each approach has strengths that independently add value in many use cases.

Approach	Strength	Weakness
Business rule	It can explicitly implement our business policy and it is very *easy to audit the reasoning behind the result returned by the rules*.	Rules can only be as perceptive as the person who has authored the rules. If there aren't rules to check for a particular condition, that condition does not form part of the business decision.
Machine learning	Ability to account for conditions across historic data. This ability is only limited by the range, amount, and quality of historic data that you can give it.	It doesn't implement and enforce our business policies. It is also a 'black box' in that it is difficult to audit the *reasoning that went into the particular score* that is assigned to a particular input to the model.

However, it is more rewarding to look at how the two approaches complement each other. In combining the two technologies, the relative weaknesses of each approach are counterbalanced by the strengths of the other. Therefore, this paper advocates the combined approach. This approach provides the best features of both technologies, so that you get the most flexible, dynamic, but still auditable, business decisions.

Scoring for smarter decisions: ODM & ML integration

Introduction

Predictive analytics uses historical data to find patterns. The outcome of such analysis is often mathematical scores that are best applied when combined with an experienced person to deliver the smarter decisions.

A smart decision system encapsulates business rules and predictive models. Business rules codify the best practices and human knowledge that a business builds up over time. Predictive models recommend the best action at any given time based on these resources:

- Statistics
- Mathematical algorithms

The combination of those two disciplines enables organizations to optimize their decisions.

A solution combining technologies such as predictive analytics, and business rules can improve customer service, reduce fraud, manage risk, increase agility, and drive growth.

Solution architecture

Figure 7 shows the relationship between the key components that implement a scoring execution solution for smarter decisions.

The client application that runs in CICS or IMS on z/OS interacts with a component called “Decision Services Orchestration.” This component is responsible for orchestrating the calls to the following three services:

- Data Preparation
- Predictive (“Scoring”) Model Execution
- Business Rules Execution

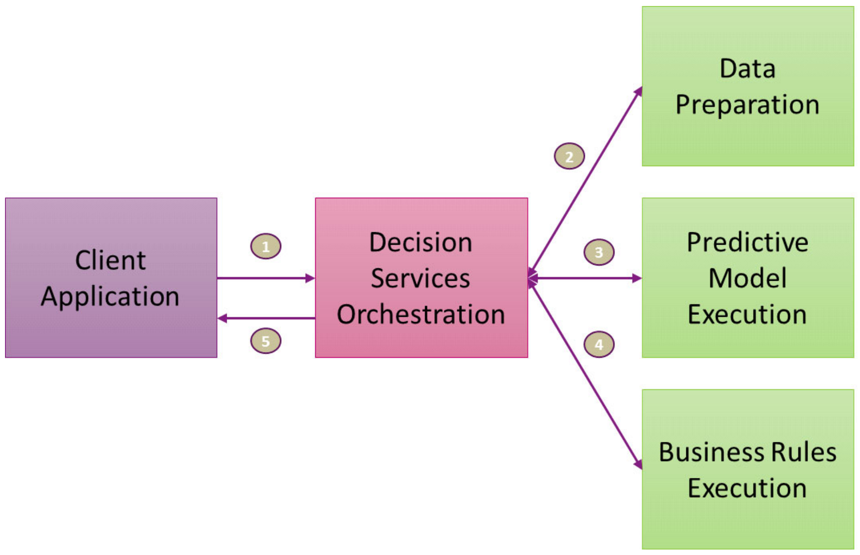


Figure 7 Key components

This “Decision Service Orchestration” component can be implemented in the CICS/IMS application or in an integration layer that runs on z/OS, such as IBM Integration Bus or WebSphere Liberty.

Data Preparation

Before the execution of the scoring model, the data might need to be prepared and formatted before sending it to the scoring execution engine.

In some scenarios, the IBM Db2® Analytics Accelerator can significantly accelerate the preparation of those data. The accelerator calculates some aggregates in real time and removes many of the burdens that are associated with the traditional batches for data aggregation.

Predictive Model Execution

Predictive model execution allows the analysis of every individual incoming transaction. This form of execution assigns each transaction a “score” based on specific factors that are indicators of the quality of the transaction. Those indicators might be calculated during the data preparation stage.

The IBM Machine Learning scoring engine for predictive model execution can be used on IBM z/OS.

Business Rules Execution

The business rules execution step refers to the core logic of how the business decision is made, taking the input of the context and the predictive behavior of the subject of interest. Basically, a business rule is a statement that describes a business policy or procedure. Business rules describe, constrain, or control some aspect of the business. IBM Operational Decision Manager can automate the decision-making process.

Simplified solution architecture

This architecture can be simplified when you allow ODM to handle the orchestration, rules, and data.

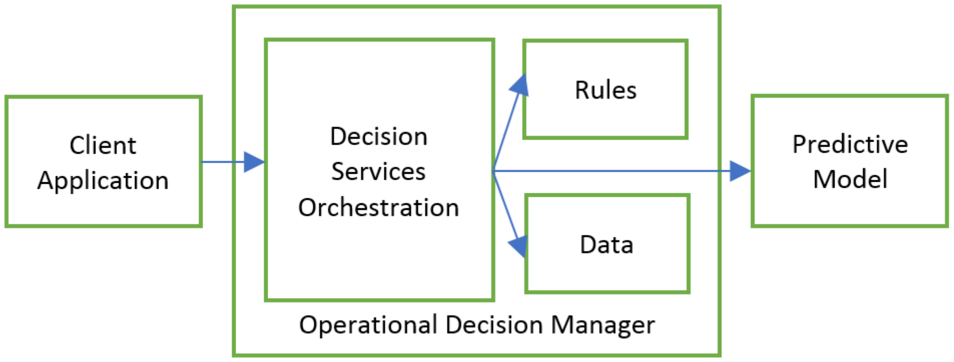


Figure 8 Simplified architecture for decision services

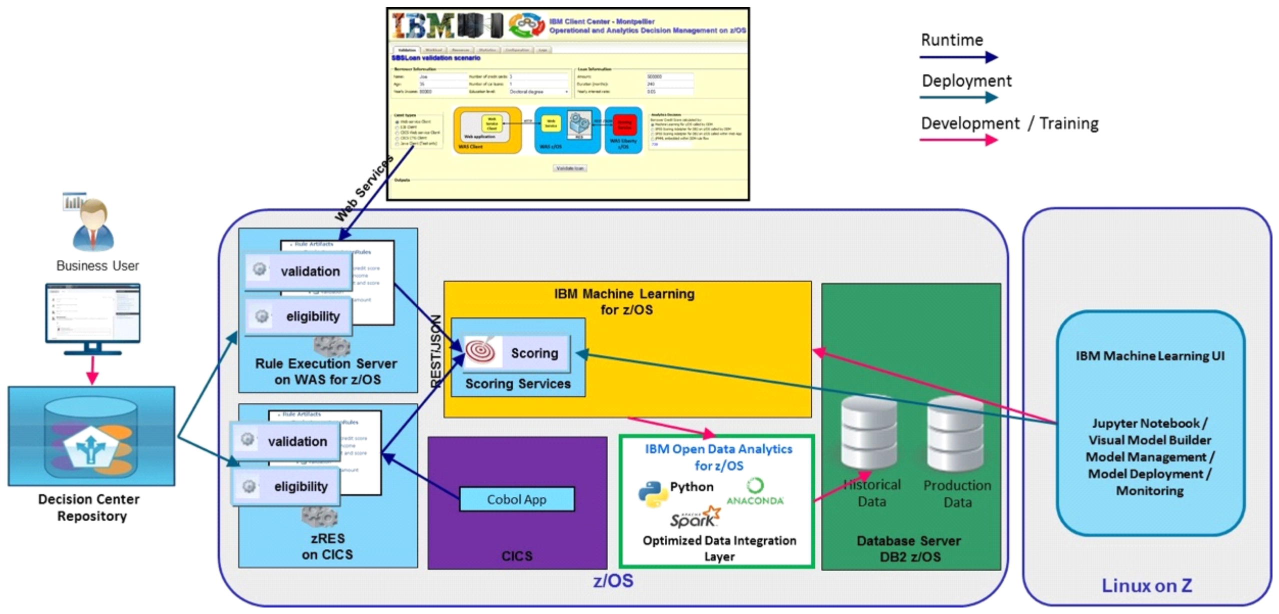
Figure 8 shows the approach that we document in the remaining pages of this Redpaper.

Integration example with ODM on z/OS calling ML on z/OS

Solution overview

Figure 9 shows an optimized infrastructure solution based on these components:

- IBM Machine Learning for the scoring engine
- ODM on z/OS for the business rules management system



Runtime
Deployment
Development / Training
IBM Client Center - Montpellier
Operational and Analytics Decision Management on z/OS
Web Services
Business User
Decision Center Repository
validation
eligibility
Rule Execution Server on WAS for z/OS
zRES on CICS
REST/JSON
IBM Machine Learning for z/OS
Scoring
Scoring Services
Cobol App
CICS
IBM Open Data Analytics for z/OS
Python
ANACONDA
Spark
Optimized Data Integration Layer
Historical Data
Production Data
Database Server DB2 z/OS
z/OS
IBM Machine Learning UI
Jupyter Notebook / Visual Model Builder Model Management / Model Deployment / Monitoring
Linux on Z

Figure 9 The Infrastructure

In this example, the decision service orchestration is implemented in ODM. ODM calls the Machine Learning scoring service running on z/OS.

Invoking IBM Machine Learning on z/OS

Environment description

Here under the list of software that is used for the integration of ML and ODM:

- Operational Decision Manager for z/OS 8.9.1. The Rule Execution Server (RES) is running on a stand-alone Liberty profile.
- IBM Machine Learning for z/OS 1.1.0.3. The Scoring Service is running on a stand-alone Liberty profile.

The RES and the Scoring Services Liberty profiles are two different instances of Liberty running in the same z/OS LPAR.

Technical Use Case

The technical Use Case for the invocation of the ML Scoring Services is the following:

```
If no valid token available then,
   ODM sends a request (with userid/pass word credential) to the Authentication
   Service running on Linux
   The authentication service verifies the credential with LDAP on z/OS server and
   returns a JWT token
End if
```

ODM sends a JSON request and attach the security token in the “Authorization” Header.

JSON Request/Reply Schema for the authentication call

The JSON Request and Reply format for the authentication is the following:

Request message format

```
{ "username": "tsoyann",
"password": "ya1217nn" }
```

Reply message format

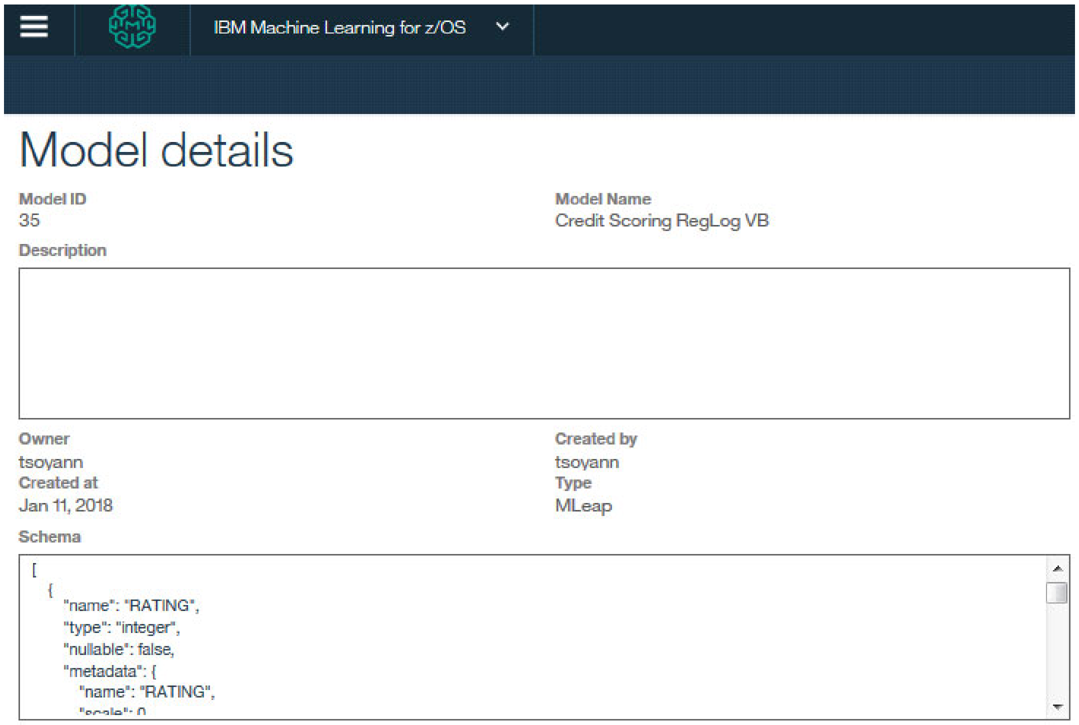
```
{
    "username": "tsoyann",
    "token":
"eyJhbGciOiJSUzI1NiIsInR5cCI6IkpXVCJ9.eyJ1c2VybmFtZSI6InRzb3lhbm4iLCJyb2xlIjoibWxh
ZGOiLCJwcml2aWxlZ2VzIjp7ImNyZWF0ZV9tb2RlbHMiOnRydWUsImRlcGxveV9tb2RlbHMiOnRydWUsIn
Njb3JlX21vZGVscyI6dHJ1ZXOsInVpZCI6IjEwMDQiLCJkaXNwbGF5X25hbWUiOiJZYW5uIEtpbmRlbGJl
cmdlciIsImlhdCI6MTUxNzM5NDkxNCwiZXhwIjoxNTE3NDQxNzE0fQ.Usr8b-U9en9WSZjzljcwQsF_a26
8UA0Of72iW-BCzwrWt1KBJuAgGW5b-pt1cbzqjTD8dSHkCYoMSsif8BQ_i2OzOFXK29UU-cf8yHgApqDSM
1aC_poqJrDLxdBsxnQ-EUItpwrwVQvOpZ_-XgNQL5y5TxG2YF6oQdw-fihWKwcfp3jQ9kdigYlIsSYrlZS
urkbjbObwOq-PMtNe61d7cOVhOmTXPR8BxssLUWLcTIHn5M8DCXNchC7eQ4btZAflWBuKQo_cAkPEH2BDw
RqKze92BKOOTHl6wCMaqxL8Z4PRRrmFv3cCcThvMY2EJKogjT52R9wvHyAs4Um49m7pwQ",
    "_messageCode_": "success",
    "message": "success"
}
```

JSON Request/Reply Schema for the ML call

The JSON Request and Reply format depends on the model that is built within Machine Learning.

Request schema format

You get the JSON Request format from the Machine Learning user interface. Go to the 'Model Management' part of the product, select the 'Models' tab, and finally the model that was previously saved. Figure 10 shows a screen capture of the detail of the model that is used in this test.



Model details

Model ID
35

Model Name
Credit Scoring RegLog VB

Description

Owner
tsoyann

Created by
tsoyann

Created at
Jan 11, 2018

Type
MLeap

Schema

```
[
  {
    "name": "RATING",
    "type": "integer",
    "nullable": false,
    "metadata": {
      "name": "RATING",
```

Figure 10 Model panel showing Model deployment ID and schema

Example 1 shows the full JSON request schema of the selected model.

Example 1

```
[
    {
        "name": "RATING",
        "type": "integer",
        "nullable": false,
        "metadata": {
            "name": "RATING",
            "scale": 0
        }
    },
    {
        "name": "AGE",
        "type": "integer",
        "nullable": false,
        "metadata": {
            "name": "AGE",
            "scale": 0
        }
    },
    {
        "name": "INCOME",
        "type": "integer",
        "nullable": false,
        "metadata": {
            "name": "INCOME",
            "scale": 0
        }
    },
    {
        "name": "CARDS",
        "type": "integer",
        "nullable": false,
        "metadata": {
            "name": "CARDS",
            "scale": 0
        }
    },
    {
        "name": "EDUCATION",
        "type": "string",
        "nullable": false,
        "metadata": {
            "name": "EDUCATION",
            "scale": 0
        }
    },
    {
        "name": "CARLOANS",
        "type": "integer",
        "nullable": false,
        "metadata": {
            "name": "CARLOANS",
            "scale": 0
        }
    },
    {
        "name": "CREDITREIMBURSED",
        "type": "integer",
        "nullable": false,
        "metadata": {
            "name": "CREDITREIMBURSED",
            "scale": 0
        }
    }
]
```

Reply message format

Example 2 shows the reply message format when you invoke the scoring service:

Example 2

```
[
    {
        "EDUCATION": "Elementary school",
        "CREDITREIMBURSED": 0,
        "features": [
            0,
            2.7479620351039955,
            0.14532499380084998,
            2.190682544490108,
            4.103721416104103,
            1.8466279100613288
        ],
        "RATING": 0,
        "CARDS": 3,
        "nodeADP_class": 1,
        "prediction": 1,
        "AGE": 23,
        "INCOME": 20000,
        "rawPrediction": [
            -1.187588229560474,
            1.187588229560474
        ],
        "CARLOANS": 2,
        "nodeADP_classes": [
            0,
            1
        ],
        "probability": [
            0.23369055645660813,
            0.7663094435433918
        ]
    }
]
```

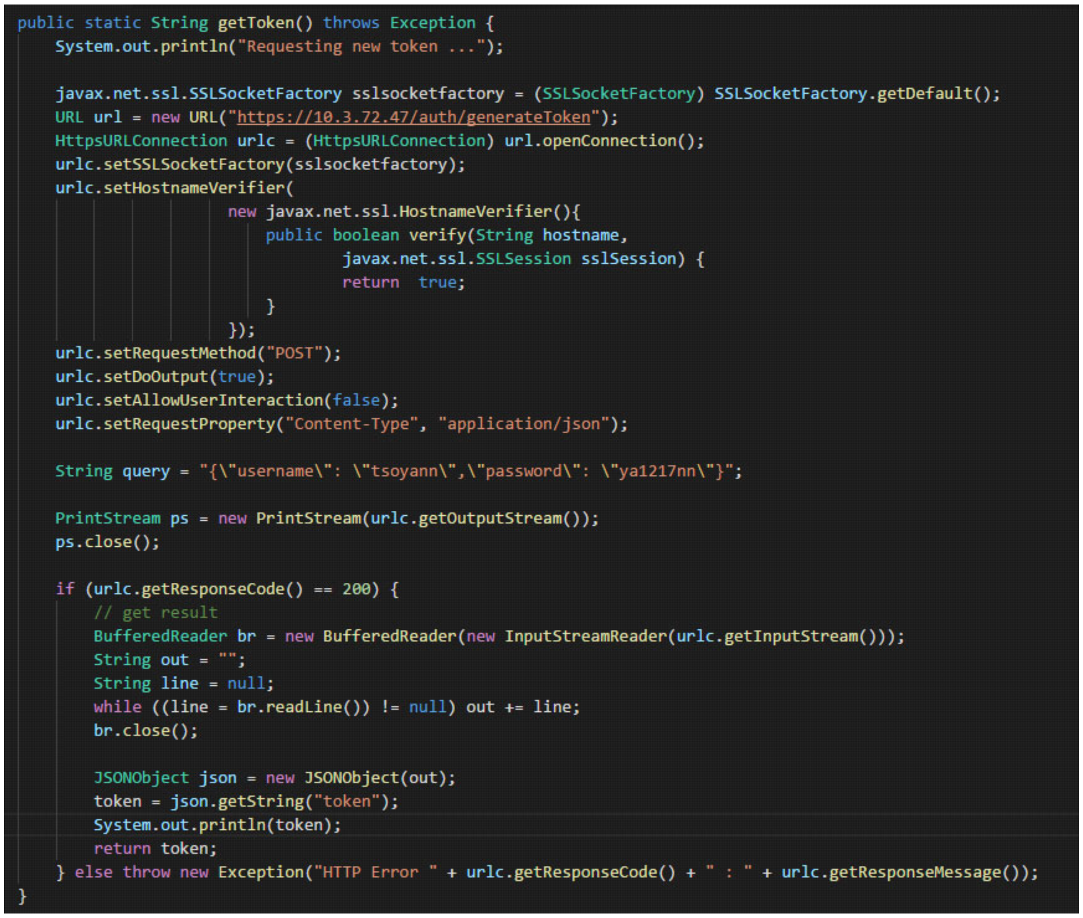
Java implementation within ODM

A Java class has been created in the XOM. This Java code performs both the call to the authentication service and the call to the scoring service.

Java code calling the authentication service

Example 11 shows the Java code to perform the authentication. The following steps are required:

1. Set a SSL connection with the authentication service
2. Create the JSON request message. In this example, the user/pwd are hardcoded in the source code.
3. Call the authentication service.
4. Process the reply from the authentication service.



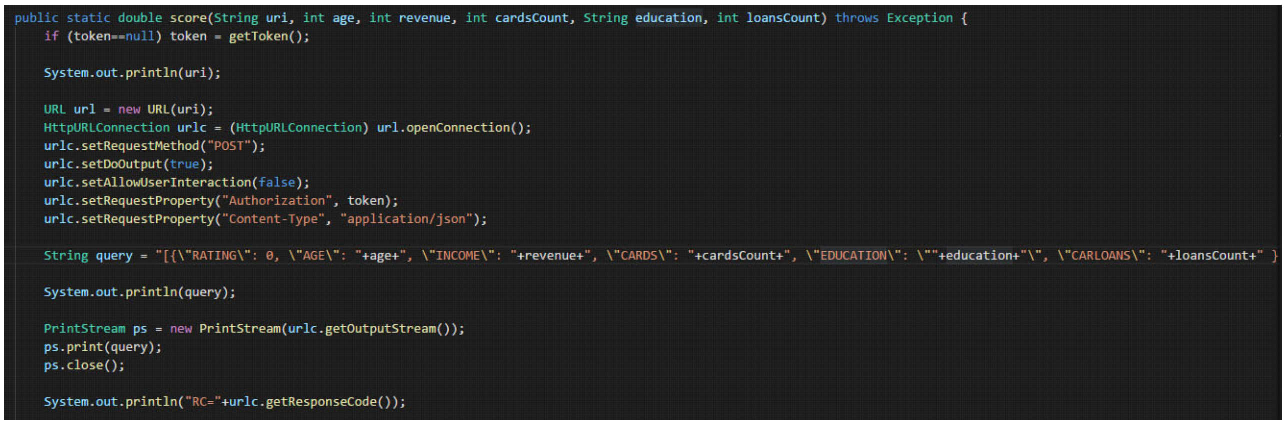
```
public static String getToken() throws Exception {
    System.out.println("Requesting new token ...");

    javax.net.ssl.SSLSocketFactory sslsocketfactory = (SSLSocketFactory) SSLSocketFactory.getDefault();
    URL url = new URL("https://10.3.72.47/auth/generateToken");
    HttpsURLConnection urlc = (HttpsURLConnection) url.openConnection();
    urlc.setSSLSocketFactory(sslsocketfactory);
    urlc.setHostnameVerifier(
                         new javax.net.ssl.HostnameVerifier(){
                             public boolean verify(String hostname,
                                     javax.net.ssl.SSLSession sslSession) {
                                     return  true;
                             }
                         });
    urlc.setRequestMethod("POST");
    urlc.setDoOutput(true);
    urlc.setAllowUserInteraction(false);
    urlc.setRequestProperty("Content-Type", "application/json");

    String query = "{\"username\": \"tsoyann\",\"password\": \"ya1217nn\"}";

    PrintStream ps = new PrintStream(urlc.getOutputStream());
    ps.close();

    if (urlc.getResponseCode() == 200) {
        // get result
        BufferedReader br = new BufferedReader(new InputStreamReader(urlc.getInputStream()));
        String out = "";
        String line = null;
        while ((line = br.readLine()) != null) out += line;
        br.close();

        JSONObject json = new JSONObject(out);
        token = json.getString("token");
        System.out.println(token);
        return token;
    } else throw new Exception("HTTP Error " + urlc.getResponseCode() + " : " + urlc.getResponseMessage());
}
```

Figure 11 Java code calling the authentication service

Java code calling the scoring service

The following screen capture shows the Java code to call the scoring service. The following steps are implemented:

1. Set the HTTP request.
2. Create the JSON request message.
3. Call the scoring service.



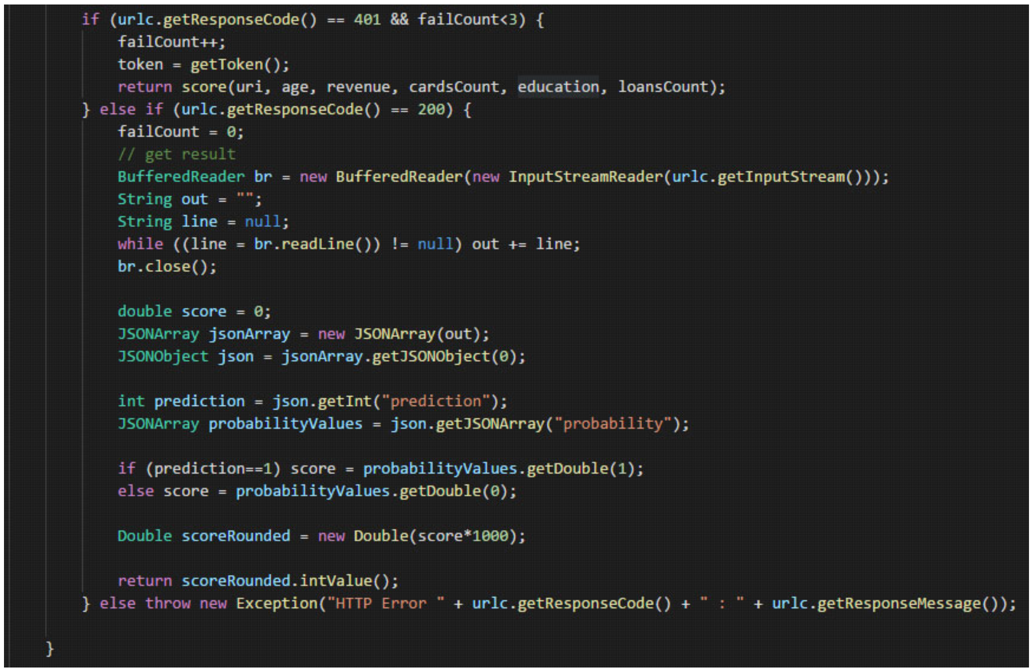
```
public static double score(String uri, int age, int revenue, int cardsCount, String education, int loansCount) throws Exception {
    if (token==null) token = getToken();

    System.out.println(uri);

    URL url = new URL(uri);
    HttpURLConnection urlc = (HttpURLConnection) url.openConnection();
    urlc.setRequestMethod("POST");
    urlc.setDoOutput(true);
    urlc.setAllowUserInteraction(false);
    urlc.setRequestProperty("Authorization", token);
    urlc.setRequestProperty("Content-Type", "application/json");

    String query = "[{\"RATING\": 0, \"AGE\": "+age+", \"INCOME\": "+revenue+", \"CARDS\": "+cardsCount+", \"EDUCATION\": \""+education+"\", \"CARLOANS\": "+loansCount+" }

    System.out.println(query);

    PrintStream ps = new PrintStream(urlc.getOutputStream());
    ps.print(query);
    ps.close();

    System.out.println("RC="+urlc.getResponseCode());
```

Figure 12 Java code calling the scoring service

Java code to process the response of the scoring service

The following screen capture shows the Java code to process the response of the scoring service. The following steps are implemented:

1. Check the response code from the HTTP request.
2. If there is an authentication problem, then
   - Request a new token
3. Extract the prediction and the probability value from the JSON reply message.
4. Return the score, which is the probability multiplied by 1000.



```
    if (urlc.getResponseCode() == 401 && failCount<3) {
        failCount++;
        token = getToken();
        return score(uri, age, revenue, cardsCount, education, loansCount);
    } else if (urlc.getResponseCode() == 200) {
        failCount = 0;
        // get result
        BufferedReader br = new BufferedReader(new InputStreamReader(urlc.getInputStream()));
        String out = "";
        String line = null;
        while ((line = br.readLine()) != null) out += line;
        br.close();

        double score = 0;
        JSONArray jsonArray = new JSONArray(out);
        JSONObject json = jsonArray.getJSONObject(0);

        int prediction = json.getInt("prediction");
        JSONArray probabilityValues = json.getJSONArray("probability");

        if (prediction==1) score = probabilityValues.getDouble(1);
        else score = probabilityValues.getDouble(0);

        Double scoreRounded = new Double(score*1000);

        return scoreRounded.intValue();
    } else throw new Exception("HTTP Error " + urlc.getResponseCode() + " : " + urlc.getResponseMessage());

}
```

Figure 13   Java code to process the response of the scoring service

ODM and Scoring Decisions

This section describes the steps that are needed to make a call to the Machine Learning scoring engine with IBM Operational Decision Manager for Z.

Enable a rule to make a call to the scoring engine

What is the interface to the scoring model

The scoring model that ODM will use has been trained with historical data and deployed to the scoring engine. See MLz documentation on how to do this.

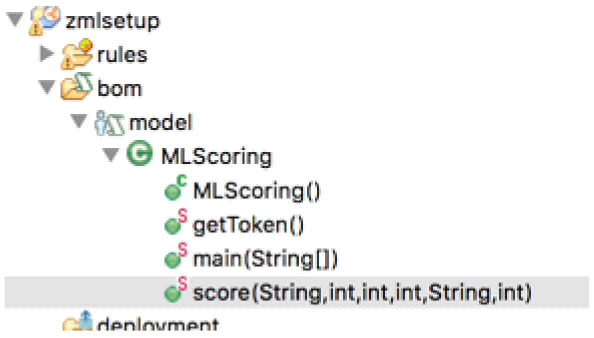
This model will have specific input values that the scoring model will be called with to return a score.

The scoring interface is described by a schema. The interface will be described in Java as a method call.

The Project

The rule project which makes the RESTful call to the rule engine might look like the following example:

- **`score`** takes six input parameters to return a score.
- **`getToken`** the get token is used to obtain credentials from the scoring engine to make the scoring request.

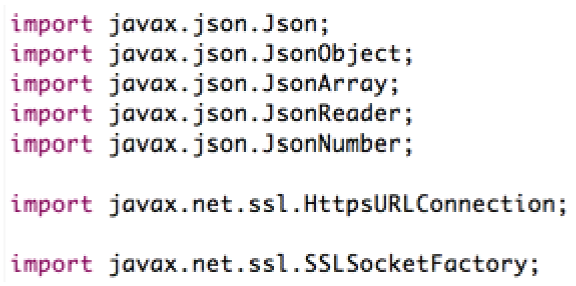


```
▼ zmlsetup
  ▶ rules
  ▼ bom
    ▼ model
      ▼ MLScoring
          MLScoring()
          getToken()
          main(String[])
          score(String,int,int,int,String,int)
    deployment
```

Figure 14   Example showing a rule project with a Business Object model for the 'score'

Some observations about the Java code to call the score

In the previous section, the java project implementation could have used the javax package to manage the JSON formats, based on the following imports:



```java
import javax.json.Json;
import javax.json.JsonObject;
import javax.json.JsonArray;
import javax.json.JsonReader;
import javax.json.JsonNumber;

import javax.net.ssl.HttpsURLConnection;

import javax.net.ssl.SSLSocketFactory;
```

Figure 15   javax imports

The code that is used is supplied, as is, as a download with this paper.

Authorization and Certificates

To make a score that uses the mechanism, you must get a token. **`getToken()`** is the code that specifies where the keystore is on the system. The keystore must contain the certificate, which is supplied from the backend system from which you are requesting a token. See MLz documentation for further information.

In our test environment, we override the **Verify** of the hostname. We are using the IP addresses as the CN and not a host name. See **Subject Alternative Name** in the certificate for further information.

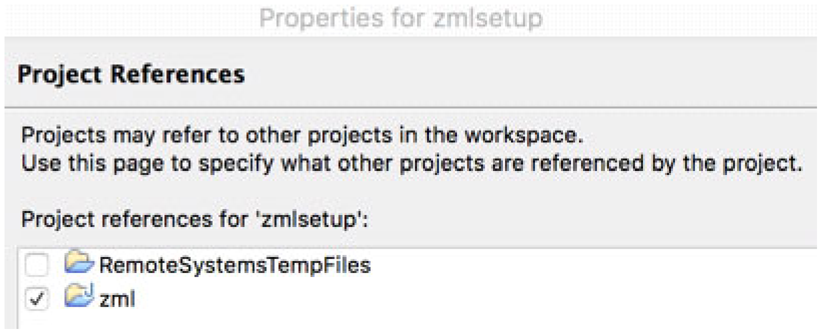
The scoring model

In this paper, the model that has been deployed to the ML engine on z/OS has the following ID: `http://winmvsga.hursley.ibm.com:11224/iml/v2/scoring/online/6`

You need to deploy a model in your MLz setup to obtain the ID.

Import the java code into ODM

To have a project like the preceding example, make an Eclipse project for the Java code. For the Java code, see Example 4 in “Appendix B. Sample calls” on page 31. Then in your rule project, associate the Rule Project References to this Java Project.



Properties for zmlsetup

Project References

Projects may refer to other projects in the workspace.
Use this page to specify what other projects are referenced by the project.

Project references for 'zmlsetup':

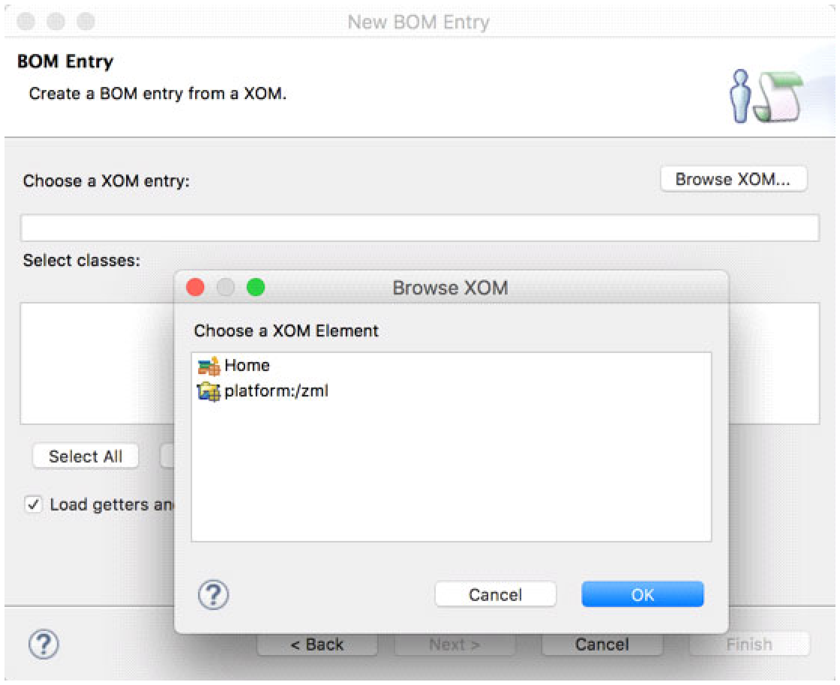
- [ ] RemoteSystemsTempFiles
- [✓] zml

Figure 16 Import the java code into ODM

In the preceding example, the java project **`zml`** is going to be referenced by this rule project.

The java project is added to the project references.

Next, you create a new BOM entry / import the XOM to this project.



New BOM Entry

BOM Entry

Create a BOM entry from a XOM.

Choose a XOM entry:

Select classes:

Load getters an

Browse XOM

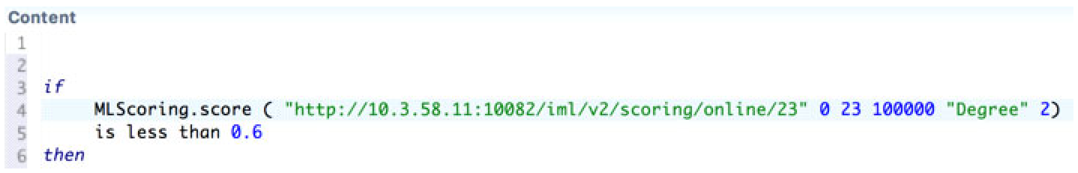
Choose a XOM Element

- Home
- platform:/zml

Figure 17 Creating a BOM entry

The preceding example shows selection of the XOM from the Java object to make a new BOM/XOM entry.

Now that the XOM code has been imported, Rules can now be written to make a score with the input values.



```
Content
if
     MLScoring.score ( "http://10.3.58.11:10082/iml/v2/scoring/online/23" 0 23 100000 "Degree" 2)
     is less than 0.6
then
```

Figure 18   Changing visualization for the score

In the preceding example, we changed the verbalization for the score to feature spacing between the input variables.

Summary

The goal of this paper was to introduce two products:

- Machine Learning on Z
- Operational Decision Manager on Z

On its own, the Machine Learning approach is good at perceiving and learning from historic data. Machine Learning is not so good at abstracting and reasoning when there is no contextual capability. It is good at identifying patterns in historic data.

On the other hand, decisions based on rules are good at reasoning in defined problems, but have no learning capability and handle uncertainty poorly.

ML use cases

Healthcare

Machine learning is branching out across different fields. One of the most interesting fields is healthcare, for example, building predictive scoring models to predict heart rate failure.

A challenge of using machine learning in healthcare is a "black box problem." With this problem, the machine learning algorithm can't explain how it arrived at a diagnosis, even when it is correct.

Finance

Machine learning is transforming all areas of business, including how financial institutions and other industries are approaching tighter compliance requirements and risk management. Deploying a financial risk model to determine customer credit worthiness and incorporating the data into business applications.

This risk assessment and management takes place securely in place and in real time. As a result, financial institutions more accurately determine credit worthiness and achieve other business needs.

Financial institutions need to continually weigh the risks of their transactions, and they determine their risk level through credit scoring.

Business Rules use cases

Repeatable business decisions can be captured and automated to improve your business responsiveness, minimize compliance risk and streamline workflow processes. A business rules management system adds agility and real-time decision making to your day-to-day operations. At the same time, such a system enables quick optimization and change management capabilities to keep up with the pace of your business.

Once you separate business rules from applications, you get business agility.

Empower business analysts and policy managers to develop and maintain the decision logic themselves, without having to depend on IT, with an easy-to-use, and with a low-code interface that facilitates collaboration.

Business Rules management systems are used extensively in the finance, healthcare, and insurance industries, where the business is governed by legislative and changing rules.

Combining Technologies

The combination of increased requirements and the development of advanced new technologies brings a new era: scoring based on machine learning and business rules systems.

Improved accuracy and speed with systems that can constantly analyze feeds of data and events, reveals the risks and opportunities, and lets you make real-time decisions.

By combining these two disciplines, you get the best of both a deterministic and a probabilistic approach. As a result, you can make the best decisions given the data at any given point in time.

Real-time authorization and decisioning that applies thousands of data points and sophisticated modeling techniques to help increase the accuracy of approvals of genuine transactions. Credit card companies have deployed this technology across global networks to counteract fraudulent transactions. Their systems work by assigning a score to every transaction, which is then used to help judge future payments.

The paper outlines how the two products work together to make more accurate decisions:

- The decisions can be governed by scoring.
- The input data can be scored against a model.
- The model predictions make better decisions based on the rules, in a specific time frame.
- You can make a REST call from Java to make a ‘Score’ from the Machine Learning product on Z.
- The Java that you use to make the call can be imported into Operational Decision Manager on z/OS and used by a rule to generate a score.

As shown in this document, it is relatively straightforward to make a REST call to obtain a score. Use of Machine Learning scores in your decisions generates much better decisions.

Business Rules enable decisions to be made based on your machine-learning scores. Machine Learning models enable more accurate and better decisions to be made based on predictive analytics.

Appendix A. The REST client

This appendix illustrates technical integration between a RESTClient and IBM Machine Learning for z/OS scoring engine. As of today, only the previous implementation is coded in the Java code that we shared. It differs from URL formats and JSON message formats. They are not compatible.

Environment	► IBM Machine Learning for z/OS 1.1.0.3 ► Scoring engine (z/OS): 10.3.58.11 ► Authentication Services (Linux): 10.3.72.47 ► Client: Firefox RESTClient plug-in **Note:** As of today, the Credit Scoring demo is set up to call a IBM Machine Learning for z/OS 1.1.0.3 scoring engine with a different JSON format than 1.1.0.3 and later. (The old format seems not to be supported any more).
Technical Use Case	► RESTClient sends a request (with user ID/password credential) to the Authentication Service. ► The Authentication Service verifies credentials with the LDAP on z/OS server and returns a JWT token. ► RESTClient sends JSON request and attaches the security token in the “Authorization” Header. **Note:** The security token is set with an expiration delay.

JSON Request Schema

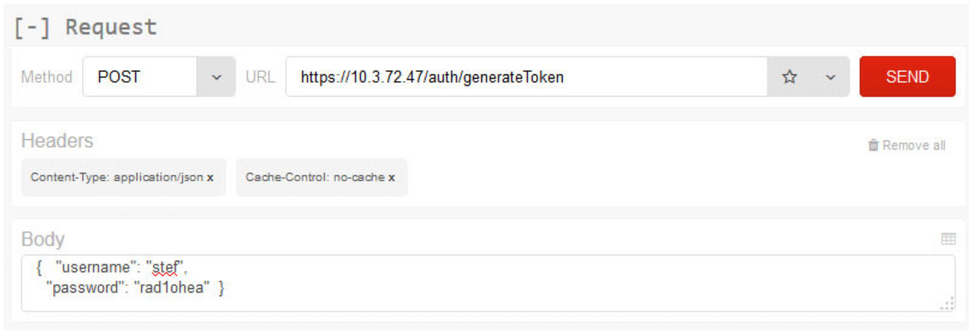
Example 3 shows a JSON Request Schema:

Example 3

```
[
    {
        "name": "RATING",
        "type": "integer",
        "nullable": false,
        "metadata": {
            "name": "RATING",
            "scale": 0
        }
    },
    {
        "name": "AGE",
        "type": "integer",
        "nullable": false,
        "metadata": {
            "name": "AGE",
            "scale": 0
        }
    },
    {
        "name": "INCOME",
        "type": "integer",
        "nullable": false,
        "metadata": {
            "name": "INCOME",
            "scale": 0
        }
    },
    {
        "name": "CARDS",
        "type": "integer",
        "nullable": false,
        "metadata": {
            "name": "CARDS",
            "scale": 0
        }
    },
    {
        "name": "EDUCATION",
        "type": "string",
        "nullable": false,
        "metadata": {
            "name": "EDUCATION",
            "scale": 0
        }
    },
    {
        "name": "CARLOANS",
        "type": "integer",
        "nullable": false,
        "metadata": {
            "name": "CARLOANS",
            "scale": 0
        }
    }
]
```

Authentication Call

Request


[-] Request

Method: POST URL: https://10.3.72.47/auth/generateToken SEND

Headers: Content-Type: application/json x Cache-Control: no-cache x Remove all

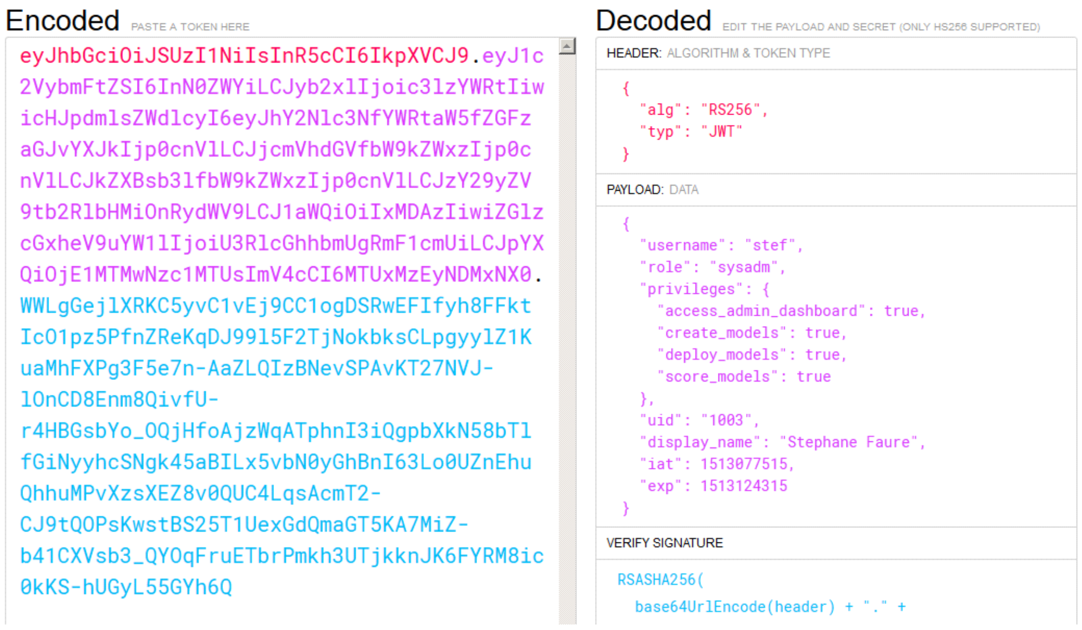
Body:
```
{ "username": "stef",
 "password": "rad1ohea" }
```

Figure 19 Request for an authentication call

Response

```
{"username":"stef",
"token":"xxxxx",
"_messageCode_":"success",
"message":"success"}
```

The token can be inspected with the JWT parser at jwt.io:


Encoded
PASTE A TOKEN HERE
Decoded
EDIT THE PAYLOAD AND SECRET (ONLY HS256 SUPPORTED)
HEADER: ALGORITHM & TOKEN TYPE
PAYLOAD: DATA
VERIFY SIGNATURE

```
eyJhbGciOiJSUzI1NiIsInR5cCI6IkpXVCJ9.eyJ1c
2VybmFtZSI6InN0ZWYiLCJyb2xlIjoic3lzYWRtIiw
icHJpdmlsZWdlcyI6eyJhY2Nlc3NfYWRtaW5fZGFz
aGJvYXJkIjp0cnVlLCJjcmVhdGVfbW9kZWxzIjp0c
nVlLCJkZXBsb3lfbW9kZWxzIjp0cnVlLCJzY29yZV
9tb2RlbHMiOnRydWV9LCJ1aWQiOiIxMDAzIiwiZGlz
cGxheV9uYW1lIjoiU3RlcGhhbmUgRmF1cmUiLCJpYX
QiOjE1MTMwNzc1MTUsImV4cCI6MTUxMzEyNDMxNX0.
WWLgGejlXRKC5yvC1vEj9CC1ogDSRwEFIfyh8FFkt
Ic01pz5PfnZReKqDJ99l5F2TjNokbksCLpgyylZ1K
uaMhFXPg3F5e7n-AaZLQIzBNevSPAvKT27NVJ-
lOnCD8Enm8QivfU-
r4HBGsbYo_OQjHfoAjzWqATphnI3iQgpbXkN58bTl
fGiNyyhcSNgk45aBILx5vbN0yGhBnI63Lo0UZnEhu
QhhuMPvXzsXEZ8v0QUC4LqsAcmT2-
CJ9tQOPsKwstBS25T1UexGdQmaGT5KA7MiZ-
b41CXVsb3_QYOqFruETbrPmkh3UTjkknJK6FYRM8ic
0kKS-hUGyL55GYh6Q
```

```
{
  "alg": "RS256",
  "typ": "JWT"
}
```

```
{
  "username": "stef",
  "role": "sysadm",
  "privileges": {
    "access_admin_dashboard": true,
    "create_models": true,
    "deploy_models": true,
    "score_models": true
  },
  "uid": "1003",
  "display_name": "Stephane Faure",
  "iat": 1513077515,
  "exp": 1513124315
}
```

```
RSASHA256(
  base64UrlEncode(header) + "." +
```

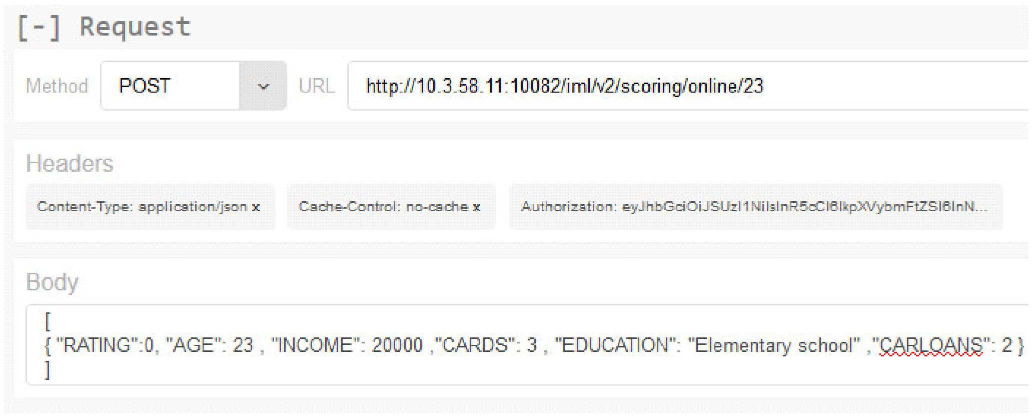
Figure 20 Response for an authentication call

Call to the scoring engine with the JWT token:

Note: In this example, 23 is the deployment ID for the model I want to call.

JWT token value is now in the “Authorization” header.

Request



POST http://10.3.58.11:10082/iml/v2/scoring/online/23

Content-Type: application/json x
Cache-Control: no-cache x
Authorization: eyJhbGciOiJSUzI1NiIsInR5cCI6IkpXVybmFtZSI6InN...

```
[
{ "RATING":0, "AGE": 23 , "INCOME": 20000 ,"CARDS": 3 , "EDUCATION": "Elementary school" ,"CARLOANS": 2 }
]
```

Figure 21 Request for a call to the scoring engine

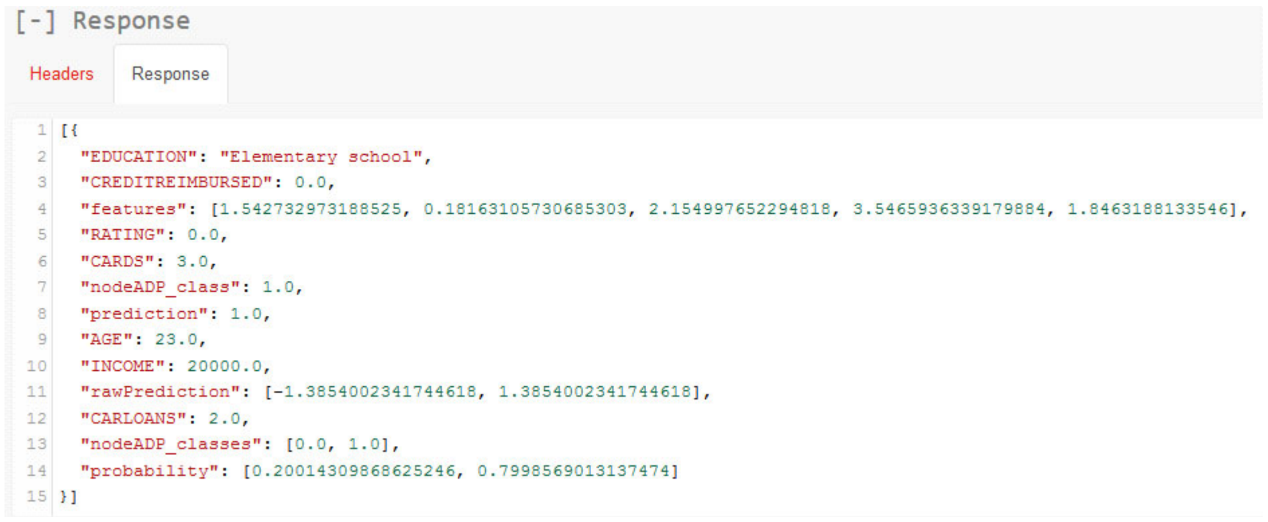
Message:

```
[
{"RATING": 0, "AGE": 23 , "INCOME": 20000 ,"CARDS": 3 , "EDUCATION": "Elementary
school" ,"CARLOANS": 2 }
]
```

Alternate format:

```
{
  "fields": ["RATING", "AGE" ,"INCOME" ,"CARDS" , "EDUCATION" ,"CARLOANS"],
  "values": [[ 0,  23 ,  20000 , 3 , "Elementary school" ,3]]
}
```

Response:



```
[{
  "EDUCATION": "Elementary school",
  "CREDITREIMBURSED": 0.0,
  "features": [1.542732973188525, 0.18163105730685303, 2.154997652294818, 3.5465936339179884, 1.8463188133546],
  "RATING": 0.0,
  "CARDS": 3.0,
  "nodeADP_class": 1.0,
  "prediction": 1.0,
  "AGE": 23.0,
  "INCOME": 20000.0,
  "rawPrediction": [-1.3854002341744618, 1.3854002341744618],
  "CARLOANS": 2.0,
  "nodeADP_classes": [0.0, 1.0],
  "probability": [0.20014309868625246, 0.7998569013137474]
}]
```

Figure 22   Response for a call to the scoring engine

Raw response:

```
[{
  "EDUCATION": "Elementary school",
  "CREDITREIMBURSED": 0.0,
  "features": [1.542732973188525, 0.18163105730685303, 2.154997652294818, 3.5465936339179884,
1.8463188133546],
  "RATING": 0.0,
  "CARDS": 3.0,
  "nodeADP_class": 1.0,
  "prediction": 1.0,
  "AGE": 23.0,
  "INCOME": 20000.0,
  "rawPrediction": [-1.3854002341744618, 1.3854002341744618],
  "CARLOANS": 2.0,
  "nodeADP_classes": [0.0, 1.0],
  "probability": [0.20014309868625246, 0.7998569013137474]
}]
```

What we extract is the second value in probability (probability that a credit will be reimbursed), but typically a generic interface returns the prediction and its probability.

Official Documentation

`https://www.ibm.com/support/knowledgecenter/en/SS9PF4_1.1.0.3/src/tpc/mlz_scoringapi.html`

Appendix B. Sample calls

This appendix is a collection of samples to facilitate making a call to zML to make a score prediction.

Sample Java code

To build request to invoke scoring model.

MLScoring

Example 4 shows an example of ML Scoring:

Example 4

```
import java.io.BufferedReader;
import java.io.InputStreamReader;
import java.io.PrintStream;
import java.net.HttpURLConnection;
import java.net.URL;


import javax.json.Json;
import javax.json.JsonObject;
import javax.json.JsonArray;
import javax.json.JsonReader;
import javax.json.JsonNumber;

import javax.net.ssl.HttpsURLConnection;

import javax.net.ssl.SSLSocketFactory;

public class MLscoring1 {

   //private static String token = 
"eyJhbGciOiJSUzI1NiIsInR5cCI6IkpXVCJ9.eyJ1c2VybmFtZSI6InR1c2VyMDEiLCJyb2xlIjoic3lz
YWRtIiwicHJpdmlsZWdlcyI6eyJhY2Nlc3NfYWRtaW5fZGFzaGJvYXJkIjp0cnVlLCJjcmVhdGVfbW9kZW
xzIjp0cnVlLCJkZXBsb3lfbW9kZWxzIjp0cnVlLCJzY29yZV9tb2RlbHMiOnRydWV9LCJ1aWQiOiIxMDAx
IiwiZGlzcGxheV9uYW1lIjoidHVzZXIwMSIsImlhdCI6MTUyMTgxMjgxNiwiZXhwIjoxNTIxODU5NjE2fQ
.d3erNTN_PIBIey5cOarR-ye1DhNWWjxcizf4JQY7duKJFsIwXMeFOHi7MGC2pBl1huFC4jAXsC9iop4Si
AM72caKfNh9YECmzQl2jyx_1Z8KexORLmAliASjRKEiZUOUsUzbV7d45HOxM17LBOZk7aAGhdNfHoweUqq
LoPAEPti-Pu4WmEZbzMuFX3nt87LQYKvOLMtC2PdJFoeNGlGskR5MtdBfOWGSkYZMo1t_yUOPqSmH1aj86
MkQC4bwvl3Nxj2FqiPO6OxOH_GOFgO7J7rLBZEK_p6Ab7pgR3cFMF2oo9LR_WluCr--BQuYL54KoXK3iKu
G3M5jNEORui2mzA";

   private static String token = null;
   private static int failCount = 0;

   /**
    @param args
    @throws Exception
    */
   public static void main(String[] args) throws Exception {
```

```
System.out.println(score("http://mvs.hursley.ibm.com:11224/iml/v2/scoring/online/6
", 1, 51, "F", 8, 28326, 1, "CA"));
   }

   public static double score(String uri, int education, int age, String sex, int 
negtweets, int income, int activity, String state) throws Exception {
      if (token==null) token = getToken();

      System.out.println(uri);

      URL url = new URL(uri);
      HttpURLConnection urlc = (HttpURLConnection) url.openConnection();
      urlc.setRequestMethod("POST");
      urlc.setDoOutput(true);
      urlc.setAllowUserInteraction(false);
      urlc.setRequestProperty("Authorization", token);
      urlc.setRequestProperty("Content-Type", "application/json");

      String query = "{\"fields\":[\"EDUCATION\", \"AGE\", \"SEX\",\"NEGTWEETS\", 
\"INCOME\", \"ACTIVITY\", \"STATE\"],\"values\":[[" + education + "," + age + 
",\"" + sex + "\","+ negtweets + "," + income + "," + activity + ",\"" + state + 
"\"]]}";
      System.out.println(query);

      PrintStream ps = new PrintStream(urlc.getOutputStream());
      ps.print(query);
      ps.close();

      System.out.println("RC="+urlc.getResponseCode());

      if (urlc.getResponseCode() == 401 && failCount<3) {
         failCount++;
         //token = getToken();
         return score(uri, education, age, sex, negtweets, income, activity, 
state);

      } else if (urlc.getResponseCode() == 200) {
         failCount = 0;

         // get result
         BufferedReader br = new BufferedReader(new 
InputStreamReader(urlc.getInputStream()));

JsonReader reader = Json.createReader(br);
            JsonObject obj = (JsonObject)reader.read();
            JsonArray arr = obj.getJsonArray("values");
            System.out.println("array = " + arr);
            arr = arr.getJsonArray(0);
            System.out.println("array = " + arr);
            JsonNumber num = arr.getJsonNumber(0);
            double score = num.doubleValue();
            System.out.println("score = " + score);
         return num.intValue();
      } else throw new Exception("HTTP Error " + urlc.getResponseCode() + " : " +
urlc.getResponseMessage());

   }

   public static String getToken() throws Exception {
      System.out.println("Requesting new token ...");

      System.setProperty ("javax.net.ssl.trustStore",
"/Users/mikejohn/Documents/Machine Learning /certificate/CDL/keystore.jks");
        System.setProperty ("javax.net.ssl.trustStorePassword", "password");

      javax.net.ssl.SSLSocketFactory sslsocketfactory = (SSLSocketFactory)
SSLSocketFactory.getDefault();

      URL url = new URL("https://x.xx.xxx.xxx/auth/generateToken");
      HttpsURLConnection urlc = (HttpsURLConnection) url.openConnection();
      urlc.setSSLSocketFactory(sslsocketfactory);
      urlc.setHostnameVerifier(
                  new javax.net.ssl.HostnameVerifier(){
                      public boolean verify(String hostname,
                          javax.net.ssl.SSLSession sslSession)  {
                          return true;
                      }
                  });

      urlc.setRequestMethod("POST");
      urlc.setDoOutput(true);
      urlc.setAllowUserInteraction(false);
      urlc.setRequestProperty("Content-Type", "application/json");

      String query =   "{\"username\": \"tuser01\",\"password\": \"xxxxxxx\"}";


      PrintStream ps = new PrintStream(urlc.getOutputStream());
      ps.print(query);
      ps.close();

      if (urlc.getResponseCode() == 200) {
         // get result
         BufferedReader br = new BufferedReader(new
InputStreamReader(urlc.getInputStream()));

         JsonReader reader = Json.createReader(br);
         JsonObject obj = (JsonObject)reader.read();
         String token = obj.getJsonString("token").getString();

         System.out.println(token);
         return token;
      } else throw new Exception("HTTP Error " + urlc.getResponseCode() + " : " +
urlc.getResponseMessage());
   }
}
```

Appendix C. Additional materials

This paper refers to additional material that can be downloaded from the internet as described in the following sections.

Locating the web material

The web material that is associated with this paper is available in softcopy on the internet from the IBM Redbooks web server:

`ftp://www.redbooks.ibm.com/redbooks/REDP5502`

Alternatively, you can go to the IBM Redbooks website:

`ibm.com``/redbooks`

Search for REDP5502, select the title, and then click **Additional materials** to open the directory that corresponds with the IBM Redpaper form number, REDP5502.

Using the web material

The additional web material that accompanies this paper includes the following files:

File name	*Description*
Test-model-hao0208.tar.gz	Sample code that is referenced in Appendix B.
MLscoring2.java	Sample code for class MLscoring2.

Downloading and extracting the web material

Create a subdirectory (folder) on your workstation, and extract the contents of the web material .zip file into this folder.

Authors

This paper was produced by a team of specialists from around the world.

Mike Johnson
David Griffiths
Chris Backhouse
IBM Hursley Park ODM Development Team

Hao Zhang
Ke Wei Wei
IBM China MLz Development Team

Stéphane Faure
Yann Kindelberger
IBM Client Center Montpellier

Now you can become a published author, too!

Stay connected to IBM Redbooks

Notices

Trademarks

REDP-5502-00

ISBN 0738456926

Printed in U.S.A.