**Lab Guide**

IBM Business Automation Manager Open Edition

Nigel Crowther – ncrowther@uk.ibm.com

Hands-on Lab

Building watsonx Orchestrate Skills with BAMOE and Cloudant

V1.0

07-Dec-2023



NOTICES

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

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

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

IBM Director of Licensing

IBM Corporation

North Castle Drive, MD-NC119

Armonk, NY 10504-1785

United States of America

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.

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.

**TRADEMARKS**

IBM, the IBM logo, and ibm.com are trademarks or registered trademarks of International Business Machines Corp., registered in many jurisdictions worldwide. Other product and service names might be trademarks of IBM or other companies. A current list of IBM trademarks is available on the web at “Copyright and trademark information” at www.ibm.com/legal/copytrade.shtml.

Adobe, the Adobe logo, PostScript, and the PostScript logo are either registered trademarks or trademarks of Adobe Systems Incorporated in the United States, and/or other countries.

Cell Broadband Engine is a trademark of Sony Computer Entertainment, Inc. in the United States, other countries, or both and is used under license therefrom.

Intel, Intel logo, Intel Inside, Intel Inside logo, Intel Centrino, Intel Centrino logo, Celeron, Intel Xeon, Intel SpeedStep, Itanium, and Pentium are trademarks or registered trademarks of Intel Corporation or its subsidiaries in the United States and other countries.

IT Infrastructure Library is a Registered Trade Mark of AXELOS Limited.

ITIL is a Registered Trade Mark of AXELOS Limited.

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

Linear Tape-Open, LTO, the LTO Logo, Ultrium, and the Ultrium logo are trademarks of HP, IBM Corp. and Quantum in the U.S. and other countries.

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

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

UNIX is a registered trademark of The Open Group in the United States and other countries.

© Copyright International Business Machines Corporation 2020.

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

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

**Table of Contents**

[0. Introduction 4](#_Toc153007310)

[0.1 Pre-requisites 4](#_Toc153007311)

[1. Build a decision service to calculate speeding fines 5](#_Toc153007312)

[1.1 Before you start 6](#_Toc153007313)

[1.2 Run the Decision Service 8](#_Toc153007314)

[1.3 Expose the decision service to the outside world 8](#_Toc153007315)

[1.4 Test the Decision Service 10](#_Toc153007316)

[2. Generate an OpenAPI specification 12](#_Toc153007317)

[3. Import the OpenAPI into watsonx Orchestrate 14](#_Toc153007318)

[4. Persisting data using Cloudant 17](#_Toc153007319)

[4.1 Local host Prerequisites 17](#_Toc153007320)

[4.2 Run on local host 17](#_Toc153007321)

[4.3 Code Engine Hosting Prerequisites 18](#_Toc153007322)

[4.4 Deployment to code engine on IBM Cloud 19](#_Toc153007323)

[5. Conclusion 20](#_Toc153007324)

# Introduction

In this Lab you will connect a decision service to watsonX Orchestrate and persist the results to a Cloudant database.

The Lab is structured into four sections.

* Build a decision service to calculate speeding fines.
* Generate an OpenAPI for the decision service.
* Import the OpenAPI into watsonx Orchestrate.
* Save the decision to a Cloudant database.

## Pre-requisites

To perform this Lab, you need:

* An IBM watsonx Orchestrate SaaS account with Builder role.
* A local machine with VSC BAMOE V9.0 installed.
* A local machine with Eclipse IDE installed.
* Git Bash Command line installed on your local machine.

# Build a decision service to calculate speeding fines

In this task, you ...

* Create a decision service.
* Define the data and logic of the model in Decision Model and Notation (DMN).
* Run the model locally

When defined, the model takes a driver and speeding data as input and produces a penalty.

## Before you start

In this exercise, you will build and test the decision service using Visual Studio Code (VSC) which is the developer environment.

The prerequisites for this Lab are:

* A local Windows Machine with VSC and BAMOE V9.x plugins
* Git Bash shell

1. Open Git Bash by clicking the Git Bash  icon on the Windows **Taskbar** at the bottom of the screen or in the Windows main screen.
2. Within the Git Bash shell, clone the rule project using the command:

git clone <https://github.com/ncrowther/bamoe2wxo>

1. Hit return. You should see the git repository cloned into the local drive:

Text

Description automatically generated

Now open the decision project with the VSC:

cd bamoe2wxo/

code SpeedingDecisionService/

1. VSC will open. See below:

A screenshot of a computer

Description automatically generated

1. On the left-hand side of the Visual Studio Code in the file explorer locate *src\main\resources\TrafficViolation.dmn*

**A screenshot of a computer

Description automatically generated**

1. Double-click the DMN file to view it. This opens the DMN Editor:

A screenshot of a computer

Description automatically generated

## Run the Decision Service

In this step we are going to build and run our decision service in development mode using Quarkus. Quarkus is a container-native Java stack. For more details see <https://quarkus.io/>

1. Within the VSC editor, click on the **Terminal** menu at the top of the screen:



1. In the drop down click on **New Terminal**.

A screenshot of a computer

Description automatically generated with medium confidence

1. In this new terminal window, paste the following:

mvn quarkus:dev

1. Hit return to start the Maven build. You may see warnings in the console. Once built, our decision model is automatically deployed to Quarkus and ready to run in development mode. You should see this:

A computer screen with text

Description automatically generated

## Expose the decision service to the outside world

1. Within the VSC editor, click the PORTS tab:

A screen shot of a computer

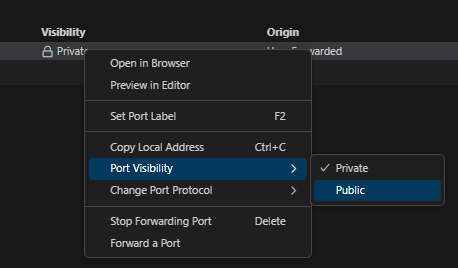
Description automatically generated

1. Select the Forward a Port button and
2. Enter the kogito service port (usually 8080) as the Port and hit return. You should now see this:

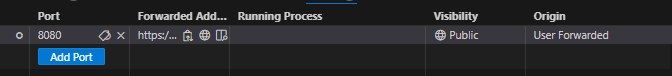
A black background with white text

Description automatically generated

In the Visibility column, click the value *Private* and change its port visibility to *Public*:

1. 
2. Accept the warning message by pressing *Continue*:
3. A screenshot of a computer error message

   Description automatically generated
4. You should now see this:



## Test the Decision Service

Let’s look at the API generated by Quarkus.

1. Open Chrome
2. Click the new tab button (+) and paste the following URL into the browser address field and press **Enter**.

[localhost:8080/q/swagger-ui](localhost:8123/q/swagger-ui)

**Note:** port *8080* is the default port, your port may be different. You can find the port in the Quarkus terminal output above. You should see the following:  
A screenshot of a computer

Description automatically generated

We can see the server URL we created in the previous section, and three endpoints. It’s the two POST endpoints that we are interested in as they execute the model; the only difference between them is that the second one includes additional trace information about intermediate decisions.

1. Click on the **POST /TrafficViolation** endpoint to expand it, then click the **Try it out** button on the right-side.

A white rectangular object with a black border

Description automatically generated

1. Press *Execute* and then paste the generated curl command into a bash command window (you can use the Git Bash command line you used earlier):

curl -X 'POST' \

'https://p05122wf-8080.uks1.devtunnels.ms/TrafficViolation' \

-H 'accept: application/json' \

-H 'Content-Type: application/json' \

-d '{

"Violation": {

"Code": "1",

"Date": "2023-12-07",

"Type": "speed",

"SpeedLimit": 50,

"ActualSpeed": 90

},

"Driver": {

"Name": "Jim Flanders",

"Age": 40,

"State": "NY",

"City": "New York",

"Points": 0

}

}'

1. Using the above command, you should see the result of:  
     
   Suspend":"No","Fine":{"Points":7,"Amount":1000}}
2. Congratulations, you have built and deployed your decision service on your local machine and exposed it to the outside world!

# Generate an OpenAPI specification

1. Open the project in this directory with the Eclipse IDE.
2. A screenshot of a computer program

   Description automatically generated
3. Edit *./data/config.json*
4. Set the decisionId to the name of the traffic violation decision service (not the filename).  Also set both internal and external URLs to the URL of your running Kogito service defined in the section above.
5. A screenshot of a computer

   Description automatically generated
6. Save the config.json file
7. Generate the OpenAPI file by running the default target in ./build.xml.
8. Once finished, hit *f5* on the generated folder to refresh it. You should see the generated file *TrafficViolation.json:*

A screenshot of a computer

Description automatically generated

# Import the OpenAPI into watsonx Orchestrate

1. Login to IBM watsonx Orchestrate
2. Under skills, select add skills from Files then drag and drop the generated open API spec:

A screenshot of a computer

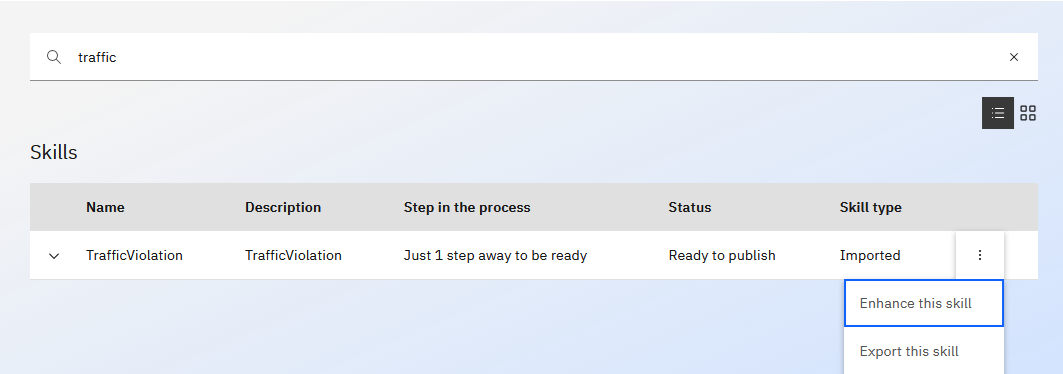
Description automatically generated

1. Click Next. Select the *TrafficViolation* skill and press Add.

A screenshot of a computer

Description automatically generated

1. Find the *TrafficViolation* skill.
2. Right click the button and select *Enhance this skill*:



1. Publish the skill:
2. A close up of a logo

   Description automatically generated
3. Click on Home:
4. 
5. Click *Add skills from the catalog*
6. *Find the TrafficViolation skill and select Add skill*

A white background with black text

Description automatically generated

1. Import the skill into your home skills
2. Click on Connect app  and enter any username and password (these credentials are not used).
3. Go back to Home and run the skill by typing *trafficviolation*
4. Enter the following data:

"Violation": {

"Code": "1",

"Date": "2023-12-07",

"Type": "speed",

"SpeedLimit": 50,

"ActualSpeed": 90

},

"Driver": {

"Name": "Jim Flanders",

"Age": 40,

"State": "NY",

"City": "New York",

"Points": 0

}

The response should be:

**Points: 7**

**Fine Amount: 1000**

# Persisting data using Cloudant

## Local host Prerequisites

1. Create a Cloudant instance in your IBM Cloud account.
2. See: <https://cloud.ibm.com/docs/Cloudant?topic=Cloudant-getting-started-with-cloudant>
3. Create a Cloudant database with name *wxodb.*
4. Create credentials.

## Run on local host

1. Edit *setenv.bat* to Cloudant credentials (see above).
2. Open DOS prompt (NOT powershell).  Enter:

   setenv.bat

 Start a local nodejs server:

   npm start

## Code Engine Hosting Prerequisites

1. Follow Local Host Prerequisites above.
2. Create a *Code Engine* instance in IBM Cloud account.
3. Create a code engine project called cloudant-façade, with the image referencing of your docker image.

A screenshot of a computer

Description automatically generated

1. Within your CodeEngine project, select Secrets and configmaps. Press Create.
2. Create a configmap with name *dataserviceconfig* and press *Create.*
3. Create a code engine *configmap configuration* with your Cloudant credentials:

A screenshot of a computer

Description automatically generated

## Deployment to code engine on IBM Cloud

1. Open Git Bash shell from VSC
2. Login to IBM Cloud.

ibmcloud login --sso

1. Select your resource group. E.g. *default*

ibmcloud target -g default

1. Select the code engine project: e.g

ibmcloud ce project select -n [PROJECT\_NAME]

1. Start Docker Desktop
2. In Bash Shell, go to the folder where you cloned the git repo and go to the dataservice folder e.g: /bamoe2wxo/cloudantFacade/dataservice
3. Within this folder, edit *build.sh* and *run.sh* and change the REGISTRY to your Docker registry.
4. Using the same bash shell, deploy the sample application to your docker repo:

./build

1. Deploy the application to Code Engine on IBM Cloud. From the app's folder do:

./run

1. Open your IBM Cloud app route in the browser and copy the URL
2. Using curl, execute the following command:

# Conclusion

In this lab were crated a decision service and then invoked it from wastonx Orchestrate. We then created a composite flow to store the results of the decision into a Cloudant database.

Thank you for taking this lab and I hope you create new applications using the technology featured in this lab.