**Lab Guide**

IBM Business Automation Manager Open Edition

Nigel Crowther – ncrowther@uk.ibm.com

Hands-on Lab

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 connect watsonX Orchestrate to BAMOE and Cloudant.

The Lab is structured into four sections.

* Build a speeding ticker decision service
* Generate an OpenAPI for this decision service.
* Invoke the decision service as a watsonx Orchestrate skill.
* Save the results of 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.
* Git Bash on your local machine.
* Java

# Build a decision service to calculate speeding fines

In this task, you ...

* Import a speeding ticket decision service.
* Run the decision locally.
* Expose the decision to the outside world

## Before you start

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

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 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
2. If not already done so, install the BAMOE Developer Tools extension:

A black background with white text

Description automatically generated

1. Double-click the DMN file to view it in the BAMOE DMN Editor:

A screenshot of a diagram

Description automatically generated

1. Familiarize yourself with the two inputs and single decision table.

## 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

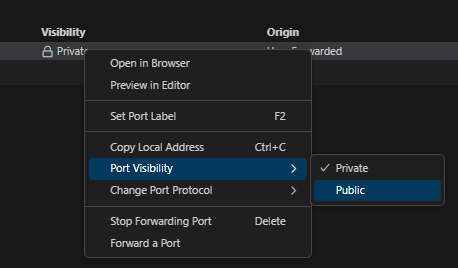
Description automatically generated

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:

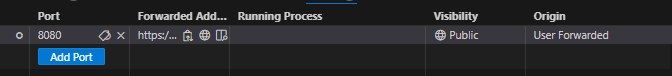
A black background with white text

Description automatically generated

1. Right-click the *Visibility* column and change *Port Visibility* from *Private* to *Public*:



1. Accept the warning message. You should now see this:



## Test the Decision Service

1. Open the BAMOE OpenApi Generator using VSC:

code OpenApiGenerator/

1. VSC will open. See below:

Let’s look at the API generated by Quarkus.

1. Open your favorite browser.
2. Paste the following URL into the browser address 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 your port in the Quarkus terminal output. If successful, you should see this:  
A screenshot of a computer

Description automatically generated

We can see three endpoints; It’s the two POST endpoints that we are interested in. 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
2. Press *Execute* and paste the curl command into the bash command:

curl -X 'POST' \

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

-H 'accept: application/json' \

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

-d '{

"Violation": {

"Date": "2023-12-09",

"SpeedLimit": 20,

"ActualSpeed": 50

},

"DriverId": "D123456"

}'

1. You should see the result containing the fine:  
     
   "Fine":{"Points":7,"Amount":1000}
2. Congratulations, you have built and deployed your Speeding Fine decision service on your local machine and exposed it to the outside world!

# Generate an OpenAPI specification for the Decision Service

* 1. Open the WoApiGenerator in VSC

code OpenApiGenerator/

* 1. The VSC editor should appear with the OpenApiGenerator folder.

A screenshot of a computer program

Description automatically generated

1. Expand the *data* folder and edit *config.json*
2. The *decisionId* is the name of the decision service. In this lab it is *TrafficVioloation*. The internal URL should be set to the local URL of your running Kogito service. The external URL is the Address in the Port forwarding defined in section 1.3.8.

A screen shot of a computer

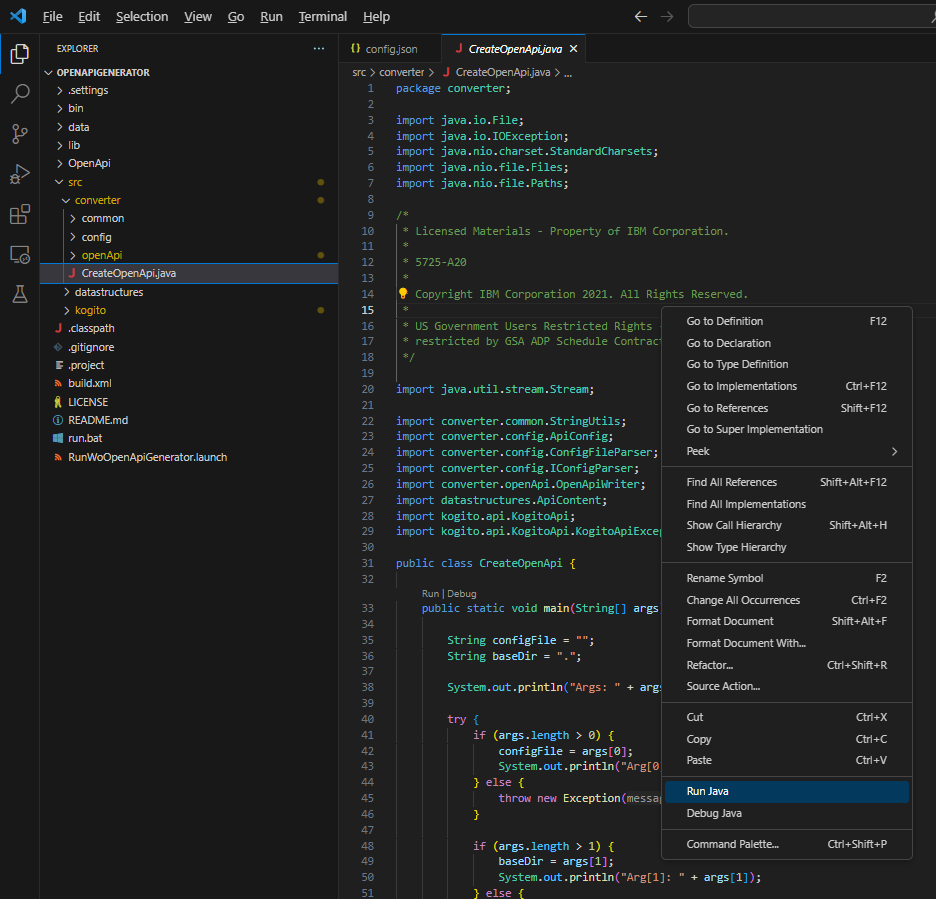
Description automatically generated

1. Save *config.json*
2. Now open *CreateOpenApi.java* in the *src/converter* folder:

A screenshot of a computer

Description automatically generated

1. Right-click anywhere in the code and select *Run Java*:



1. You should see the generator run and create a file in *.\generated\TrafficViolation.json :*

A screen shot of a computer

Description automatically generated

Open Trafficviolation.json and paste it into https://editor.swagger.io

You should see the same decision service we inspected in section 1.4

A screenshot of a computer

Description automatically generated

If you look closely at the Yaml you will see the following differences to that in section 1.4:

* *x-ibm* annotations have been added
* The data model has been appended
* Only a single TrafficVioloation endpoint is generated.

This OpenAPI is now ready to be imported into watsonx Orchestrate, which is what we will do in the next section.

# 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 *TrafficViolation.json* API spec:

A screenshot of a computer

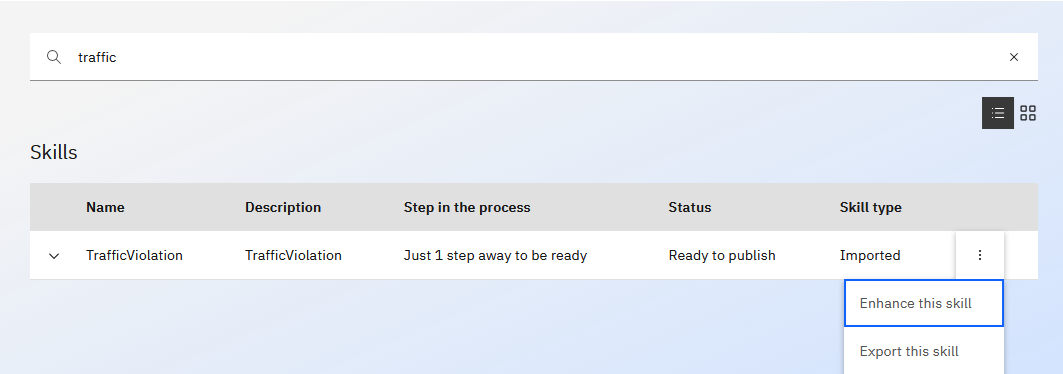
Description automatically generated

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 with three vertical dots, and select *Enhance this skill*:



1. Publish the skill:

A close up of a logo

Description automatically generated

1. Click on *Home*:
2. Click *Add skills from the catalog*
3. Find the *TrafficViolation* skill (type in the full name to find it). Select it and press *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:

A screenshot of a computer

Description automatically generated

Hit *Apply*. The response should be:

**A screenshot of a computer

Description automatically generated**

Congratulations, you have run your Decision Service from watsonx Orchestrate!

# Persist Decisions

In this section we call a database skill to store the speeding tickets. You will create your own database skill instance in section 5, but in this section we will use a public instance that is already created.

* 1. Open the *cloudantFacade* in VSC

code cloudantFacade/

* 1. The VSC editor should appear with the *cloudantFacade* folder.

A screenshot of a computer

Description automatically generated

* 1. Expand the *openapi* folder and edit *dataApi.yaml*
  2. Open *dataApi.yaml* and paste it into <https://editor.swagger.io>
  3. You should see the Apis defined below:

A screenshot of a computer

Description automatically generated

1. Now we will import *dataApi.yaml* into watsonx Orchestrate
2. Login to **IBM watsonx Orchestrate**
3. Under *skills*, select *add skills* from *Files* then drag and drop the *dataApi.yaml* API spec:

A screenshot of a computer

Description automatically generated

1. Click Next. Select the threeskills and press *Add*.

A screenshot of a computer

Description automatically generated

1. Find each skill and publish it as described in section 3
2. You should now have all three skills been published under the *cloudantWxoFacade*, and added to your catalog. Run the skill *write a doc to cloudant.* Enter a *driverId* containing your initials and six random numbers to make it unique:

A screenshot of a phone

Description automatically generated

Now run the *get docs from cloudant* skill. You should see data retrieved from the database. Find the entry just created by using the search  button:

A screenshot of a computer

Description automatically generated

Now under *Skills*, add a skill flow called *[YourInitials]\_SpeedingDatabase.* The composite skill flow should look like this.

A white rectangular object with text

Description automatically generated

Publish the skill and add it to your catalog.

Run the skill. Enter a violation like the one below:

A screenshot of a computer

Description automatically generatedAfter a while you should see your decision along with all others:

A screenshot of a computer

Description automatically generated

# Advanced – Create Cloudant Façade for Watsonx Orchestrate

In this section we implement the Cloudant skill used in the previous section.

## 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.