Build a smart traffic management app for connected cars using Watson IoT Platform and Node-RED on Bluemix

Use Business Rules, Geospatial Analytics, Insights for Weather and Watson AlchemyAPI

Author: U. Siddiqui, usiddiqu@us.ibm.com

Introduction

The Watson Internet of Things (IoT) Platform on IBM Bluemix enables your applications to use information from connected devices. You can use additional cognitive, data and analytics services on Bluemix to augment the data received from the connected devices, and instrument the intelligence of your application. This tutorial shows how to manage traffic for connected cars by adjusting their routes and speeds depending on current traffic and weather conditions in traffic zones. In the case of accidents, the application determines the accident severity and the need for emergency medical services using cognitive sentiment analysis, and the difference in speeds of the cars.

Traffic management scenarios

For the purposes of this tutorial, a traffic zone can be of the following types: Residential, Industrial, Commercial, Construction or Default. The speed limit for a traffic zone depends on its type and current weather conditions (for example, if it is raining in that zone). The level of traffic density also depends on the type of a traffic zone (for example, 5 cars in a Residential zone may be considered high density, but 5 cars in a Commercial zone may be considered low density).

A traffic zone can also be an Accident zone that was created around the scene of an accident.

You will model the following scenarios for managing traffic in the traffic zones.

Scenario 1: When a car enters a traffic zone

- Direct the car to adjust its speed depending on the speed limit of the traffic zone. Business rules dynamically determine the speed limit of the traffic zone depending on its type and if it is raining in the zone (using the Insights for Weather service).
- Direct the car to reverse out of the zone if it is highly congested. Business rules determine the level of traffic density (HIGH, MEDUM or LOW) depending on the type of the traffic zone, and the current number of cars and accidents in the zone.
- Direct the car to reduce its speed and reverse out of an Accident zone.

Scenario 2: When an accident occurs between two cars

Determine the accident severity and if medical attention is needed, using:

- The difference in speeds of the cars at the time of the accident
- The sentiment of the drivers' response to the question "Are you doing alright?" asked immediately after the accident. The sentiment analysis is performed using the Watson AlchemyAPI service.

The time it takes for an accident to be cleared depends on the severity of the accident.

Traffic management implementation

To simulate connected cars in an urban environment, this tutorial uses an extended version of the Connected Vehicle Starter Kit by Bryan Boyd. It uses Geospatial Analytics to detect when cars exit or enter a geofence, as detailed in the tutorial Build a connected-car IoT app with Geospatial Analytics (also

by Bryan Boyd). A Node-RED app invokes the Business Rules service for decision making in response to IoT alerts, in conjunction with the Watson AlchemyAPI and Insights for Weather services.

After you complete this tutorial, you can use the power of business rules to implement flexible decision management in an IoT solution, in conjunction with other cognitive, data and analytics services on Bluemix.

Step 1. Set up the traffic simulator kit

Start by deploying the traffic simulator kit, which extends the Connected Vehicle Starter Kit (henceforth referred to as the connected vehicle kit). Read the overview of the connected vehicle kit in the tutorial Build a connected-car IoT app with Geospatial Analytics (henceforth referred to as the connected car tutorial).

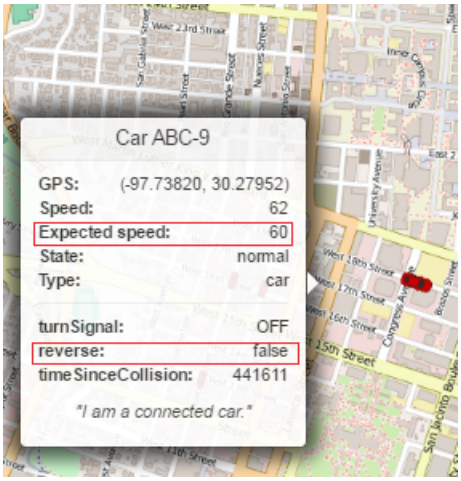
1. Create a Bluemix account and a DevOps Services account, both linked to your IBM ID. Log into the Bluemix console using your Bluemix account, and create a space called `dev`.
2. Configure the Watson IoT Platform service by following **Steps 1 and 2** of the connected car tutorial. In **Step 2**, create one device (e.g. `ABC`) of the type `vehicle`, instead of three.
3. Configure and deploy the traffic simulator kit by following **Step 3** of the connected car tutorial, with the following modifications:
   - Clone the usiddiqu | trafficsim-usiddiqu project on DevOps Services.
   - In **Step 3.5**, keep the default value of `1` instance in manifest.yml.
   - In **Step 3.6**, for `iot_deviceSet`, enter the details only for the single device that you created (`ABC`) in config/settings.js.
   - In **Step 3.10**, in the Map app at http://*app-name*.mybluemix.net, you will see 15 simulated vehicles for the `ABC` device. They are named `ABC-1`.. `ABC-15`. You can skip **Step 3.11**.
4. Configure the Geospatial Analytics service by following **Step 4** of the connected car tutorial. Make sure you start the Geospatial Analytics service (browse to http://`app-name`.mybluemix.net/GeospatialService_start and wait for the page to show success.) Create a geofence in the Map app by clicking the "!" button in the toolbar. Observe how the Map app consumes geofence alerts to display notifications when a vehicle enters or exits a geofence.

Step 2. Explore the extensions in the traffic simulator kit

You will explore how the traffic simulator kit has extended the connected vehicle kit to model traffic zones, and also accidents between cars.

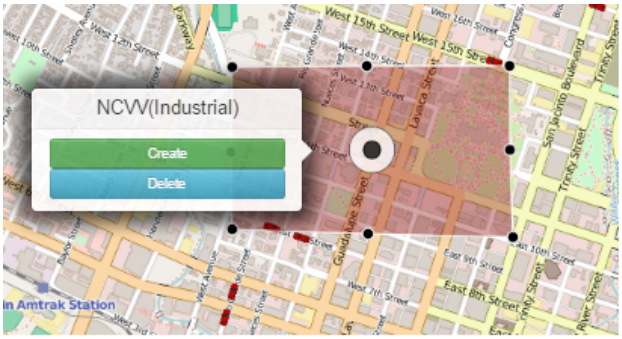
1. Explore the extensions to the behavior of the connected cars.

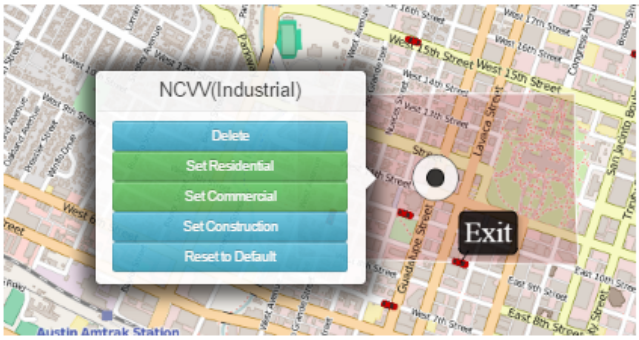
   - If you click a car, you will notice the `reverse` property. The connected vehicle simulator allows a car to reverse back if its `reverse` property is set to `true`. (If it is not possible to reverse back, the car continues along a random path.)
   - Also, you will notice the `expectedSpeed` property. When the `expectedSpeed` property of a car is set, the actual speed of the car (modeled by the `speed` property) is adjusted to within 10mph of the expected speed.



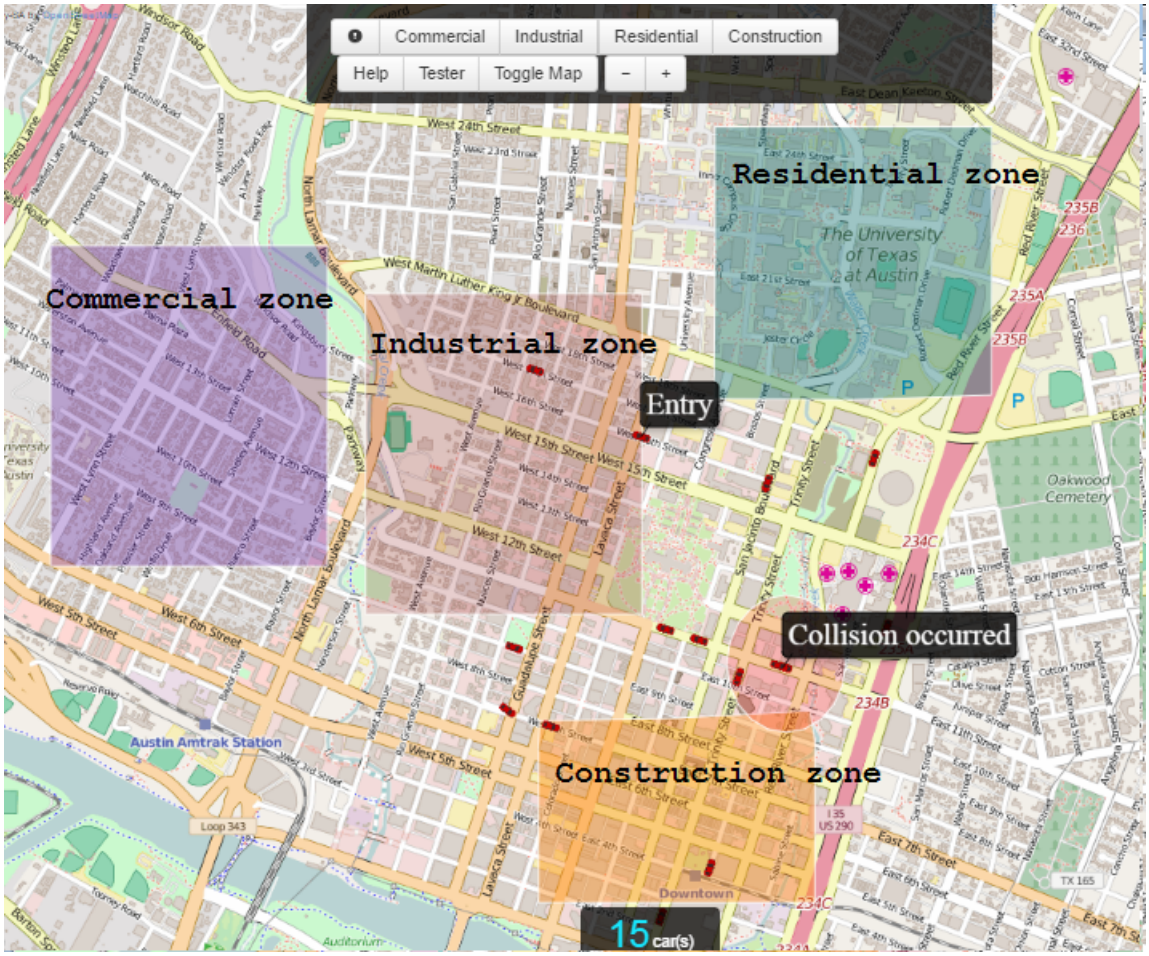
2. Explore the extensions to the Map app to model traffic zones

   - The Map app associates geofences with traffic zones. Click the Industrial button on the toolbar to create an Industrial traffic zone in the center of the view, and click Create after adjusting the boundaries of the zone. Click the center black circle of the zone, once it has been created. You will see the type displayed along with its name, and also buttons allowing you to change the type of the zone.





   Use the toolbar buttons to create a traffic zone of each type (Commercial, Industrial, Residential and Construction). Each zone type has a different color to distinguish it. (You can also use the "!" button to create a Default zone.) The Map app publishes a `trafficZones` alert with a list of all the zones and their types each time a traffic zone is created, deleted or its type is changed.



Note that the type of each zone is not yet persisted. If you refresh or restart the Map app, each zone will default to the “Default” type, and you will have to reset the type of each zone. You can extend this kit to persist the zone types in Cloudant NoSQL database on Bluemix, for example.

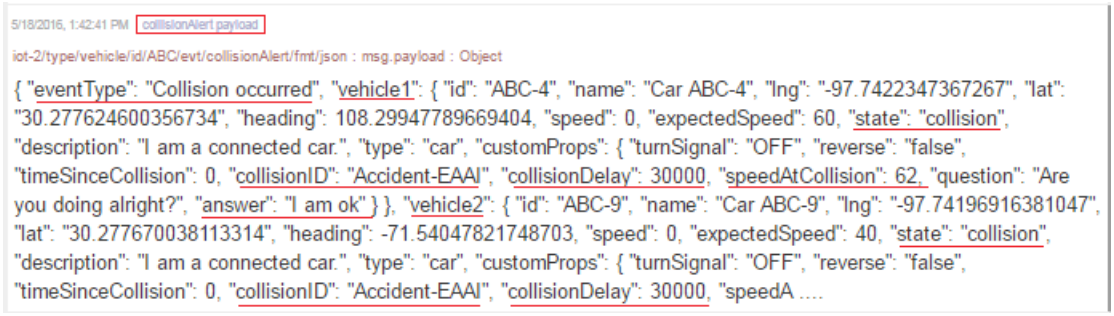
- The Map app publishes traffic alerts when cars enter or exit a traffic zone, in response to the geofence alerts published by the Geospatial Analytics service. Make sure the Geospatial Analytics service is started (*Step 1*) to be able to generate the traffic alerts. The traffic alerts' payload shows the `eventType` (“Exit” or “Entry”), the ID of the car that entered or exited the traffic zone, and the current state of the traffic zone (including its type and the cars and accidents currently in the zone).

5/18/2016, 8:10:19 AM trafficAlert:payload
iot-2/type/api/id/traffic/cmd/trafficAlert/fmt/json : msg.payload : Object
{ "eventType": "Entry", "vehicleId": "ABC-1", "trafficZone": { "name": "LDRK", "type": "Industrial", "numberOfCars": 4, "numberOfCollisions": 0, "trafficDensity": "", "status": "accident-free zone", "vehicles": ["ABC-8", "ABC-3", "ABC-2", "ABC-1"], "collisions": [], "longitude": -97.74015271641454, "latitude": 30.275761878811437 } }

You will use the contents of this payload to determine the speed limit and traffic density when a car enters a traffic zone, in *Steps 4-6*, to model Scenario 1.

3. Explore the extensions to model accidents. Both the connected vehicle simulator and Map app have been extended to simulate accidents between cars.

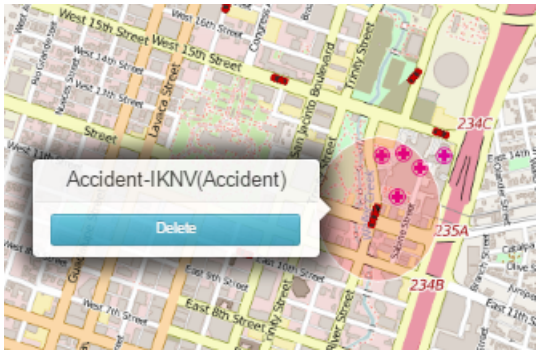
   - The connected vehicle simulator detects if two cars are involved in an accident, and publishes a collision alert. The collision alert payload shows the `eventType` ("Collision occurred"), and information about the two vehicles involved in the collision.



     You will use the contents of this payload to determine the severity of an accident, in *Steps 9* and *10*, to model Scenario 2.

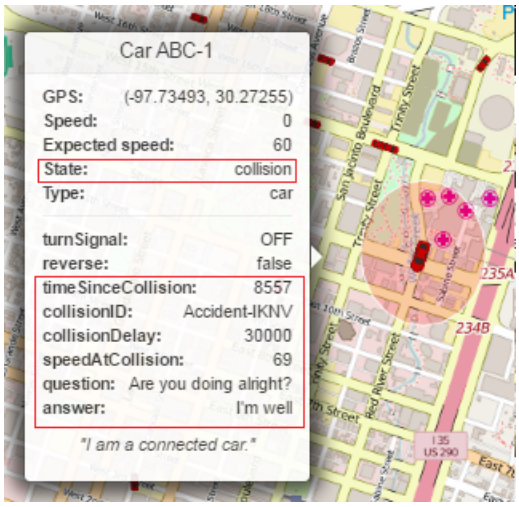
     The cars remain stationary until the accident is cleared, and then resume motion. The simulator publishes another collision alert (with the `eventType` "Collision cleared") to indicate that the accident has cleared.

   - The Map app subscribes to the collision alerts and creates red circular Accident zones around an accident while it has not been cleared, as shown in the following screen capture. Click the center black circle of an accident zone, to see the contextual menu which allows you to delete the Accident zone.



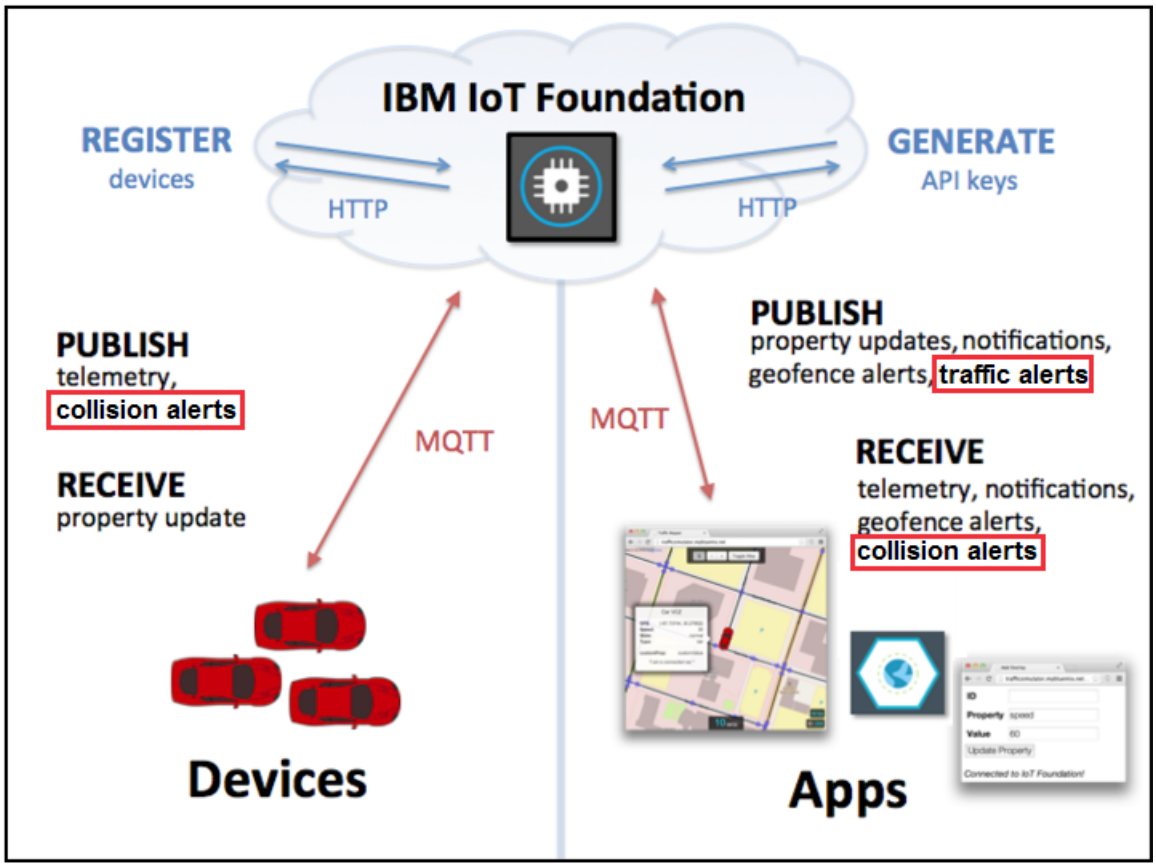
   - Click a car involved in the accident to see the custom properties that have been added for collision processing.
     - The time elapsed since the collision occurred, in milliseconds (`timeSinceCollision`)
     - The ID of the collision (`collisionID`). This corresponds to the name of the Accident zone in the screen capture above
     - The time needed to recover from the collision, in milliseconds (`collisionDelay`)
     - The speed of the car when the collision occurred (`speedAtCollision`)
     - The question "Are you doing alright?" asked to the driver immediately after the collision (`question`)

- The simulated answer of the driver to the question (`answer`)



The accident clears and the cars start driving again once the `timeSinceCollision` exceeds the `collisionDelay`. The default value for the `collisionDelay` is `30000` milliseconds.

The following diagram shows an overview of the messaging alerts in the traffic simulator kit. (This is an updated version of the overview diagram of the connected vehicle kit.)



Step 3. Create and configure the Node-RED app

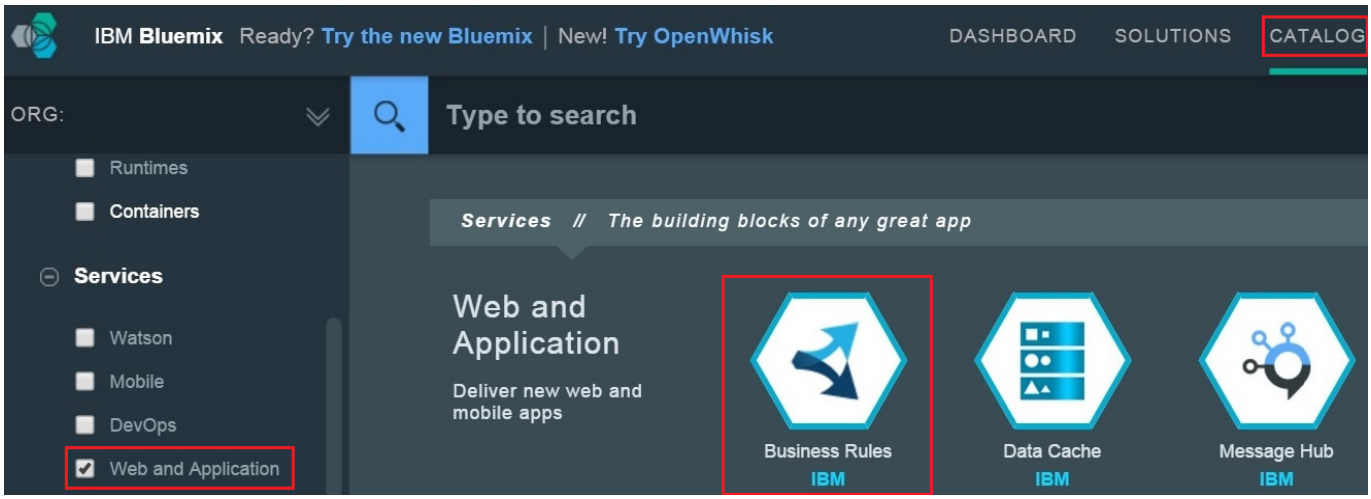
Next you will create and configure a Node-RED app which you will use to implement the traffic management scenarios.

1. Create and configure a Node-RED app by following **Steps 6.1-6.4** of the connected car tutorial.

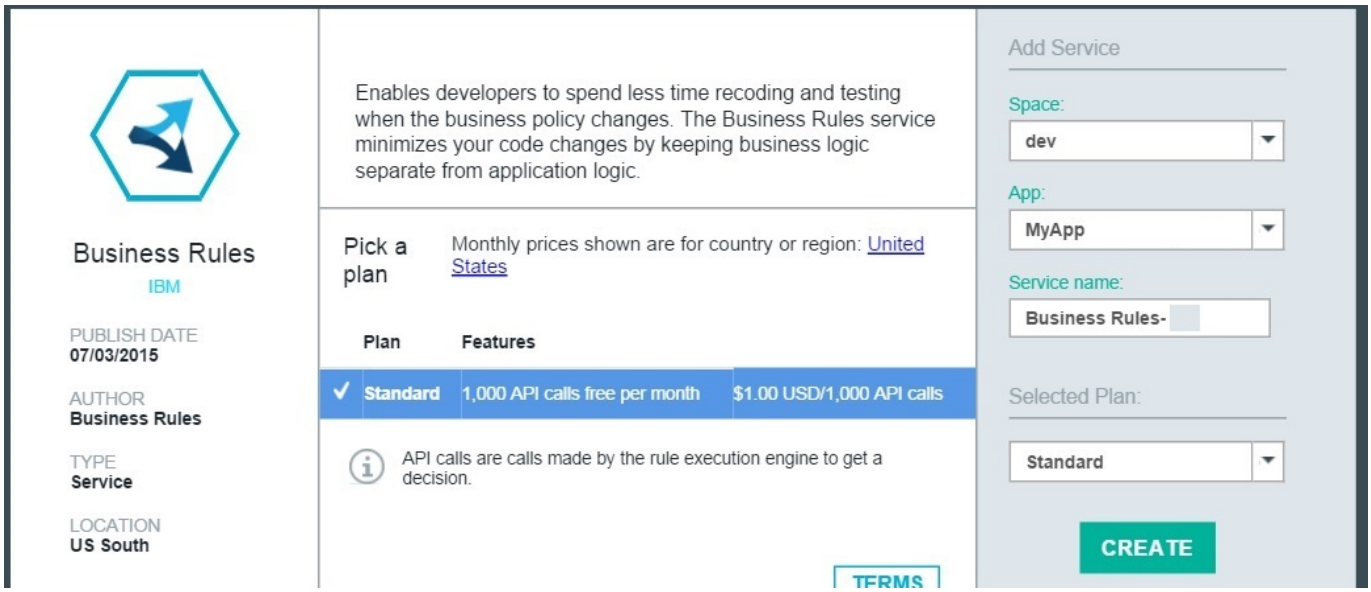
   Next you will bind Bluemix service instances (Business Rules, Insights for Weather and Watson AlchemyAPI) which the Node-RED app will use.

   (You can restage the Node-RED app after you have created and bound all three instances.)

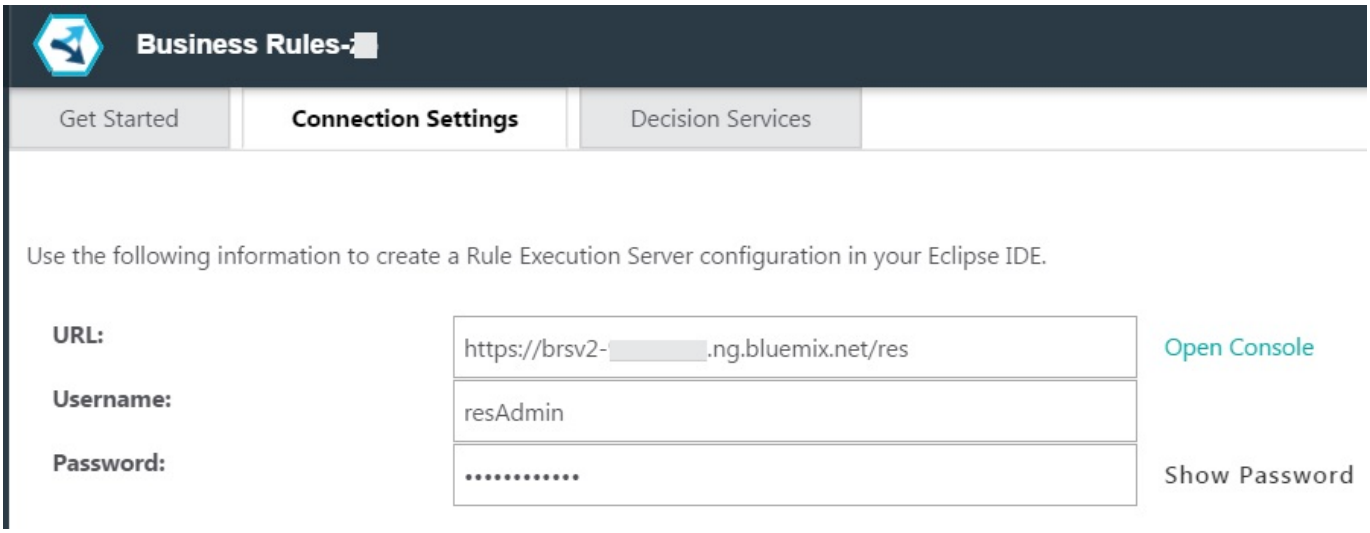
2. In the Bluemix Catalog, click the Business Rules service in the Web and Application category.



Specify the space `dev`, and the name of your Node-RED app to bind the Business Rules service instance to it. Press the **Create** button and do not restage the Node-RED app yet.

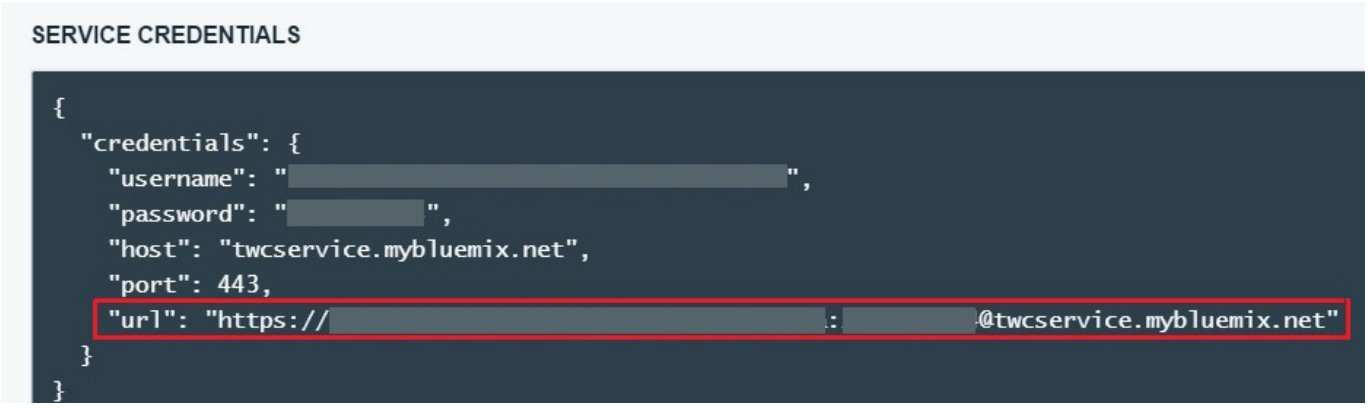


Explore the **Connection Settings** tab for the service instance, shown in the following screen capture. You will use the URL, Username and Password when deploying rules to this service instance in *Steps 5* and *9*, and when configuring the Node-RED flow in *Steps 6* and *10*.



3. Follow the instructions in the **Add the Insights for Weather service** section in the Getting started with Insights for Weather Bluemix documentation to bind the Insights for Weather service to your Node-RED app. Do not restage the Node-RED app yet.

   Go to the service instance to see the URL and credentials, as shown in the following screen capture. You will use these credentials when configuring the Node-RED flow in *Step 6* to model Scenario 1.



4. Follow the instructions in Getting Started with AlchemyAPI on Bluemix to create and bind an instance of the Watson AlchemyAPI service on Bluemix.

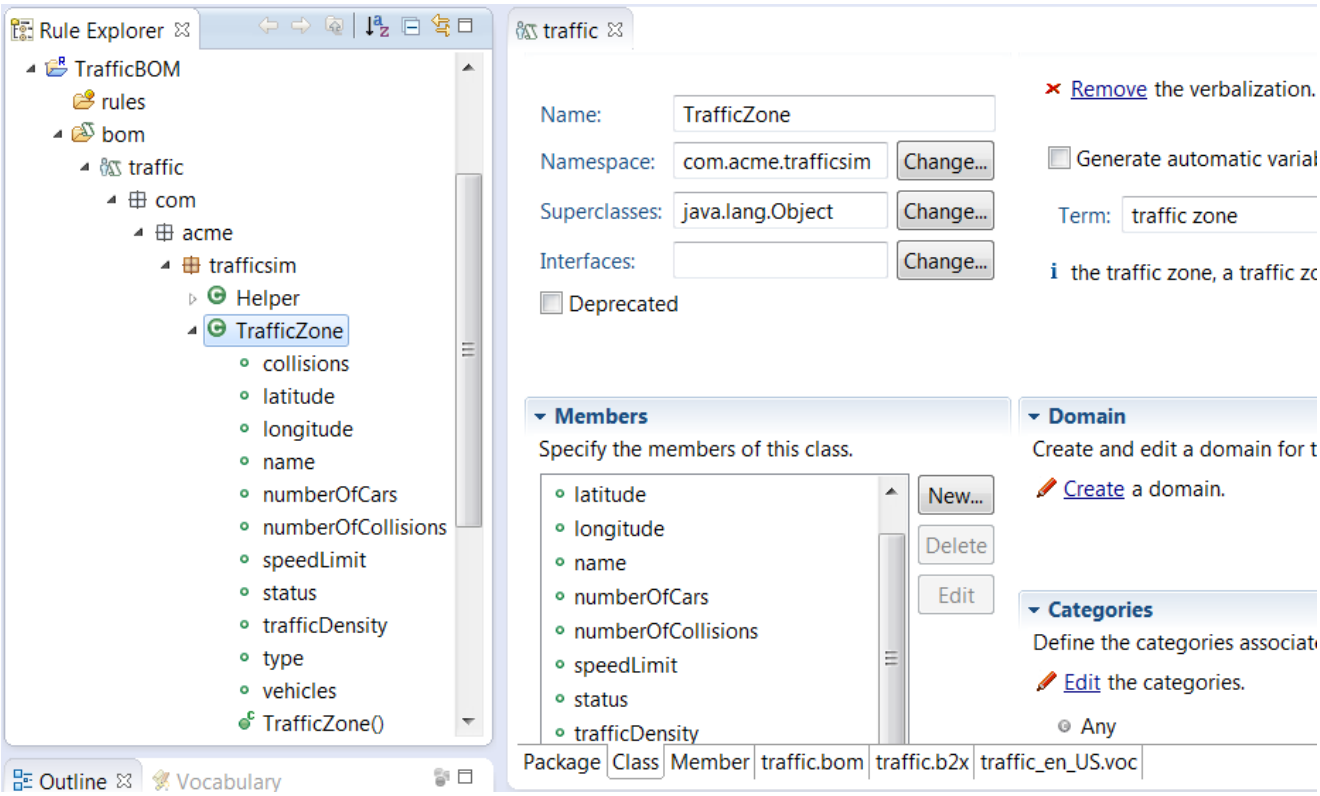
   Keep track of the access key for your service instance. You will use the access key in *Step 9* when configuring the Node-RED flow to model Scenario 2 to determine the collision severity.

5. If you have not restaged the Node-RED app yet, do so now.

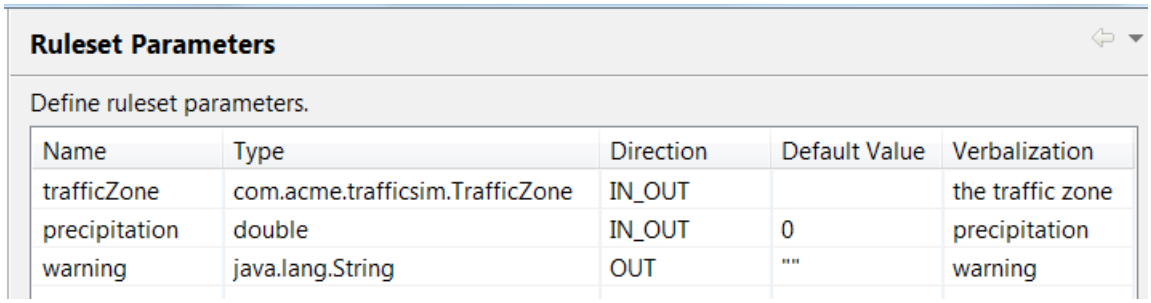
Step 4. Design traffic rules for Scenario 1 with Rule Designer

You will design the business rules which determine the speed limit and traffic density when a car enters a traffic zone (for Scenario 1). The rules will be executed using the Business Rules service.

1. Install the Rule Designer plugin for the Business Rules service on your Eclipse Juno 4.2.2 IDE.
2. Download the `TrafficXOM`, `TrafficBOM`, `TrafficRules` and `TrafficRuleApp` projects. Import them into your Eclipse workspace, and open them. Switch to the Rule perspective to open Rule Explorer.
3. Explore the the execution object model (XOM), which is the run-time model against which the rules are executed. The XOM is defined in the `TrafficXOM` Java™ project. Explore the `TrafficZone` class, which represents the traffic zones defined traffic simulator kit. Note that the attributes of this class correspond to the attributes of the `trafficZone` element of the `trafficAlert` payload (*Step 2*).
4. Explore the `TrafficBOM` project and the `traffic` business object model (BOM), which is created from the XOM in Rule Designer. This BOM entry is already created in the sample code for you to use in this tutorial. It contains the `TrafficZone` class that corresponds to the Java-defined class. The rules are written against the classes defined in the BOM.



5. Define the contract between the business logic and your client application using ruleset parameters. You can see the defined Ruleset Parameters by right-clicking on the `TrafficRules` project, and choosing Properties, and then clicking Ruleset Parameters in the left pane. The input-output parameter is named `trafficZone` and is of type `TrafficZone`. The rules will set the `speedLimit` and `trafficDensity` attributes for `trafficZone`. The ruleset also uses an input parameter `precipitation` as a measure of the rain in a traffic zone, and sets an output parameter `warning`.

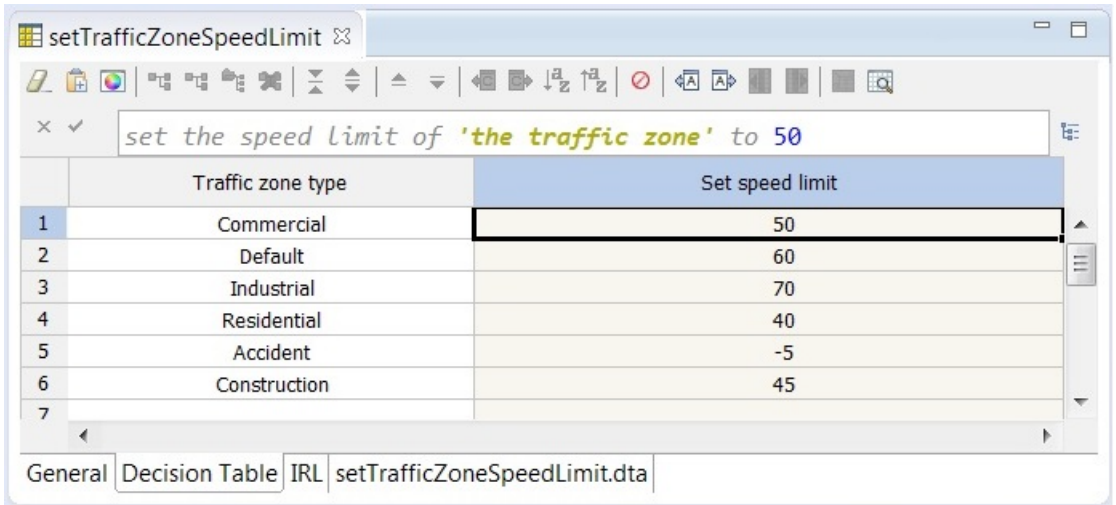


Ruleset Parameters

Define ruleset parameters.

Name	Type	Direction	Default Value	Verbalization
trafficZone	com.acme.trafficsim.TrafficZone	IN_OUT		the traffic zone
precipitation	double	IN_OUT	0	precipitation
warning	java.lang.String	OUT	""	warning

6. Explore the `TrafficRules` Rule project. The `rules/speedLimit` folder contains rules to determine the speed limit of traffic zone depending on its type. The value of `-5` for the Accident zone signifies that a car should reduce its speed by 5. A rule also adjusts the speed limit depending on the precipitation in the zone.



setTrafficZoneSpeedLimit

set the speed limit of ***'the traffic zone'*** *to* 50

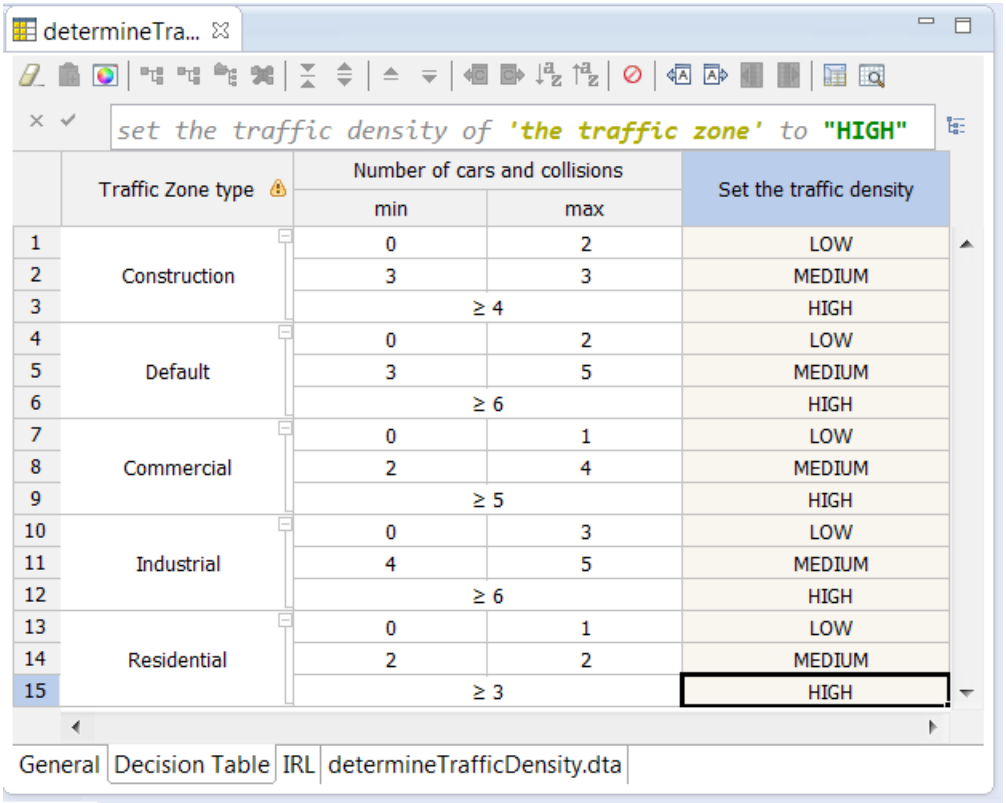
	Traffic zone type	Set speed limit
1	Commercial	50
2	Default	60
3	Industrial	70
4	Residential	40
5	Accident	-5
6	Construction	45
7		

General | Decision Table | IRL | setTrafficZoneSpeedLimit.dta

```
if precipitation is more than 0

then set the speed limit of 'the traffic zone'
                            to the speed limit of 'the traffic zone' - 5;
```

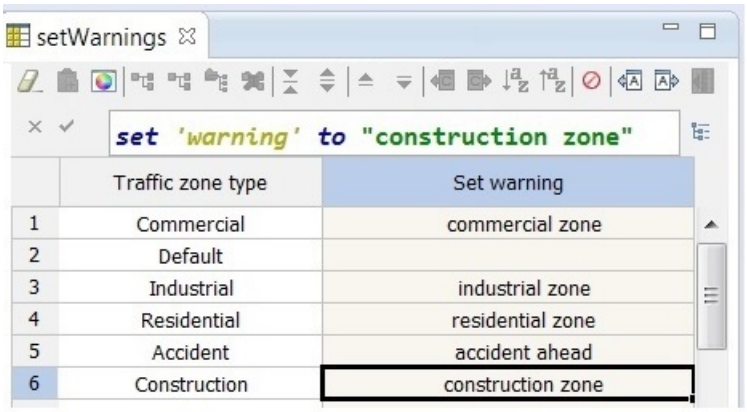
The `rules/trafficDensity` folder contains rules to determine the traffic density of a traffic zone, depending on its type and the number of cars and collisions in it.



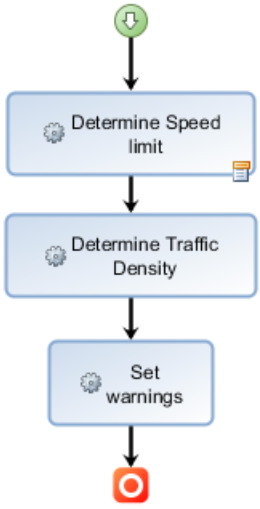

```
if the type of 'the traffic zone' is "Accident"

then
    set the traffic density of 'the traffic zone'  to "HIGH" ;
```

The `rules/warnings` folder contains rules to generate warnings. The `mainRuleflow` defined in the `rules` folder executes the rules in the specified order.




```
if precipitation is more than 0
then
 set warning to concatenate "wet weather alert" to warning ;
```



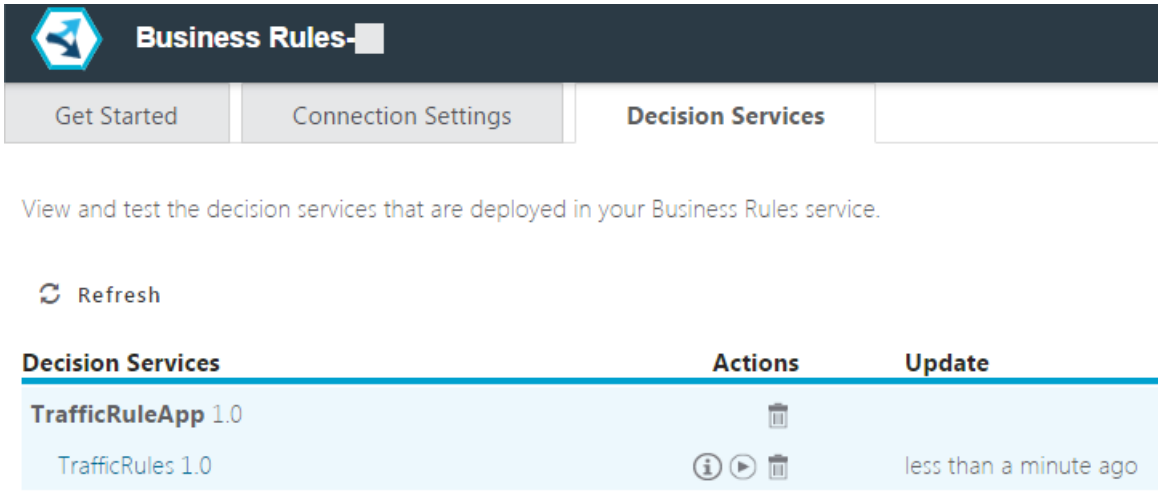
Step 5. Deploy the traffic rules and test the payload

To execute your business rules, you must deploy them to the Business Rules instance that you created in *Step 3*. The rules are deployed in archives that are called RuleApps, from the Rule Designer to the Rule Execution Server that is associated with the service instance. Complete the following steps.

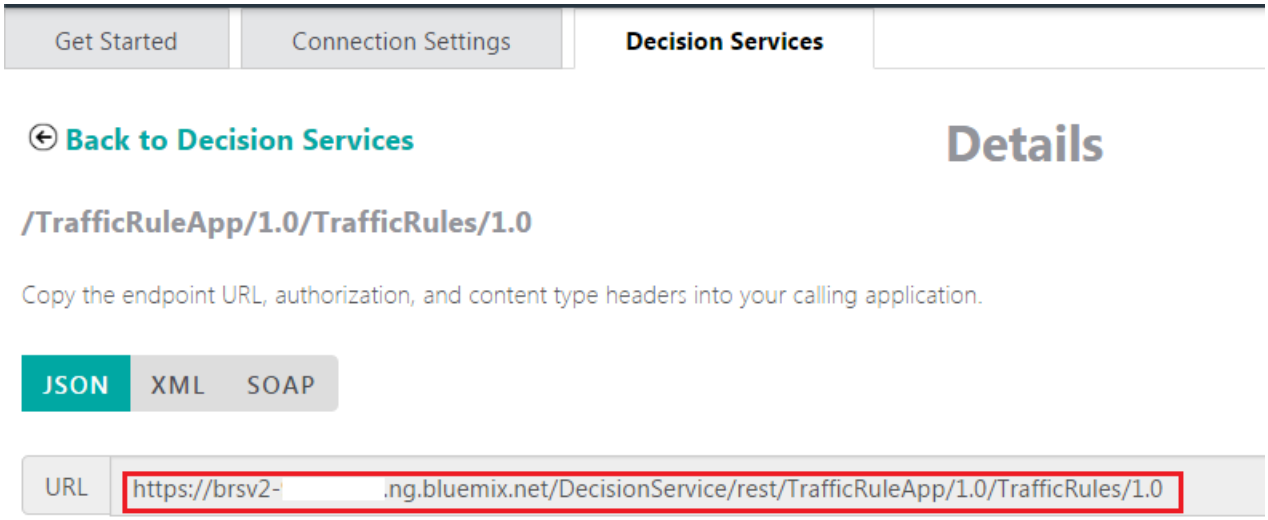
1. Create a Rule Execution Server configuration project for your Business Rules service instance
   - In the Rule Designer, click **File > New > Project > Rule Designer > Rule Execution Server Configuration Project** and click Next. Specify a project name, then click Next. Set the Application Server to IBM® WebSphere® AS 8.5, and click Next.
   - For the URL, Login, and Password in the Server Configuration window, retrieve the information from the Connection Settings tab of the Business Rules service instance (*Step 3*).
   - Click **Test Connection**, and then click Next and Finish.

2. Deploy your business rules to the service instance.
   - Right-click the `TrafficRuleApp` project, and select **RuleApp > Deploy**.
   - Then select the **Increment RuleApp major version** deployment type, as described in the Deploying the RuleApp to the Business Rules service instance Bluemix documentation.
   - On the **Deploy a RuleApp to Rule Execution Server** window, select the Rule Execution Server configuration for your Business Rules service instance and make sure that **Deploy XOM of rule projects and archives contained in the RuleApp** is selected.
   - Click Finish.

3. Go to your Business Rules service instance on the Bluemix dashboard, and select the **Decision Services** tab to see your deployed RuleApps and rulesets. Click **Refresh** if needed.



4. Click the ruleset name (TrafficRules 1.0) to navigate to the **Details** page. Note the URL for the ruleset, which specifies the versions of the RuleApp and ruleset set when you deployed the RuleApp, based on deployment type you selected. You use this URL in *Step 6* when you configure the Node-RED flow to execute the ruleset.

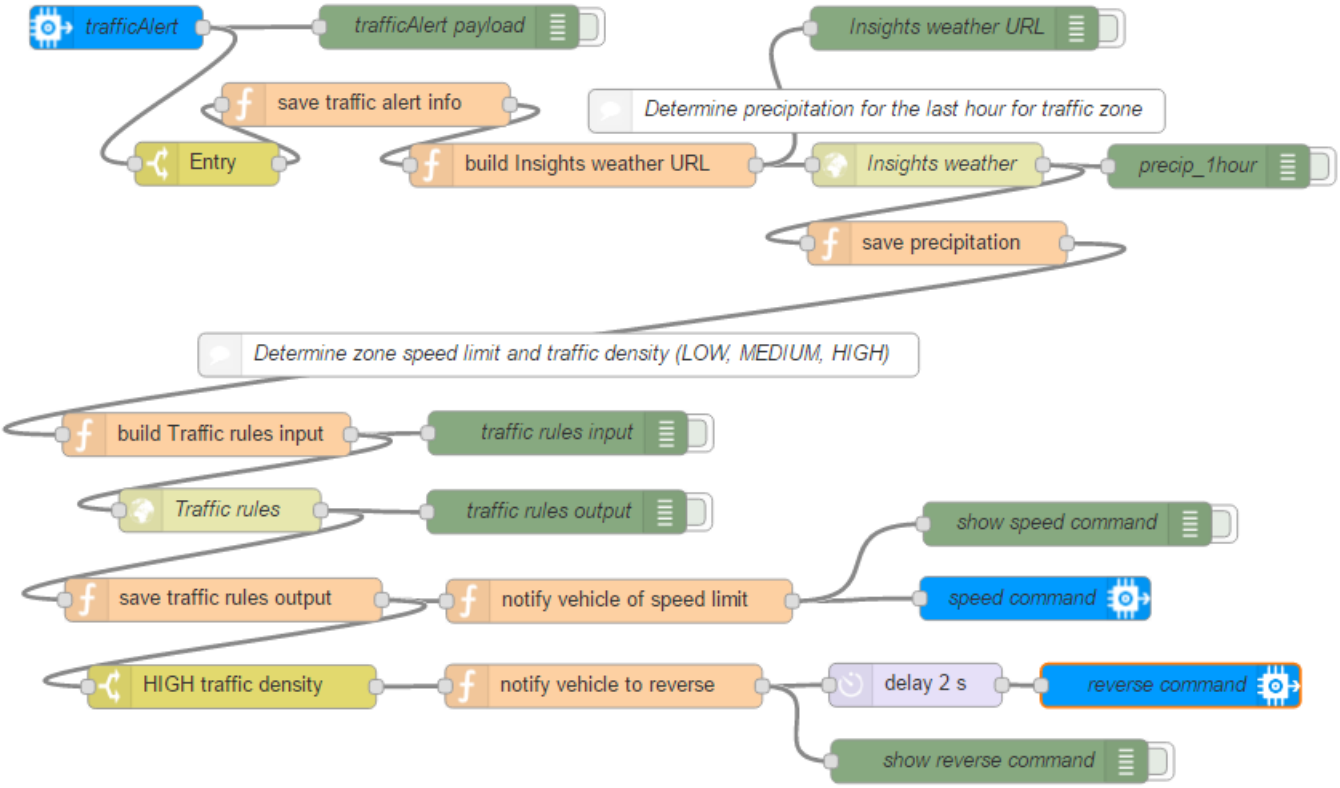


5. Test your ruleset payload from the Bluemix console.
   - From the ruleset **Details** page, click **Back to Decision Services** to go to the **Decision Services** page. To test the ruleset, click the arrow next to it. The **Test** page is loaded with the Decision Request and Decision Response panes. The Decision Request pane is populated with a sample payload in JSON format, and the Decision Response pane is empty.
   - Populate the sample payload with test data as shown in the following screen capture, and click **Run Test**. The ruleset is executed with the test data, and the response is displayed under Decision Response. The response contains values for the `speedLimit` and `trafficDensity` of the `trafficZone`, and the associated `warning`.

Step 6. Create the new Node-RED flow for Scenario 1

You will create a Node-RED flow in the Node-RED app to implement Scenario 1.

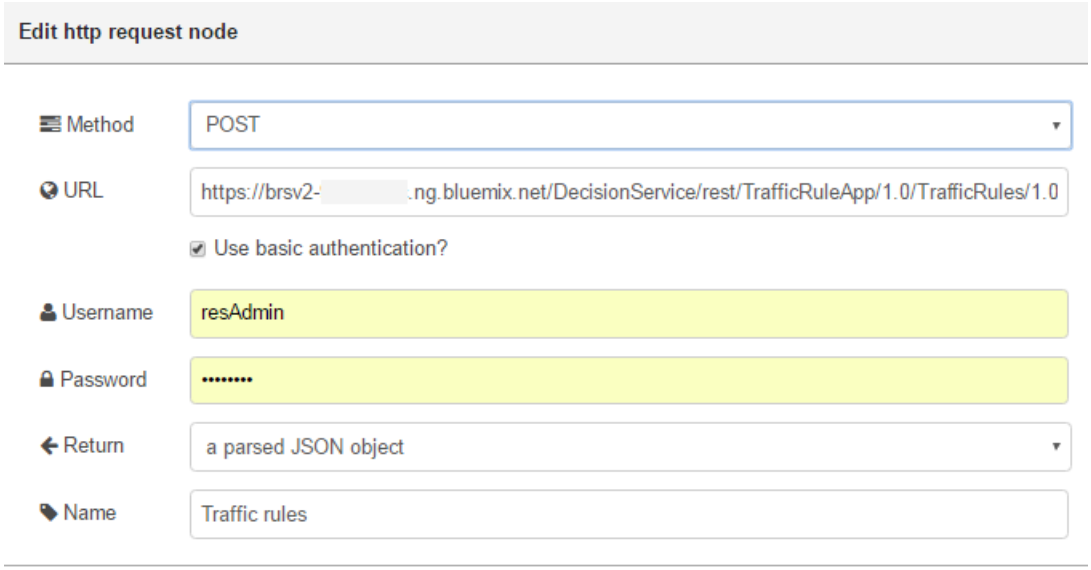
1. Understand the new flow. The following screen capture shows what the flow will look like.



- The nodes `trafficAlert`, `Entry` and `save traffic alert info` save the information from the traffic alert when a vehicle enters a traffic zone.
- The node `build Insights weather URL` builds the URL for the Insights for weather service using the service credentials. It also builds the parameters for the service to specify that the service should retrieve the current weather conditions for the geographical coordinates of the traffic zone. The nodes `Insights weather` and `save precipitation` call the Insights for Weather service and retrieve the precipitation for the last hour through the attribute `observation.metric.precip_1hour` from the response payload. Refer to the Getting current conditions Bluemix documentation for details on how to get the current weather using the Insights for Weather service, and the format of the response.
- The node `build Traffic rules input` builds the input parameters for the traffic rules using the `trafficZone` element from the traffic alert, and the precipitation from the Insights for weather service. These input parameters correspond to the parameters you designed for the traffic rules in *Step 4*. The nodes `Traffic rules` and `save traffic rules output` invoke the traffic rules deployed to the Business Rules service instance, and save the output. The rules determine the speed limit and traffic density of the traffic zone, and any associated warnings.
- The nodes `notify vehicle of speed limit` and `speed command` build and send the commands to notify the car to set its `expectedSpeed` property to the speed limit of the traffic zone. They also send a notification to display the traffic density and any warnings.

- The nodes `HIGH traffic density` and `notify vehicle to reverse` and `reverse command` build and send the commands to notify the car to set its `reverse` property to `true` if it has entered a HIGH density traffic zone.

2. In the Bluemix Dashboard, go to your Node-RED app and open the Node-RED flow editor (http://*node-red-app-name*.mybluemix.net/red/). Copy the contents of the file `scenario1_flow` from the traffic simulator kit to your clipboard and then import into Node-RED using the **Import** > **Clipboard** (Ctrl-I) menu option.
3. Double-click the `trafficAlert` node and click the pencil icon beside the API Key. Assign the values of `iot_apiKey` and `iot_apiToken` from your `config/settings.js` file to API key and API token respectively. Double-click on the `speed command` and `reverse command` nodes and choose the API Key to be the one you created for the `trafficAlert` node.
4. Double-click the `Traffic rules` node, and replace *`ruleset_url`* with the URL from the Details page of the traffic ruleset deployed to the Business Rules service instance (*Step 5*). Also, check the 'Use basic authentication' box and set the Username and Password from the Connection Settings tab (*Step 3*).



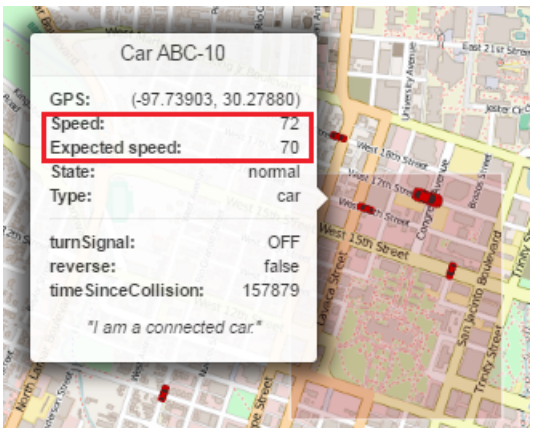
5. Double-click the `build Insights weather URL` node and replace *`insights_url`* with the URL from the Insights for Weather service credentials (*Step 3*).

Step 7. Deploy and test the Node-RED flow for Scenario 1

Deploy the new flow.

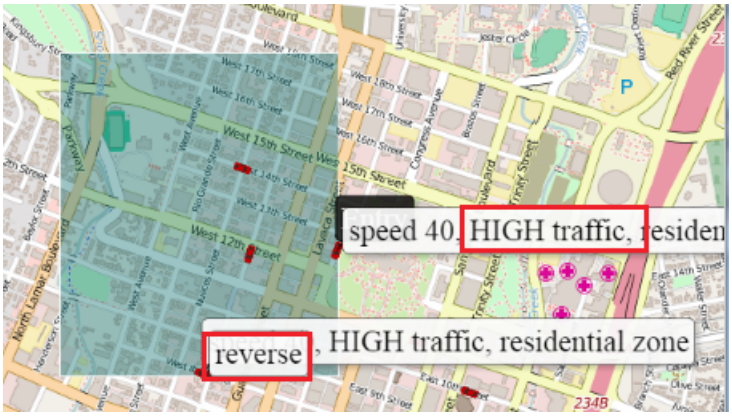
1. Activate the `trafficAlert payload` debug node by toggling the tab on its right edge. When a car enters a traffic zone in the Map app, view the Node-RED flow debug pane on the right to see the traffic alert payload. The payload shows the `eventType` ("Exit" or "Entry"), the ID of the car that entered or exited the traffic zone, and the current state of the traffic zone (including its type and the cars and accidents currently in the zone). Activate the `traffic rules input` and `traffic rules output` debug nodes. When a car enters a traffic zone in the Map app, view the Node-RED flow debug pane to see the data input into the traffic rules, and the output data of the traffic rules.

2. Deactivate the `trafficAlert payload` and `traffic rules input` debug nodes. Before a car enters a zone in the Map app, click it to note its `speed` and `expectedSpeed` properties. When the car enters a traffic zone in the Map app, see the `speed` displayed in the overlay of the car and how it corresponds to the `speedLimit` of the `trafficZone` in the `traffic rules output`.

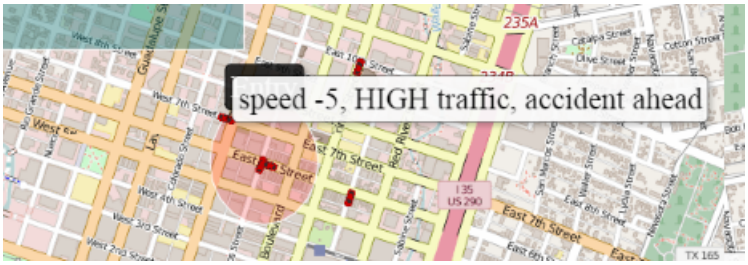


Click the car in the Map app and notice how the `expectedSpeed` property of the car has now changed to the `speedLimit` of the `trafficZone` in the traffic rules output. Also, the `speed` property of the car has been adjusted to a 10mph range of the `expectedSpeed`.

3. When the car enters a traffic zone in the Map app, see the density of the traffic zone (such as "LOW traffic") and the warning (such as "industrial zone") displayed in the overlay of the car. See how the `density` corresponds to the `trafficDensity` in the `trafficZone` of the `traffic rules output`. Similarly, notice the warning text corresponds to the `warning` element in the `traffic rules output`.
4. In the Map app, wait for a car overlay to show it is entering a HIGH traffic zone. You should see a "reverse" overlay also show above the car. Click the car and notice how the `reverse` property of the car has now changed to `true`. The car will reverse back (if possible), and the `reverse` property will change to `false`.



5. In the Map app, wait for an accident to occur and a third car to enter an Accident zone. Notice how the density is always HIGH in such a case, and the overlay shows the speed to be -5. This directs the car to reduce its `expectedSpeed` by 5mph. Also, the car is directed to reverse out of the Accident zone.



Step 8. Modify the traffic rules

You will now introduce modifications in the traffic rules, and ensure that these modified rules are executed by the Node-RED app.

1. In the Rule Designer, modify your rules in the `TrafficRules` project. In the `setTrafficZoneSpeedLimit` decision table, change the value `45` to `40` in the last row, second column, so that the speed limit for a Construction zone is set to 40.
2. Deploy the `TrafficRuleApp` to the Business Rules service instance, by selecting the **Replace RuleApp version** deployment type.
3. In the Map app, wait for a car to enter a Construction zone. The overlay should show the speed limit as `40` instead of `45`. Click the car to check the updated `expectedSpeed` (40) and `speed` properties.

   Note that you did not have to make any changes to the Node-RED app for the new rules to take effect. This is because you selected the **Replace RuleApp version** deployment type for the RuleApp.

   If you select the RuleApp deployment type **Increment the RuleApp major version** or **Increment the RuleApp minor version**, you should modify the URL in the `Traffic rules` node (*Step 5*) to point to the latest ruleset version. Deploy the latest change in the Node-RED editor to see the new version of the rules take effect.

Step 9. Design and test the collision rules for Scenario 2

Now you will design the business rules which determine the severity of an accident for Scenario 2.

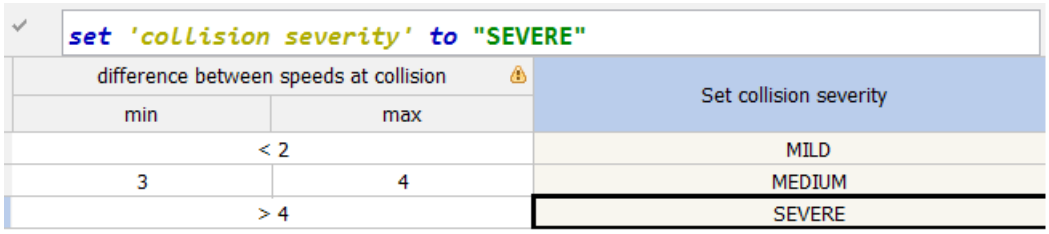
1. Define the rules in the Rule Designer. Download the `CollisionRules` and `CollisionRuleApp` projects. Import them into your Eclipse workspace and open them, then switch to the Rule perspective to open Rule Explorer.
2. Define the contract between the business logic and your client application using ruleset parameters. You can see the defined Ruleset Parameters by right-clicking the `CollisionRules` project, and choosing Properties, and then clicking Ruleset Parameters in the left pane. The input parameters consist of the speeds of the 2 vehicles at the time of collision, and the sentiment of the drivers' response (which can be “positive”, “negative”, or “neutral”). The output parameters consist of the collision severity and if medical attention is needed.

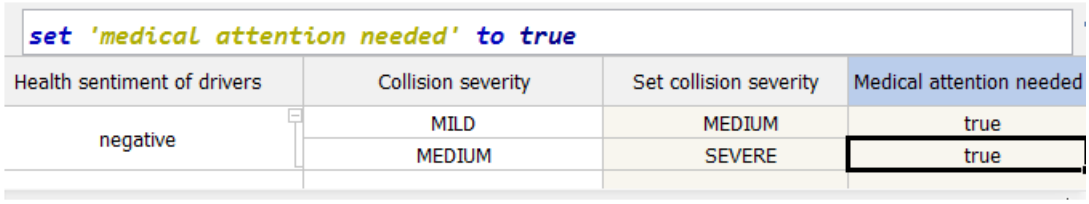
Ruleset Parameters

Define ruleset parameters.

Name	Type	Direction	Default Value	Verbalization
healthSentiment	java.lang.String	IN_OUT	"neutral"	health sentiment of drivers
vehicle1SpeedAtCollision	java.lang.Integer	IN_OUT	0	speed of vehicle1 at collision
vehicle2SpeedAtCollision	java.lang.Integer	IN_OUT	0	speed of vehicle2 at collision
collisionSeverity	java.lang.String	OUT	"MILD"	collision severity
medicalAttentionNeeded	boolean	OUT	false	medical attention needed

3. Explore the `CollisionRules` project. The `rules/collision` folder contains rules to determine the severity of an accident using the difference between the speeds of the cars, and the sentiment analysis of the drivers. The rules also determine if medical attention is needed.



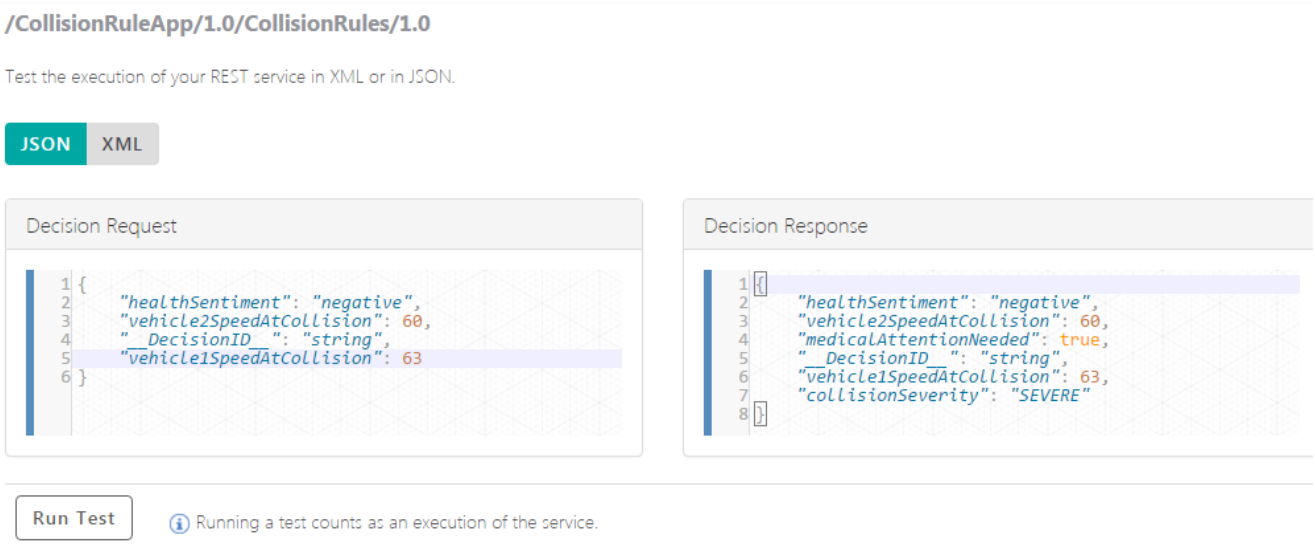



```
if
    'collision severity' is "SEVERE"
then
    set 'medical attention needed' to true;
```

4. Deploy the rules from the `CollisionRuleApp` project to the Rule Execution Server Project you created in *Step 5*.

5. In the Bluemix Dashboard, navigate to the Business Rules service instance to the **Decision Services** tab and click **Refresh** to see the newly deployed RuleApp (CollisionRuleApp 1.0).

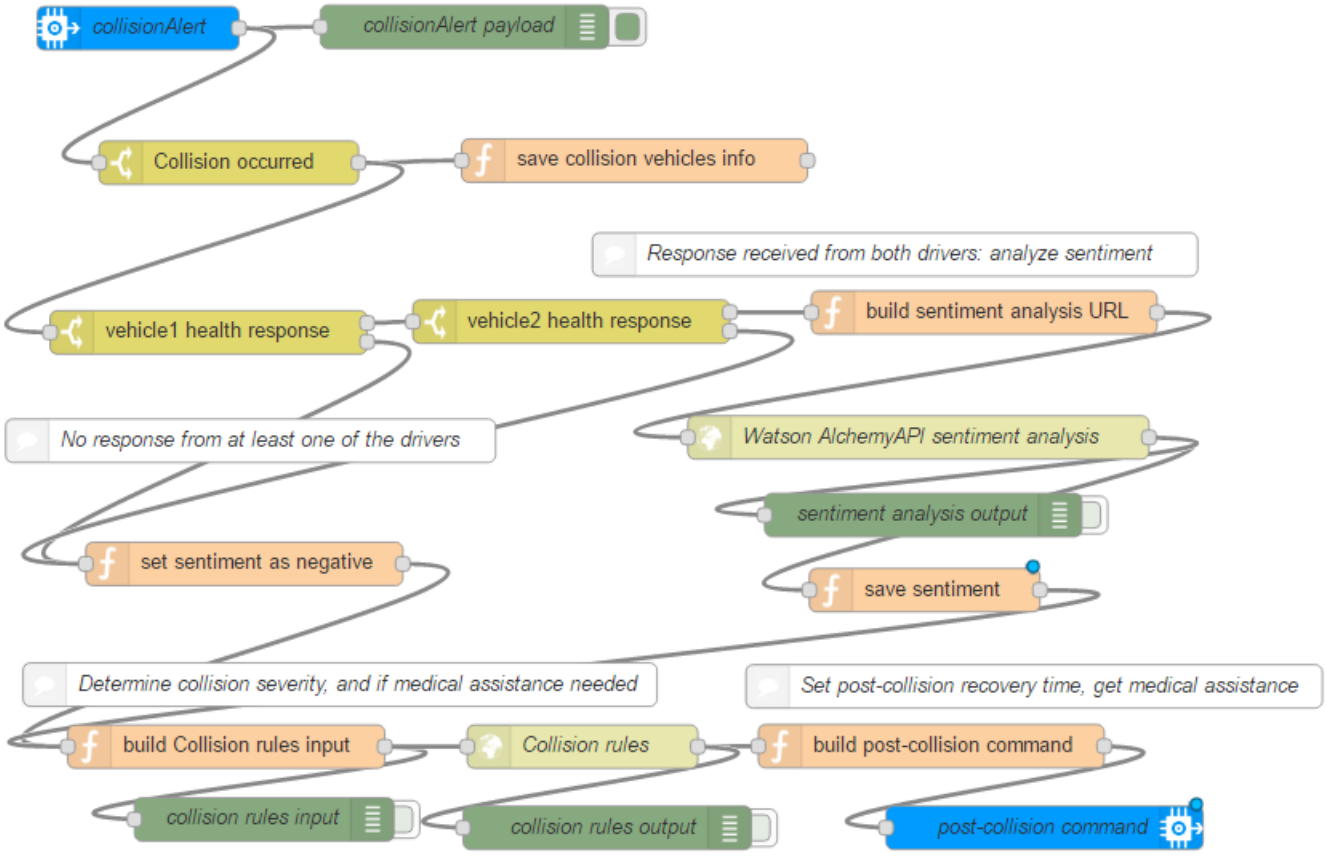
6. Click the arrow beside the ruleset (CollisionRules 1.0) to load up the **Test** page. Populate the Decision Request panel as shown in the screen capture below, and click **Run Test**.



Step 10. Create the new Node-RED flow for Scenario 2

Now you will create the Node-RED flow to determine the accident severity when a collision occurs and if medical attention is needed (Scenario 2). This flow also extends the time it takes for an accident to clear, depending on its severity.

1. Understand the new Node-RED flow. The following screen capture shows what the completed flow will look like.



- The nodes `collisionAlert`, `Collision occurred` and `save collision vehicles info` save the cars' information from the collision alert when an accident occurs.
- The nodes `vehicle1 health response`, `vehicle2 health response` and `build sentiment analysis URL` pass the combined answers from the 2 cars as an input parameter for the text sentiment analysis URL, to call the AlchemyAPI service. The nodes `Watson AlchemyAPI sentiment analysis` and `save sentiment` perform the sentiment analysis on the combined answers and retrieve the sentiment from the attribute `docSentiment.type` of the response payload. The sentiment can be “positive”, “negative”, or “neutral”. Refer to the Text API: Sentiment Analysis Bluemix documentation for details on how to use the AlchemyAPI service to extract the sentiment from text.
- If an answer is not received from either of the drivers of the two cars, it is assumed that at least one of the drivers is too hurt to answer the question. In this case, the sentiment is set to “negative” by the `set sentiment as negative` node.
- The node `build Collision rules input` builds the input parameters for the collision

rules using the sentiment and the vehicles' speed at collision. The node `Collision rules` invokes the collision rules deployed to the Business Rules service instance. The rules determine the severity of the accident and if medical attention is needed using the difference in speeds of the car and also the sentiment of the drivers' answers.

- The node `build post-collision command` builds the commands to notify the cars to set the `collisionDelay` property, which determines how long each car takes to recover from a collision. For a MILD accident, it is set to 35000 milliseconds, for a MEDIUM accident, it is set to 45000 milliseconds, and for a SEVERE accident, it is set to 60000 milliseconds.

  The node also builds the command to show an overlay on one of the cars with the severity of the accident and to call EMT, if the rules determined that medical attention is needed.

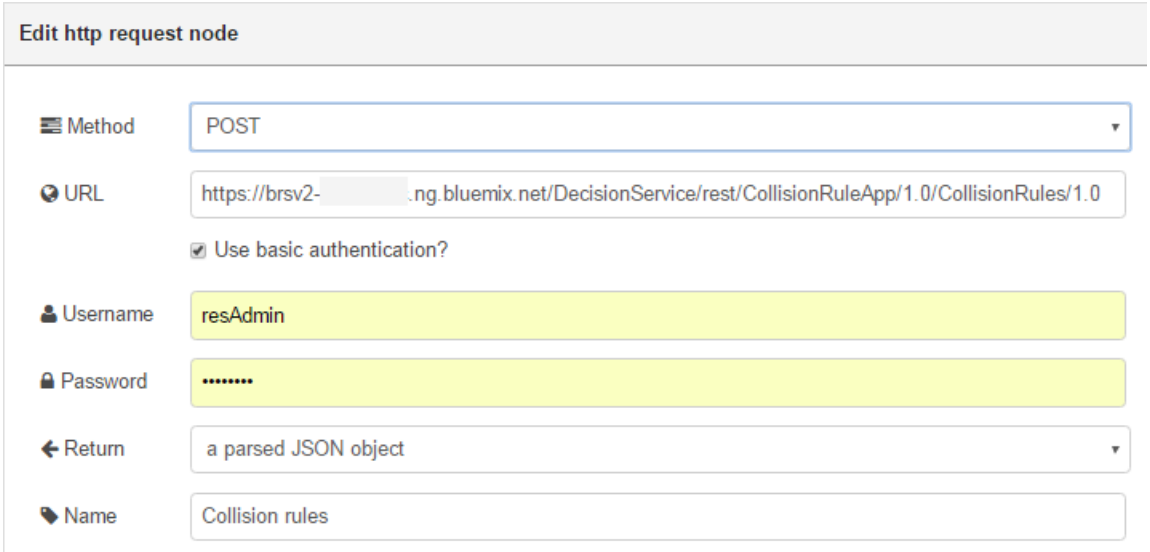
  The node `post-collision command` sends the post-collision commands to the cars.

(Missing from the figure is a part of the flow which displays overlays on the cars involved in the accident, showing the question “Are you doing alright?” and the answers from the drivers.)

2. In the Bluemix Dashboard, go to your Node-RED app and open the Node-RED flow editor. Copy the contents of the file `scenario2_flow` from the traffic simulator kit to your clipboard and then import into Node-RED using the **Import** > **Clipboard** (Ctrl-I) menu option.

3. Double-click the `collisionAlert` node and choose the API Key you created when configuring the `trafficAlert` node in *Step 6*. Repeat the same for the nodes `post-collision command`, `health request overlay`, `vehicle1 health overlay` and `vehicle2 health overlay`. (The latter 3 nodes are not shown in the figure above, but are used to display the question “Are you doing alright?” and the drivers' answers to the questions, as overlays over the cars, before the `post-collision command` displays the result of the rules as an overlay.)

4. Double-click the `Collision rules` node, and replace *`ruleset_url`* with the URL from the **Details** page of the collision ruleset deployed to the Business Rules service instance (*Step 9*). Also, check the 'Use basic authentication' box and set the Username and Password from the Connection Settings tab of the Business Rules service instance (*Step 3*).

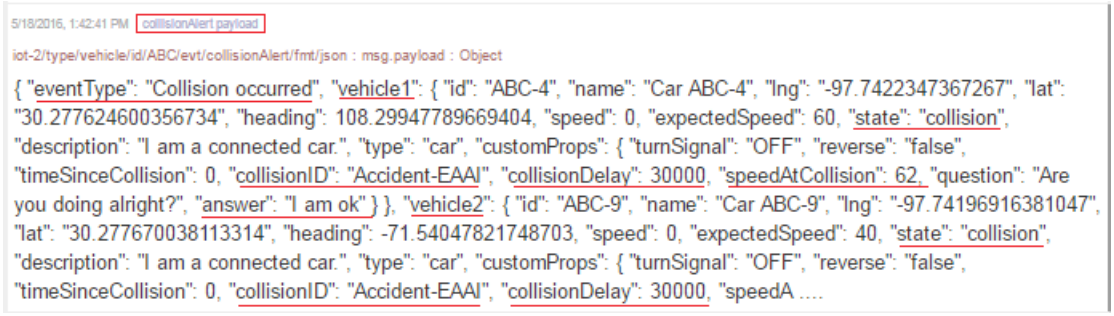


5. Double-click the `build sentiment analysis URL` node and replace *`alchemy_api_key`* with the API key you received in *Step 3* for the AlchemyAPI service.

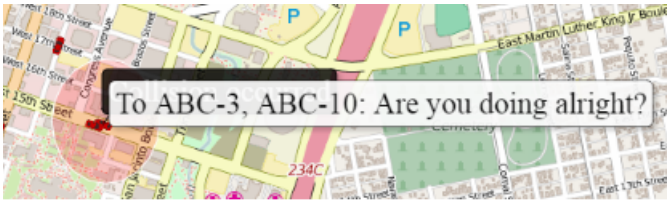
Step 11. Deploy and test the Node-RED flow for Scenario 2

Deploy the new flow.

1. In the Node-RED flow, activate the debug node `collisionAlert payload`. In the Map app, wait for the Accident zone (a circular red geofence) to be created, which shows that two cars have collided. View the Node-RED debug pane to see the collision alert payload. The collision alert payload shows the `eventType` ("Collision occurred"), and information about the two vehicles involved in the collision.



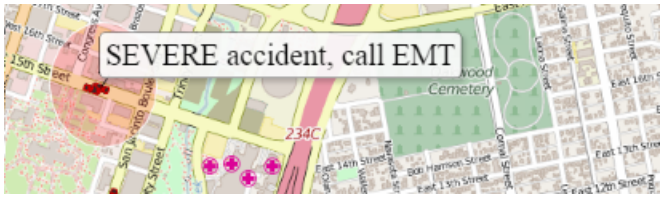
2. In the Node-RED flow, activate the debug node `sentiment analysis output`. Wait for an accident to occur in the Map app. An overlay shows the question "Are you doing alright?" and subsequent overlays show the answers to the question from both the cars involved in the accident.



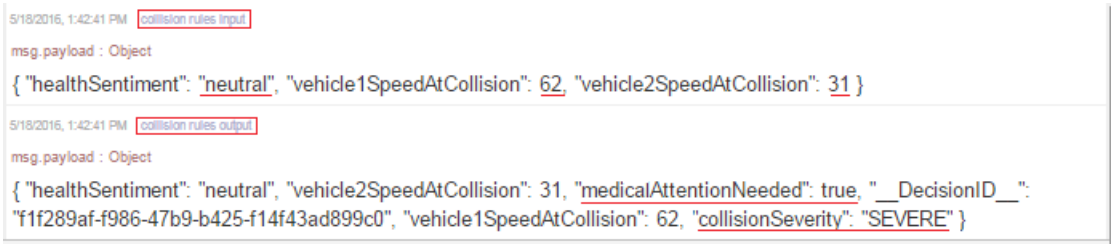
In the Node-RED debug pane, notice the sentiment extracted from the combined answers by the Watson AlchemyAPI service. For example

```
{ "status": "OK", "usage": ... , "totalTransactions": "1",
"language": "english", "text": "I am hurt. I am ok", "docSentiment":
{ "score": "-0.810976", "type": "negative" } }
```

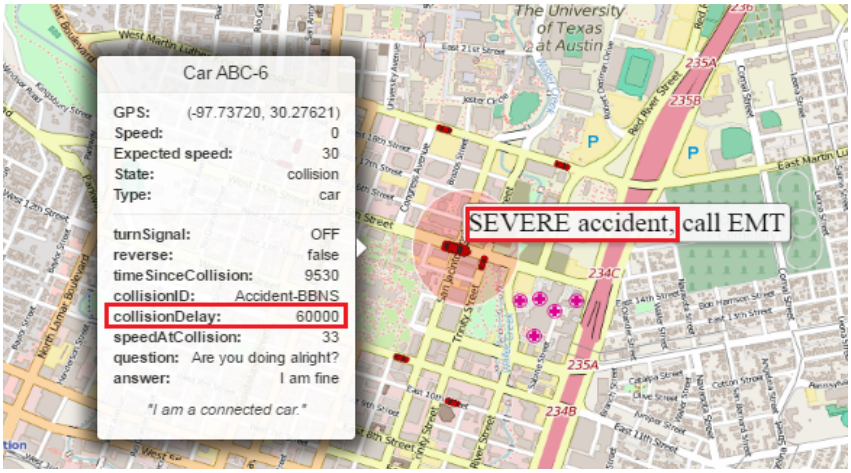
3. Activate the debug nodes `collision rules input` and `collision rules output`. Wait for an accident to occur in the Map app. After the overlays with the answers disappear, an overlay shows the severity of the accident and if medical attention is needed.



4. In the Node-RED debug pane, note the input parameters into the collision rules, and the output which shows the severity and if medical attention is needed.



5. Notice that the MEDIUM and SEVERE accidents take longer to clear than the MILD accidents. When a collision occurs, click one of the cars in the accident in the Map app, to see the value of the `collisionDelay` custom property, which determines how long a car takes to recover from a collision. For a MILD accident, it will be set to 35000 milliseconds, for a MEDIUM accident, it will be set to 45000 milliseconds, and for a SEVERE accident, it will be set to 60000 milliseconds. The car will resume driving once the value of the `timeSinceCollision` property exceeds the `collisionDelay`, and the accident will be cleared.



6. Modify the collision rules as needed in the `CollisionRules` project and deploy the RuleApp. Make sure to use the **Replace RuleApp version** deployment type to avoid having to update the Node-RED flow with the new ruleset URL.

Conclusion

In this tutorial you learnt how to introduce complex rules into an Internet of Things application, using the Business Rules service on Bluemix. You used a cognitive service (Watson AlchemyAPI) to facilitate the business rules to make decisions based on cognitive analysis. You also used current weather conditions from the Insights for Weather service as input data into business rules. Going forward, you can use the many cognitive, data and analytics services on Bluemix to expand the power of business rules within your IoT application.

You also used business rules to introduce logic changes without having to modify the application. Hence you can spend less time recoding and testing the application, when the logic can be changed and tested within the Business Rules service. This allows for greater business logic agility.

Acknowledgements

The author would like to thank Alain Robert for his recommendations for this sample, and Bryan Boyd for building and publishing his Connected Vehicle starter kit and associated tutorials.

Related Topics

- Get started with Watson IoT Platform
- Get started with Geospatial Analytics
- Get started with Business Rules
- Read IBM ODM documentation
- Get started with Insights for Weather
- Getting Started with AlchemyAPI on Bluemix