developerWorks®

Build a connected-car IoT app with Geospatial Analytics

With Geospatial Analytics and Node.js on IBM Bluemix, deploy and extend a connected-vehicle starter kit

Bryan C. Boyd (https://www.ibm.com/developerworks/community/profiles/html/profileView.do?key=a5839f3f-c1d9-42f5-8d39-68d057bfd66a&amp;lang=en&tabid=dwAboutMe)
Software Engineer, IBM
IBM

19 February 2015

Deploy and extend an Internet of Things (IoT) Connected Vehicle starter kit on Bluemix with the Internet of Things and Geospatial services. The starter kit enables you to simulate, view, and manage vehicles driving through a city and set up geofences for notification.

Want to build an Internet of Things (IoT) application on IBM Bluemix™? It's not as difficult as you might think. Most IoT apps consist of three pieces: a connected *thing*, an application to view and manage the thing, and analytics to detect events triggered by the thing. The Internet of Things service on Bluemix — IoT Foundation — makes it trivially easy to connect a thing to applications and analytics services. To demonstrate how easy it is, I built an IoT starter kit for connected cars. The Connected Vehicle kit consists of three pieces:

- A vehicle simulator (a Node.js app)
- HTML5 applications to view and manage vehicles on a map
- The Geospatial Analytics service on Bluemix, and Node-RED for analytics

Join author Bryan Boyd at IBM InterConnect 2015 for a hands-on guided tour of the IoT Foundation and Bluemix.

This tutorial guides you through configuring and deploying the Connected Vehicle starter kit on Bluemix and building analytics with the Geospatial Analytics service and Node-RED.

READ: Creating apps with the Internet of Things starter application

READ: Getting started with Geospatial Analytics

READ: Monitor mobile devices with the Geospatial Analytics service

READ: Command line interface

“ The Connected Vehicle application uses IoT Foundation for near-real-time messaging between simulated vehicles and the Map and Tester apps. ”

What you'll need to build your app

- A Bluemix account and a DevOps Services account, both linked to your IBM ID
- The Cloud Foundry command-line interface
- A basic understanding of JavaScript

Run the app
Get the code

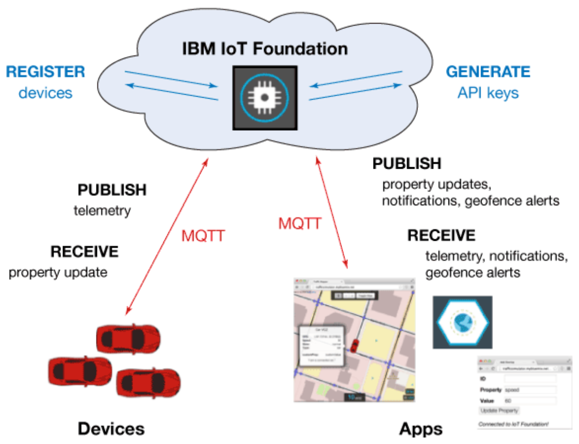
Overview of the Connected Vehicle starter kit

The Connected Vehicle application you will configure and deploy has three components:

- Node.js vehicle simulator
- HTML5 Map app
- HTML5 Tester app

These three components use the IBM Internet of Things Foundation (IoT Foundation) for real-time publish/subscribe messaging with the MQTT protocol. The simulated vehicles publish telemetry data frequently (two messages per second) and subscribe to a command topic whereby they accept commands (for example, "set speed to 60") from the tester application. The Map app subscribes to the vehicle telemetry topic to display position and status in real time. The Tester app enables you to publish control commands to the vehicles and notifications to the Map app.

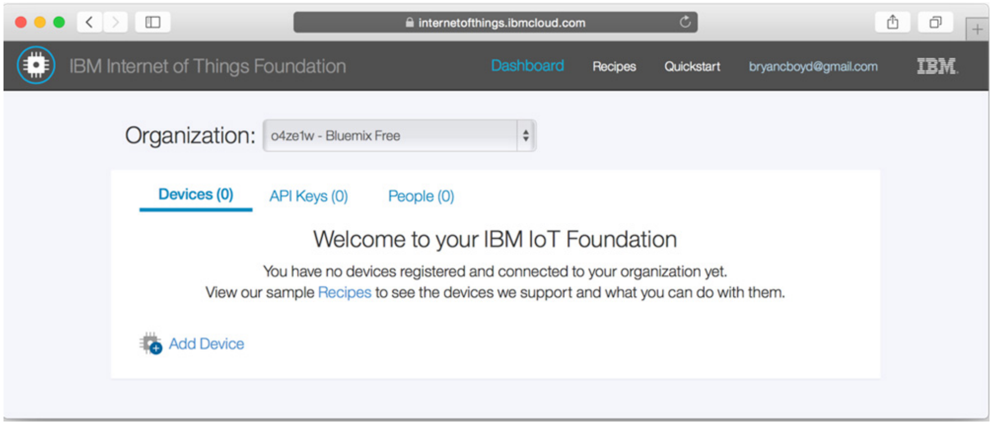
This overview diagram shows the relationships among the application's components:



Step 1. Create an Internet of Things service instance on Bluemix

1. Log in to Bluemix. In the dashboard, click **ADD A SERVICE**. Choose **Internet of Things** from the service catalog. Choose the selected plan of **Free** and click **CREATE** to create the service.

2. Select the newly created service in your dashboard, then click the **LAUNCH** button in the top-right corner of the page.
3. You are now in the IoT Foundation dashboard, where you can register devices, generate API keys for applications, view your usage, and invite others to the organization:



internetofthings.ibmcloud.com

IBM Internet of Things Foundation — Dashboard, Recipes, Quickstart, bryancboyd@gmail.com

Organization: o4ze1w - Bluemix Free

Devices (0) | API Keys (0) | People (0)

Welcome to your IBM IoT Foundation

You have no devices registered and connected to your organization yet.
View our sample Recipes to see the devices we support and what you can do with them.

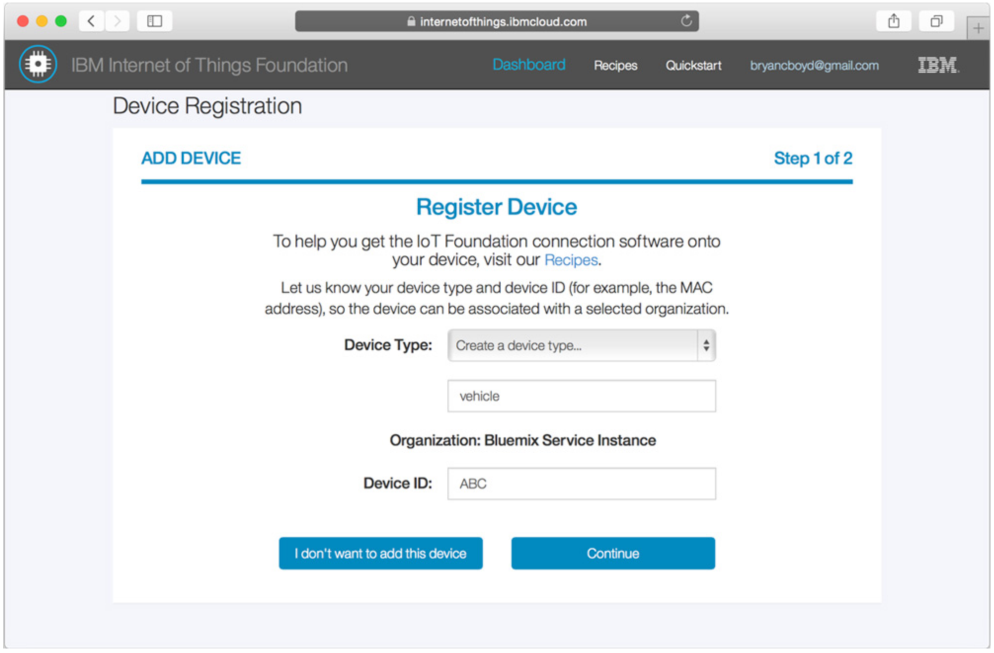
Add Device

Step 2. Configure the IoT Foundation organization

The Connected Vehicle application uses IoT Foundation for near-real-time messaging between simulated vehicles and the Map and Tester apps. To facilitate this communication, you must first register the devices and generate an API key for the applications.

With the vehicle simulator, you can model multiple vehicles from a single Node.js runtime instance. IoT Foundation treats each vehicle simulator as a device. You'll eventually run three vehicle simulators, so the first step is to register these simulators manually to obtain access credentials:

1. From the organization dashboard's Devices page, click **Add Device**. In the **Device Type** list, click **Create a device type...** and enter `vehicle` in the field that's below the **Device Type** field. Enter a device ID of any length that will be unique to this organization (for example, ABC) and



internetofthings.ibmcloud.com

IBM Internet of Things Foundation — Dashboard, Recipes, Quickstart, bryancboyd@gmail.com

Device Registration

ADD DEVICE — Step 1 of 2

Register Device

To help you get the IoT Foundation connection software onto your device, visit our Recipes.

Let us know your device type and device ID (for example, the MAC address), so the device can be associated with a selected organization.

Device Type: Create a device type...

vehicle

Organization: Bluemix Service Instance

Device ID: ABC

I don't want to add this device | Continue

click **Continue**:

2. The next page shows you the credentials for your device. Write them down! For example:
   ```
   org=o4ze1w
   type=vehicle
   id=ABC
   auth-method=token
   auth-token=5QK1rWHG9JhDw-rs+S
   ```
3. Create two more devices (for example, DEF and GHI), and write down the credentials. You'll configure the starter kit with this information.
4. From the organization dashboard, click the **API Keys** tab and select **New API Key**. Copy down the key and token. The Map and Tester apps will use these credentials to connect to IoT Foundation

Step 3. Download, configure, and deploy the starter kit

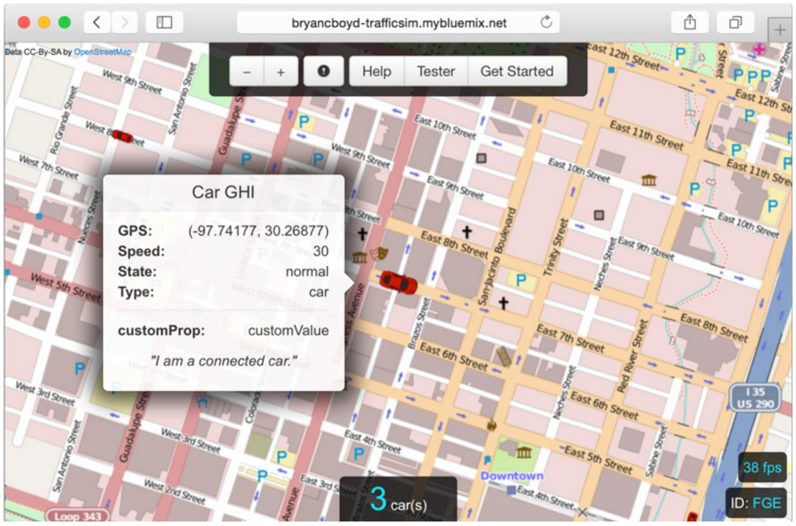
The source code for the IoT starter kit is stored in the bryancboyd | iot-vehicle-geospatial-starter-kit project on DevOps Services. You can obtain your own copy of the code in multiple ways, including either:

- Creating your own project using the **FORK PROJECT** option on the project page.
- Cloning the git repository.

If you're familiar with either method, you can obtain your own copy of the code using that method. If not, follow these instructions to create a ZIP file of the code and download it to your PC:

1. Scroll up and click this tutorial's **Get the code** button.
2. In the bryancboyd | iot-connected-vehicle-starter-kit project, click **EDIT CODE**.
3. Highlight the root of the source tree and click **File >Export > Zip** from the menu: Save the downloaded file locally.
4. Extract the ZIP file.
5. In your local project, open the manifest.yml file. Pick a globally unique application name (for example, `bryancboyd-trafficsim`) and enter the name for the **hosst** and **name** fields. Set the number of instances to `3`, and save the file.
6. Open config/settings.js. This file stores all device and API key configuration data.
   - For `iot_deviceType`, enter `vehicle`.
   - For `iot_deviceOrg`, enter your six-character organization ID (for example, `O4ze1w`).
   - For `iot_deviceSet`, enter the ID and tokens for the three devices you registered.
   - For `iot_apiKey`, enter the API key you created.
   - For `iot_apiToken`, enter the API key token.
   - Keep the default values for `notifyTopic` and `inputTopic`; these are configuration parameters used by the Geospatial Analytics service.
7. Save your changes.
8. From the root directory of your extracted application, type `cf login`. Follow the prompts, entering `https://api.ng.bluemix.net` for the API endpoint and your Bluemix IBM ID email address and password as login credentials. If you have more than one Bluemix organization and space available, you will also have to choose these.
9. Type `cf push` to deploy your application to Bluemix.

10. After your application deploys, run the Map app at http://*app-name*.mybluemix.net. You will see three simulated vehicles moving across the map. Click a vehicle to view telemetry data:



Car GHI

GPS: (-97.74177, 30.26877)
Speed: 30
State: normal
Type: car

customProp: customValue

"I am a connected car."

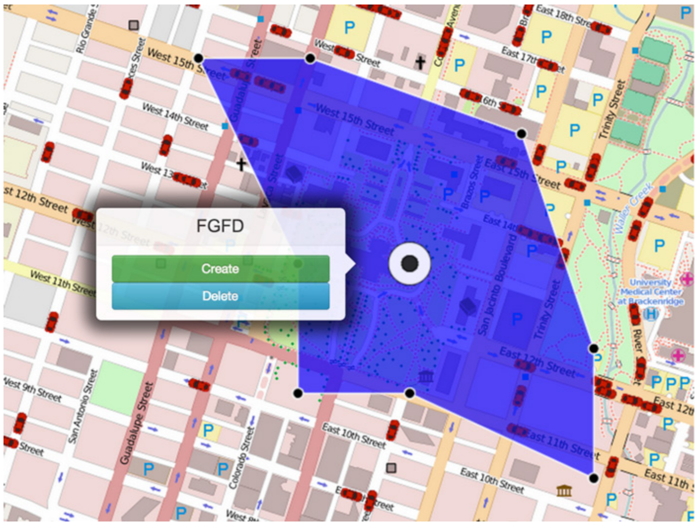
11. Each vehicle simulator can simulate multiple vehicles. The count is controlled by a Bluemix environment variable: `VEHICLE_COUNT`. Type `cf set-env <app-name> VEHICLE_COUNT 5` to increase the total number of vehicles to 15.

Step 4. Create and configure Geospatial Analytics

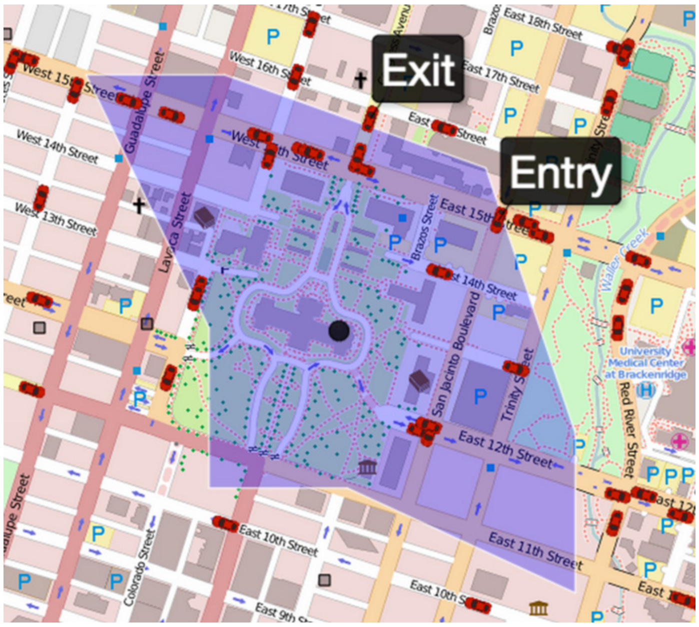
For an introduction to the Geospatial Analytics service, read "Monitor mobile devices with the Geospatial Analytics service."

Using the Bluemix Geospatial Analytics service, you can track when each vehicle enters or leaves a defined *geofence region*. The Map app can interact with the service, enabling you to graphically create and delete geofences dynamically:

1. In the Bluemix dashboard for your new application, click **ADD A SERVICE**. Choose **Geospatial Analytics** from the service catalog. Choose the selected plan of **Free** and click **CREATE** to bind the service to your application.
2. The starter-kit application includes APIs to start and stop Geospatial Analytics with the configured IoT Foundation organization and messaging credentials. To start the service, browse to http://*app-name*.mybluemix.net/GeospatialService_start and wait for the page to show success (which might take 30-45 seconds).
3. Open the Map app and create some geofences. Click the **Alert** button (the exclamation-point icon) on the toolbar to start the geofence creation process, and drag the handles and edges to

position the geofence. To create the geofence, click the center black circle to open the context menu, and then click **Create**: 

The Geospatial Analytics service adds this geofence to the list of watched regions and publishes an MQTT message through IoT Foundation whenever a vehicle enters or leaves the area. The Map app consumes the message and displays an overlay over the vehicle on the map:

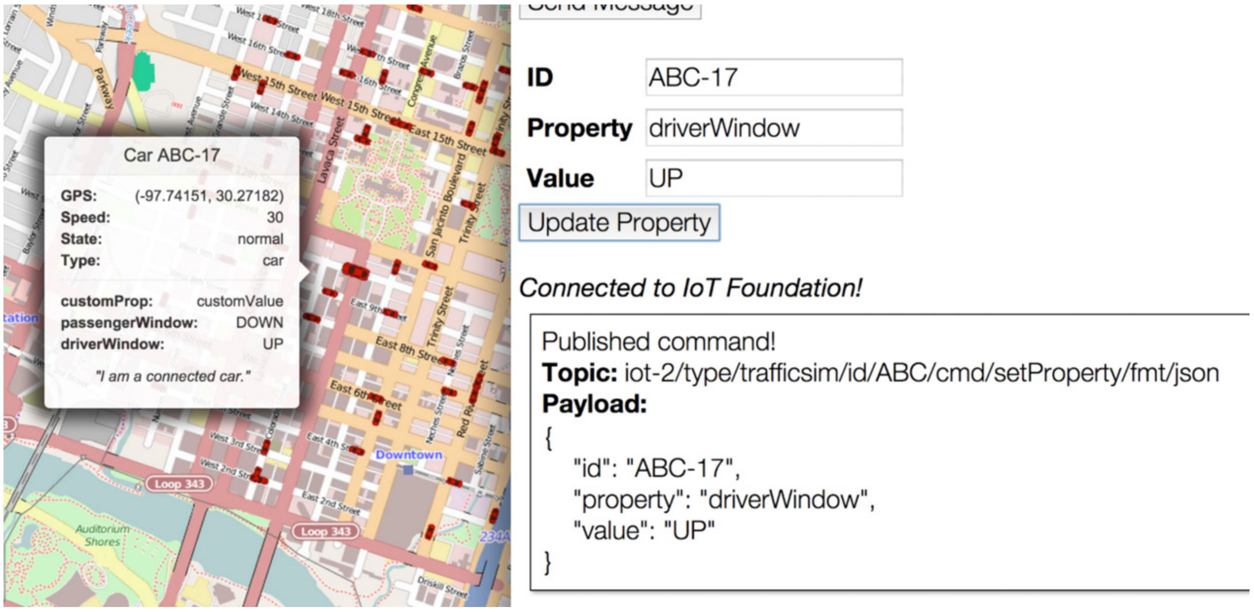


To delete a geofence, select the region and click **Delete**.

Step 5. Use the Tester app

The Tester app sends commands to the simulated vehicles and the Map app. The vehicle simulator subscribes to commands of type `setProperty` and dynamically changes its own properties, speed, and state. The Map app subscribes to commands of type `addOverlay` and dynamically displays a pop-up of text over a vehicle.

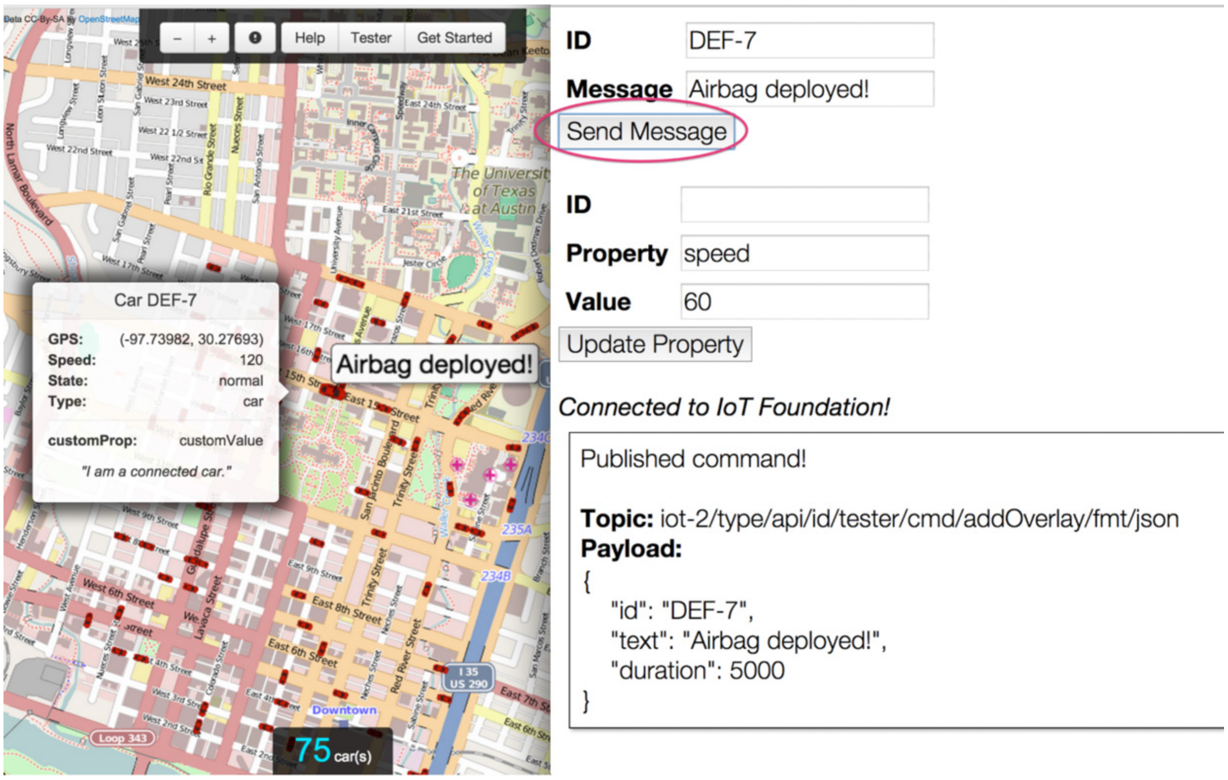
1. Open the Map app and Tester app (http://*app-name*.mybluemix.net/tester) side-by-side, so that you can see both pages.
2. To configure the `setProperty` command, select a vehicle and enter the ID in the bottom form on the Tester app. Enter a property of `speed` and a value of `100`, and click **Update Property**. You will see the MQTT topic and payload containing the command in the Tester app, and the selected vehicle will change speed to 100 mph.
   - **Note:** The vehicles simulate a set of **static properties** (location, speed, state, and type) and **custom properties**. You can use the `setProperty` API to dynamically add/change/delete a custom property on a vehicle. To add a property, publish a property that doesn't yet exist (for example, `DriverWindow = UP`). To delete a property, update the property with a value of an empty string; the vehicle will stop including the property in its telemetry message:



Connected to IoT Foundation!

Published command!
Topic: iot-2/type/trafficsim/id/ABC/cmd/setProperty/fmt/json
Payload:
```
{
    "id": "ABC-17",
    "property": "driverWindow",
    "value": "UP"
}
```

3. To configure the `addOverlay` command: In the Tester app, use the upper form to display a message over a vehicle:



Car DEF-7

GPS:	(-97.73982, 30.27693)
Speed:	120
State:	normal
Type:	car
customProp:	customValue

"I am a connected car."

ID	DEF-7
Message	Airbag deployed!

Send Message

ID	
Property	speed
Value	60

Update Property

Connected to IoT Foundation!

Published command!

Topic: iot-2/type/api/id/tester/cmd/addOverlay/fmt/json
Payload:

```
{
    "id": "DEF-7",
    "text": "Airbag deployed!",
    "duration": 5000
}
```

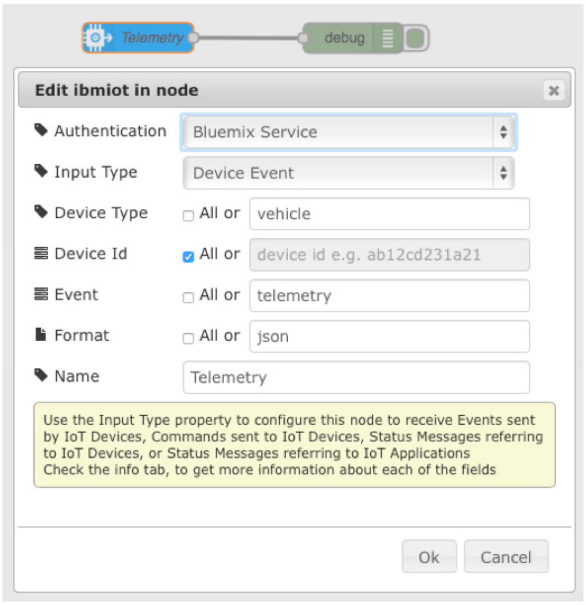
Step 6. Create a Node-RED analytics application

You can use the Node-RED boilerplate in Bluemix to extend the starter kit with basic analytics. Node-RED is a visual tool for wiring together events in IoT use cases. With Node-RED, you can design flows of logic that link a variety of inputs (such as MQTT, TCP, and HTTP) to outputs (such as MQTT, HTTP, Twitter, MongoDB, and Cloudant) to add lightweight analytics quickly:

1. Deploy the Node-RED boilerplate application in Bluemix: In your Bluemix dashboard, click **ADD APPLICATION** and choose the Internet of Things Foundation boilerplate. Provide a unique name and hostname (for example, `bryancboyd-nodered`) and click **CREATE**.
2. When the application is running, click **BIND A SERVICE** and chose the Internet of Things service you created in Step 1.
3. After the application stages, open the Node-RED canvas at http://*node-red-app*.mybluemix.net/red/.
4. The IoT boilerplate Node-RED application starts with a default flow. Select the nodes on the canvas and delete them.
5. Build a flow to monitor telemetry for all vehicles: Drag an **ibmiot** input node from the palette on the left onto the canvas. Drag a **debug** node to the right of the ibmiot node and wire the two nodes together. Double-click the **ibmiot** node and configure it as follows:
    - For **Authentication**, select **Bluemix Service**.
    - For **Input Type**, select **Device Event**.
    - For **Device Type**, enter `vehicle`.
    - For **Device Id**, select the **All** check box.

- For **Event**, enter `telemetry`.
- For **Format**, enter `json`.
- For **Name**, enter `telemetry`.

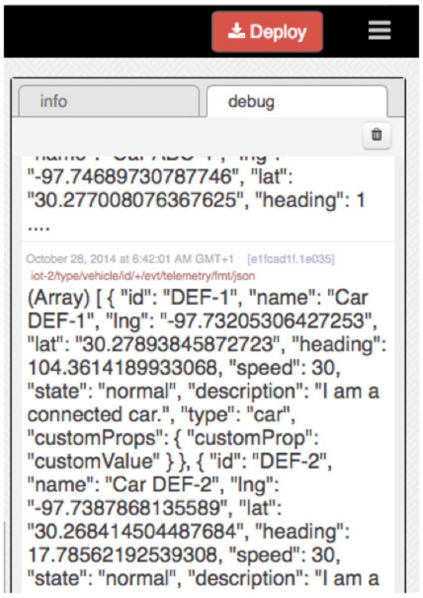
6. Click **Deploy** (top-right corner of page) and select the **debug** pane. If configuration was successful and the starter kit application is currently running, you will see the MQTT payloads for the connected vehicles in the debug pane:



Edit ibmiot in node

Authentication		Bluemix Service
Input Type		Device Event
Device Type	☐ All or	vehicle
Device Id	☑ All or	device id e.g. ab12cd231a21
Event	☐ All or	telemetry
Format	☐ All or	json
Name		Telemetry

Use the Input Type property to configure this node to receive Events sent by IoT Devices, Commands sent to IoT Devices, Status Messages referring to IoT Devices, or Status Messages referring to IoT Applications
Check the info tab, to get more information about each of the fields



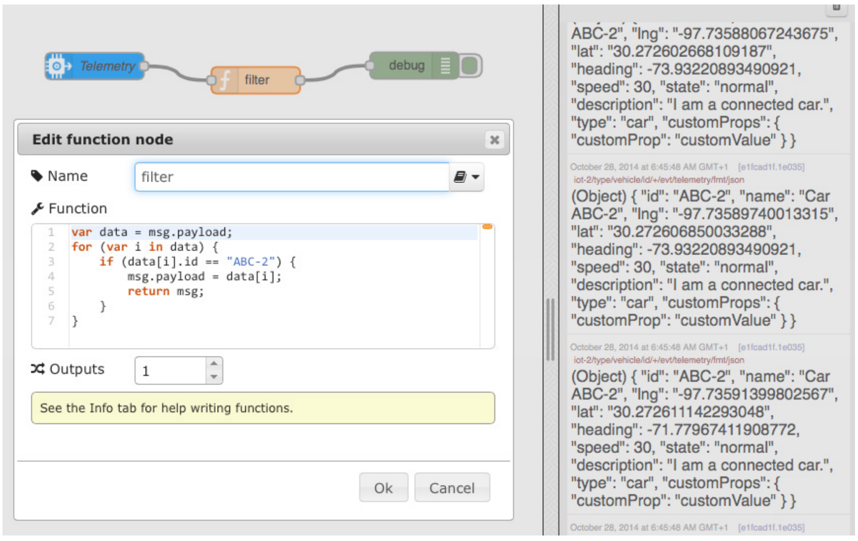
```
"-97.74689730787746", "lat":
"30.277008076367625", "heading": 1
....

October 28, 2014 at 6:42:01 AM GMT+1 [e1fcad1f.1e035]
iot-2/type/vehicle/id/+/evt/telemetry/fmt/json
(Array) [ { "id": "DEF-1", "name": "Car
DEF-1", "lng": "-97.73205306427253",
"lat": "30.27893845872723", "heading":
104.3614189933068, "speed": 30,
"state": "normal", "description": "I am a
connected car.", "type": "car",
"customProps": { "customProp":
"customValue" } }, { "id": "DEF-2",
"name": "Car DEF-2", "lng":
"-97.7387868135589", "lat":
"30.268414504487684", "heading":
17.78562192539308, "speed": 30,
"state": "normal", "description": "I am a
```

7. If your `VEHICLE_COUNT` environment variable is set to a value greater than `1`, the telemetry data for `VEHICLE_COUNT` vehicles is contained in a single array. To modify the flow to filter for a single vehicle (for example, ABC-2), add a **function** node in the middle of your flow, double-click, and fill in the following code:

```
var data = msg.payload;
for (var i in data) {
  if (data[i].id == "ABC-2") {
    msg.payload = data[i];
       return msg;
  }
}
```

8. Wire the flow together and click **Deploy**. You should now see data for a single vehicle in your debug pane:



Name: filter

```
var data = msg.payload;
for (var i in data) {
    if (data[i].id == "ABC-2") {
        msg.payload = data[i];
        return msg;
    }
}
```

Outputs: 1

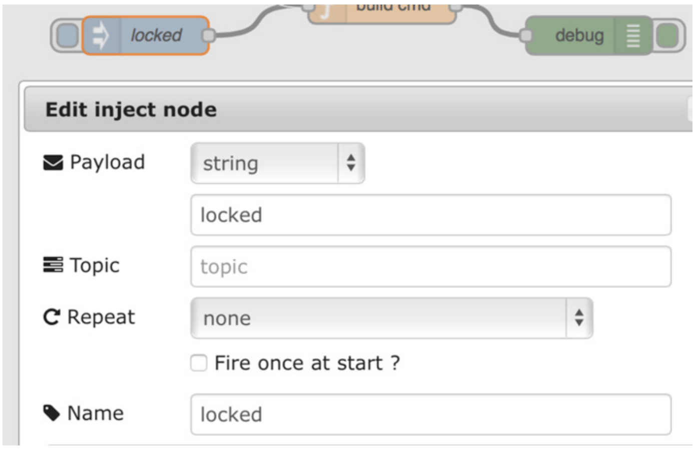
See the Info tab for help writing functions.

```
ABC-2", "lng": "-97.73588067243675",
"lat": "30.272602668109187",
"heading": -73.93220893490921,
"speed": 30, "state": "normal",
"description": "I am a connected car.",
"type": "car", "customProps": {
"customProp": "customValue" } }
October 28, 2014 at 6:45:48 AM GMT+1   [e1fcad1f.1e035]
iot-2/type/vehicle/id/+/evt/telemetry/fmt/json
(Object) { "id": "ABC-2", "name": "Car
ABC-2", "lng": "-97.73589740013315",
"lat": "30.272606850033288",
"heading": -73.93220893490921,
"speed": 30, "state": "normal",
"description": "I am a connected car.",
"type": "car", "customProps": {
"customProp": "customValue" } }
October 28, 2014 at 6:45:48 AM GMT+1   [e1fcad1f.1e035]
iot-2/type/vehicle/id/+/evt/telemetry/fmt/json
(Object) { "id": "ABC-2", "name": "Car
ABC-2", "lng": "-97.73591399802567",
"lat": "30.272611142293048",
"heading": -71.77967411908772,
"speed": 30, "state": "normal",
"description": "I am a connected car.",
"type": "car", "customProps": {
"customProp": "customValue" } }
October 28, 2014 at 6:45:48 AM GMT+1   [e1fcad1f.1e035]
```

Step 7. Create a virtual key fob in Node-RED

Next, use Node-RED to build a simple remote control to lock/unlock a vehicle. You will use two **inject** nodes: one to unlock the vehicle, the other to lock it. For each operation, you'll publish a `setProperty` command for the `lockState` property with a value of `locked` or `unlocked`, then publish an `addOverlay` command to display a notification on the Map app:

1. Go to your Node-RED flow and drag two **inject** input nodes onto the Node-RED canvas. Edit each node and configure a string payload: `locked` for one, `unlocked` for the other. This payload will be the value that is sent in the `setProperty` command:

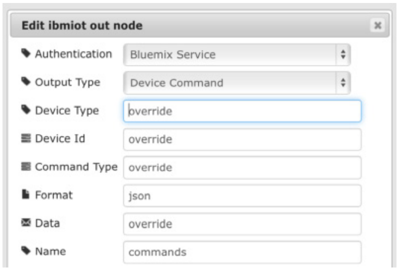


Payload: string
locked
Topic: topic
Repeat: none
☐ Fire once at start ?
Name: locked

2. Add a function node to the flow and wire both inject nodes to it. This function will build and publish two commands to IoT Foundation: one for setting the vehicle state, the other for displaying an overlay message on the Map app. Use the following code:

```
var commandValue = msg.payload;
var setPropertyMsg = {
        deviceId: "ABC",
        deviceType: "vehicle",
        eventOrCommandType: "setProperty",
        format: "json",
        payload: JSON.stringify({
               "id": "ABC-2",
                "property": "lockState",
                "value": commandValue
        })
}
var addOverlayMsg = {
        deviceId: "tester",
        deviceType: "api",
        eventOrCommandType: "addOverlay",
        format: "json",
        payload: JSON.stringify({
                "id": "ABC-2",
                "text": commandValue,
                "duration": 5000
        })
}
return [ [setPropertyMsg, addOverlayMsg] ];
```

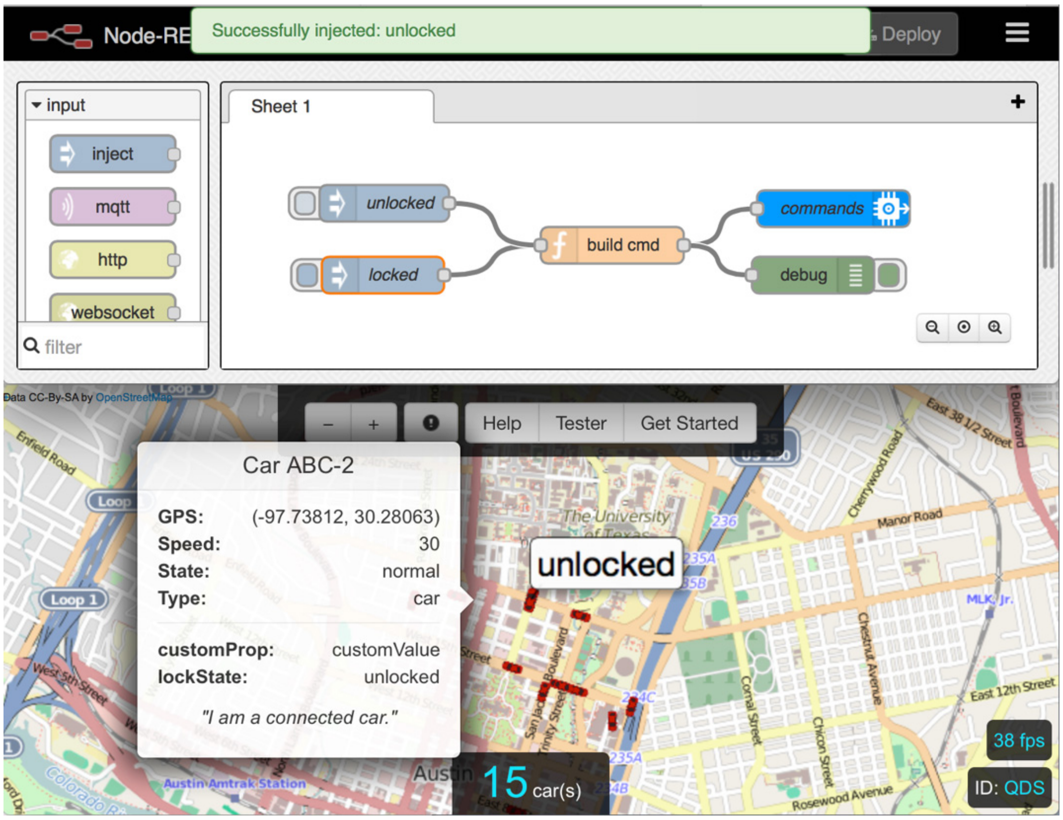
3. Add an **ibmiot** output node to the flow, after the function node. The function node will overwrite many of the configuration values on the ibmiot node; these fields we will mark with a "dummy" value of `override`. Double-click the ibmiot node and configure it as follows:
    - For **Authentication**, select **Bluemix Service**.
    - For **Input Type**, select **Device Command**.
    - For **Device Type**, enter `override`.
    - For **Device Id**, enter `override`.
    - For **Command Type**, enter `override`.
    - For **Format**, enter `json`.
    - For **Data**, enter `override`.
    - For **Name**, enter `commands`.


Edit ibmiot out node

Authentication	Bluemix Service
Output Type	Device Command
Device Type	override
Device Id	override
Command Type	override
Format	json
Data	override
Name	commands

4. Deploy the application (click **Deploy** in the top-right corner of the page) and click the button on the inject nodes while watching the Map app. You can see the `lockState`

property update for vehicle ABC-2, and an "unlocked" message appears on the map:



Car ABC-2

GPS: (-97.73812, 30.28063)
Speed: 30
State: normal
Type: car

customProp: customValue
lockState: unlocked

"I am a connected car."

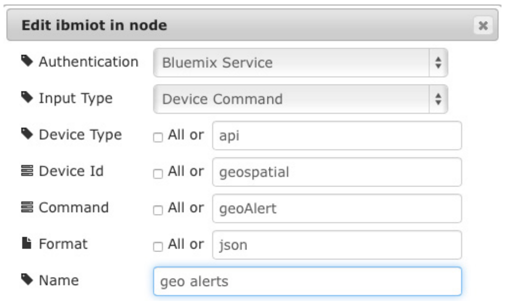
Step 8. Tweet geospatial events with Node-RED

With the Bluemix Geospatial Analytics service, you can create geofence polygons in the Map app and receive IoT Foundation events when a vehicle enters or leaves a region. The Map app already subscribes to this event topic and displays messages as a notification overlay over the vehicle. This geospatial event topic can be subscribed to by multiple applications, and it can be a building block used to create more-complex scenarios — for example, a bounding box for young drivers, signaling parents whenever their child drives outside of an "approved" zone. Or, detect when your vehicle has entered a geofence surrounding your house and notify your house automation system to prepare for your arrival.

You can use Node-RED to link geofence alerts to other applications and APIs. In this flow, you'll publish a tweet when receiving a geofence notification message for a particular vehicle:

1. Drop an **ibmiot** input node onto the canvas. Double-click the **ibmiot** node and configure it as follows:
    - For **Authentication**, select **Bluemix Service**.
    - For **Input Type**, select **Device Command**.
    - For **Device Type**, enter `api`.
    - For **Device Id**, enter `geospatial`.
    - For **Command Type**, enter `geoAlert`.
    - For **Format**, enter `json`.

- For **Name**, enter `commands`.



Edit ibmiot in node

Authentication	Bluemix Service
Input Type	Device Command
Device Type	☐ All or api
Device Id	☐ All or geospatial
Command	☐ All or geoAlert
Format	☐ All or json
Name	geo alerts

2. Add a function node after the ibmiot input node. This function will filter the data returned from the geospatial service and continue flow execution only if the vehicle for the alert matches an ID specified. Use the following code (if desired, replace the ID with a vehicle in your environment), wire the function to a debug node, and deploy:

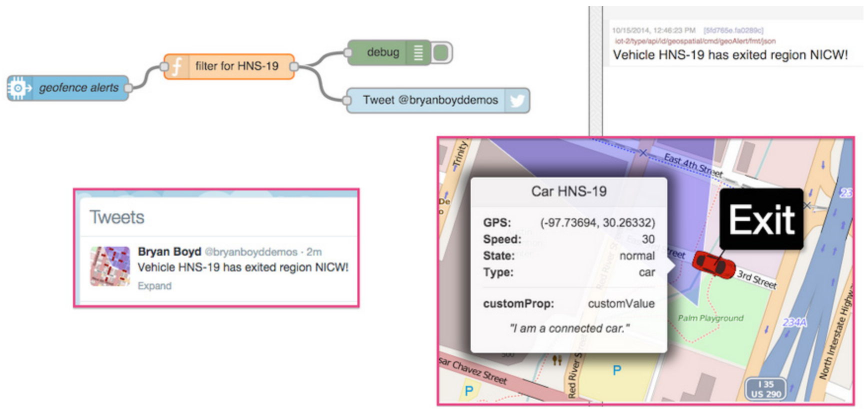
```
msg.payload = {
   time: msg.payload.time,
   id: msg.payload.deviceInfo.id,
   lon: msg.payload.deviceInfo.location.longitude,
   lat: msg.payload.deviceInfo.location.latitude,
   eventType: msg.payload.eventType
   region: msg.payload.regionId
       }
       if (msg.payload.id == "ABC-3") { return msg; }
```

3. Next, drag a **twitter** output node onto the canvas, and wire the output of the function to the input of this node. You can use the twitter node to configure an existing Twitter account using OAuth and then tweet the contents of `msg.payload`. Sign in to your Twitter account and modify the function node to build the message for Twitter:

```
msg.payload = {
   time: msg.payload.time,
   id: msg.payload.deviceInfo.id,
   lon: msg.payload.deviceInfo.location.longitude,
   lat: msg.payload.deviceInfo.location.latitude,
   eventType: msg.payload.eventType,
   region: msg.payload.regionId
       }
       if (msg.payload.id == "ABC-3") {
    var verb = "exited";
    if (msg.payload.eventType == "Entry") { verb = "entered"; }
    msg.payload = "Vehicle " + msg.payload.id + " has " + verb +
       " region " + msg.payload.region + "!";
    return msg;
       }
```

4. Deploy the application.

Now Node-RED will publish a tweet whenever a geofence is crossed by your vehicle:



Vehicle HNS-19 has exited region NICW!

Tweets

Bryan Boyd @bryanboyddemos · 2m
Vehicle HNS-19 has exited region NICW!

GPS:	(-97.73694, 30.26332)
Speed:	30
State:	normal
Type:	car
customProp:	customValue

"I am a connected car."

Conclusion

Building an IoT application with the Internet of Things and Geospatial Analytics services in Bluemix is simple and easy. Try it for yourself: Extend the Connected Vehicle starter kit with other Bluemix services, mobile applications, even physical sensors.

Geospatial Analyticshttps://console.ng.bluemix.net/#/store/serviceOfferingGuid=f5c45150-8023-4d3e-a3d4-5a3a8ca8e407&fromCatalog=trueenables your applications to track when devices enter or leave defined regions.The Internet of Things servicehttp://www.ibm.com/developerworks/topics/internet%20of%20things%20serviceprovides simple but powerful application access to IoT devices and data.The Node-RED starterhttp://www.ibm.com/developerworks/topics/internet%20of%20things%20servicedemonstrates how to run the Node-RED open-source project within IBM Bluemix.

RELATED TOPICS: Internet of Things Node.js Big data and analytics

About the author

Bryan C. Boyd

@bryanboyd on Twitter