Contents

[Synopsis 1](#_Toc521338659)

[Prerequisite: 1](#_Toc521338660)

[Step 1: Build a Queue 1](#_Toc521338661)

[Step 2: Build a Cosmos DB Collection 2](#_Toc521338662)

[Projects 2](#_Toc521338663)

[Step 3: IoTEmulator 2](#_Toc521338664)

[Step 4: AVLS 2](#_Toc521338665)

[Step 5: AzureFuncition/MonitorTraffic 2](#_Toc521338666)

[Step 6: Run AVLS project 3](#_Toc521338667)

[Step 7: Run MonitorTraffic Azure Function. 3](#_Toc521338668)

[Step 8: Run IoT Emulator to insert a fake traffic accident data in Cosmos DB. 3](#_Toc521338669)

# Synopsis

This demo shows the working of Azure Cosmos DB collection Changefeed functionality. In this demo, we are showing a control center, which is monitoring all the cars on road, and as soon as any accident happens, it shows on the control room screen. Once the location of accident is known, an ambulance or fire truck can be dispatched.

There are three part of this demo, an IoT emulator, which send data to Cosmos DB. MonitorTraffic azure function is listening on the changefeed of Colleciton, and If it finds an accident, then it update an accident queue on the storage.

There is a UI, which is listening to the queue and as soon as it finds any data on the queue, it updates the UI with right lat and long.

# Prerequisite:

## Step 1: Build a Queue

On the portal, make sure you have a storage account: scmrstorage. Create trafficqueue queue in the storage. Please note the Connection string for storage and queue.

## Step 2: Build a Cosmos DB Collection

Create a Cosmos DB Account.

Create a Database called: IoT and a collection: IoT.

Note the collection name and keys. Use will use this information in updating the IoTEmulator, App.config file.

# Projects

There are three projects for this Demo:

## Step 3: IoTEmulator

has the project, which insert the data in Collection. Update the App.config with right connection string from step 2.

## Step 4: AVLS

Project is the UI project. This shows the maps and the car current position. Update the app.config with right storage and queue connection strings from step 1.

## Step 5: AzureFuncition/MonitorTraffic

Project has the Azure function code. Update the config file with the right connection string of storage and queue from step 1.

After opening the AVLS project

Update the all the Git packages

Install typescript globally

npm install -g typescript

Install BingMap

Install-Package Microsoft.BingMaps.V8.TypeScript

Compile

Step 6: Run AVLS project

Referesh the screen so you see the initial map.

Step 7: Run MonitorTraffic Azure Function.

## Step 8: Run IoT Emulator to insert a fake traffic accident data in Cosmos DB.

