TRIPTHRU SIMULATOR

This document covers the use and configuration of the TripThru simulator

Follow the README1.md document for information regarding installation on AWS.
It explains how to install the TripThruSimulator on an AWS free-tier Ubuntu server.

General format for running the simulator:

% node start <service_js_file>

To run all service_files:

% node start all

The installation comes with one service file called fabless.js

To run it

% node start fabless.js

Sample output:

```
2015-04-09T02:53:15.997Z - init : Loading configuration fabless.js
2015-04-09T02:53:15.998Z - init : Creating network fabless.jsfrom configuration...
2015-04-09T02:53:15.999Z - init : Opening socket client fabless.js...
2015-04-09T02:53:16.009Z - init : Starting express...
server listening on port 3302
2015-04-09T02:53:16.071Z - init : Successfully connected to:
mongodb://localhost:27017/partners
2015-04-09T02:53:16.078Z - init : Connected to http://107.170.220.160/
2015-04-09T02:53:16.078Z - init : Socket open, starting simulation fabless.js...
2015-04-09T02:53:16.234Z - fabless@tripthru.com : Emitted set-network-info
fabless@tripthru.com, res: 200
2015-04-09T02:53:24.135Z - 110267@mytaxiberlin@tripthru.com : fabless@tripthru.com
received getQuote
2015-04-09T02:53:41.240Z - sim : Donald requests to be picked up at 37.771561, -
122.442372 on 2015-04-09T02:58:41+00:00 and dropped off at 37.771332, -122.435345
2015-04-09T02:53:41.241Z - sim : Queueing FablessNewYork-1@fablessnewyork@tripthru.com
2015-04-09T02:53:41.241Z - FablessNewYork-1@fablessnewyork@tripthru.com : Status has
changed from new to queued. Driver location:
2015-04-09T02:53:41.242Z - FablessNewYork-1@fablessnewyork@tripthru.com : Ready to
dispatch
2015-04-09T02:53:41.242Z - FablessNewYork-1@fablessnewyork@tripthru.com : Dispatch
locally
2015-04-09T02:53:41.242Z - FablessNewYork-1@fablessnewyork@tripthru.com : Pickup
location 37.771561,-122.442372 is outside of coverage area
2015-04-09T02:53:41.336Z - fabless@tripthru.com : Emitted dispatch-trip
1@fablessnewyork@tripthru.com, res: 500
```

You can view the activity on the TripThru web page at:

http://www.tripthru.com/signin

(use the credentials that should have been supplied to you to log in)

Setting Changes

You can run the simulator with different settings. There are four files that can be adjusted to change the behavior of the simulator. This table shows the file name and what aspects of the simulator they cover:

Network_js_file	Contains service parameters for the specific simulated client
config.js	Contains global configurations
Cities.js	Cities to be used in this simulation
tripsByCity.js	Array of trips that can be simulated in each city

These files will be discussed below along with a configuration explanation/example.

Configuring config.js

This file manages the global app settings like your simulation's port and url.

expressPort	If you wish to use a restful endpoint this is the port that will be opened.
expressUrl	If you wish to use a restful endpoint you must set your public url here so TripThru can reach your simulator.
tripthru url	TripThru's API url
tripthru cert	Path to the provided .crt file
tripthru key	Path to the provided .key file
tripthru passphrase	Ssl passphrase provided by TripThru
db	Configuration parameters for MongoDB

Configuring the network file

This file manages the general aspects of the simulation configuration, such as the simulator company name and the cities that are serviced.

Here is a copy of the provided network_config/fabless.js file

```
module.exports = {
  tripthru: {
    token: 'RjNtkcfbMIPvyCtquoXElKwQyELxcoDzBhHtnXOzgSekmFZXFc'
  },
  name: 'Fabless',
  clientId: 'fabless@tripthru.com',
  simulationInterval: 10,
```

```
  endpointType: 'socket',
  tripsPerHour: 100,
  drivers: 10,
  currencyCode: 'USD',
  capacity: 200,
  imageUrl:
'http://www.tripthru.com/assets/networks/fabless@tripthru.com.png
',
  acceptsPrescheduled: true,
```

```
  acceptsOndemand: true,
  acceptsCashPayment: true,
  acceptsAccountPayment: true,
  acceptsCreditcardPayment: true,
  coverage: [
       {city: 'Los Angeles', businessPercentage: 50},
       {city: 'New York', businessPercentage: 50}
   ]
};
```

Each field and explanation:

tripthru token	Provided by TripThru upon registration
name	this field will show up as the service name, should match the filename
clientId	Provided by TripThru upon registration
simulationInterval	number of seconds before next request
endpointType	'socket' or 'restful', if chosen endpoint is restful you need to add your public url to config.js (see above).
tripsPerHour	number of trips the simulate per hour
drivers	number of available drivers
currencyCode	do not adjust this field
capacity	Number of connections
imageUrl	image url for your company
acceptsPrescheduled	true/false - are prescheduled requests allowed
acceptsOndemand	true/false - users may request on the fly
acceptsCashPayment	true/false - users can pay with cash
acceptsAccountPayment	true/false - users can pay with account
acceptsCreditcardPayment	true/false - users can pay with credit card
coverage	List of cites covered in the simulator, NOTE these cities must exist in the cities.js file (the cities.js file is discussed below)

If you want to test out a new set of configurations – make a copy of the existing Service_js_file as a starting point for your configuration.

% cp network_config/fabless.js network_config/mysimcompany.js

Edit your copy of the file network_config/mysimcompany.js

Here is a new version of the file adjusted for mysimcompany.js

```
module.exports = {
```

```
  tripthru: {
    token: 'RjNtkcfbMIPvyCtquoXElKwQyELxcoDzBhHtnXOzgSekmFZXFc'
  },
  name:  'MySimCompany',
  clientId: 'MySimCompany@simrus.com',
  simulationInterval: 22,
  endpointType: 'socket',
  tripsPerHour: 88,
  drivers: 10,
  currencyCode: 'USD',
  capacity: 200,
  imageUrl:
'http://www.tripthru.com/assets/networks/fabless@tripthru.com.png
',
  acceptsPrescheduled: false,
  acceptsOndemand: true,
  acceptsCashPayment: true,
  acceptsAccountPayment: false,
  acceptsCreditcardPayment: true,
  coverage: [
       {city: 'San  Francisco', businessPercentage: 80},
       {city: 'Chicago', businessPercentage: 75}
   ]
};
```

To start the simulator using this new configuration file:

% node start mysimcompany.js

Sample output (note mysimcompany in output):

```
2015-04-09T04:51:12.948Z - init  :  Loading configuration mysimcompany.js
2015-04-09T04:51:12.948Z - init  :  Creating network mysimcompany.jsfrom configuration...
2015-04-09T04:51:12.950Z - init  :  Opening socket  client  mysimcompany.js...
2015-04-09T04:51:13.020Z - init  :  Successfully connected to:
mongodb://localhost:27017/partners
2015-04-09T04:51:13.025Z - init  :  Connected to http://107.170.220.160/
2015-04-09T04:51:13.025Z - init  :  Socket  open,  starting simulation mysimcompany.js...
2015-04-09T04:51:13.168Z - MySimCompany@simrus.com :  Emitted set-network-info
MySimCompany@simrus.com, res:  200
2015-04-09T04:51:49.894Z - 108634@hotelbeds@tripthru.com :  MySimCompany@simrus.com
received getQuote
2015-04-09T04:51:59.982Z - 105449@olabombay@tripthru.com :  MySimCompany@simrus.com
received getQuote
2015-04-09T04:52:02.179Z - sim :  Lawrence requests to be picked  up at -6.183931,
106.84099 on 2015-04-09T04:57:02+00:00 and dropped off  at -6.186581, 106.842338
2015-04-09T04:52:02.179Z - sim :  Queueing MySimCompanyChicago-
1@mysimcompanychicago@tripthru.com
2015-04-09T04:52:02.179Z - MySimCompanyChicago-1@mysimcompanychicago@tripthru.com :
Status  has  changed from  new  to  queued. Driver  location:
2015-04-09T04:52:02.181Z - MySimCompanyChicago-1@mysimcompanychicago@tripthru.com :
Ready  to  dispatch
2015-04-09T04:52:02.181Z - MySimCompanyChicago-1@mysimcompanychicago@tripthru.com :
```

```
Dispatch locally
2015-04-09T04:52:02.181Z - MySimCompanyChicago-1@mysimcompanychicago@tripthru.com :
Pickup location -6.183931,106.84099 is outside of coverage area
2015-04-09T04:52:10.203Z - 131722@dididachepeking@tripthru.com : MySimCompany@simrus.com
received getQuote
2015-04-09T04:52:20.383Z - 131728@dididachepeking@tripthru.com : MySimCompany@simrus.com
received getQuote
```

Configuring the Cities.js file

This file contains the cities the simulator uses to make requests.

The simulator is set up to simulate requests in 62 different cities. The list of cities is held in the file network_config/data/cities.js a snippet of the file is provided:

```
module.exports = {
  'Buenos Aires': { lat: -34.61, lng: -58.37 },
  'Vienna': { lat: 48.210390, lng: 16.372421 },
  'La Paz': { lat: -16.5, lng: -68.15 },
  'Brasilia': { lat: -15.78, lng: -47.91 },
  'Montreal': { lat: 45.52, lng: -73.57 },
  'Toronto': { lat: 43.65, lng: -79.38 },
  'Santiago': { lat: -33.46, lng: -70.64 },
  'Peking': { lat: 39.93, lng: 116.4 },
  'Bogota': { lat: 4.630861, lng: -74.087962 },
...
```

If you want to limit the cities that the simulator will make requests in you will need to edit the file network_config/data/cities.js. Here is an example of the network_config/data/cities.js with only three cities.

```
module.exports = {
  'Los Angeles': { lat: 34.049442, lng: -118.241733 },
  'New York': { lat: 40.671086, lng: -73.940440 },
  'San Francisco': { lat: 37.769996, lng: -122.447329 }
};
```

NOTE: The contents of this file must be consistent with the Service_js_file and the tripsByCity.js file. This means that the cities that are called out in both the Service_js_file and the tripsByCity.js must be present in the Cities.js file.

NOTE: it is suggested that a copy of the original network_config/Cities.js be kept for safe keeping prior to editing the cities.js file:

% cp network_config/data/cities.js network_config/data/cities_orig.js

Configuring the tripsByCity.js file

This file contains the trips within a city the simulator can to make requests.

The tripsByCity is a JSON set of trip coordinates in the form of start/end pairs which contain coordinates to indicate a requested route within a city. You may add routes to each city and they will be randomly used when a trip simulation is selected for that city.

Here is a snippet of the tripsByCity.js file:

```
module.exports = {
  'Buenos Aires': [
    {
      start: { lat: -34.610627, lng: -58.370665},
      end: { lat: -34.604384, lng: -58.372671}
    }
  ],
  'Vienna': [
    {
      start: { lat: 48.210386, lng: 16.370758 },
      end: { lat: 48.209671, lng: 16.375790}
    }
  ],
  'La Paz': [
    {
      start: { lat: -16.501529, lng: -68.146632 },
      end: { lat: -16.505092, lng: -68.144743 }
    }
  ],
...
```

You can add trips to this file for each city; here is a snippet of the file with an additional route added for Buenos Aires and Vienna:

```
module.exports = {
```

```
'Buenos Aires': [
  {
    start: { lat:  -34.610627, lng:  -58.370665},
    end:  { lat:  -34.604384, lng:  -58.372671}
  },
  {
    start: { lat:  -34.4213, lng:  -58.1889},
    end:  { lat:  -34.61623, lng:  -58.38334}
  }
],
'Vienna': [
  {
    start: { lat:  48.210386, lng:  16.370758 },
    end:  { lat:  48.209671, lng:  16.375790}
  },
  {
    start: { lat:  48.18290, lng:  16.621315 },
    end:  { lat:  48.889211, lng:  16.212348}
  }

],
'La Paz':  [
  {
    start: { lat:  -16.501529, lng:  -68.146632 },
    end:  { lat:  -16.505092, lng:  -68.144743 }
  }
],
```

NOTE: it is recommended that the lat and lng entries are legitimate locations in the cities they are configured for. To test these locations, use http://google.com/maps and enter the lat/lng in the search field to view the location. An example: -16.501529, -68.146632

NOTE: it is suggested that a copy of the original network_config/tripsByCity.js be kept for safe keeping prior to editing the tripsByCity.js file:

% cp network_config/data/tripsByCity.js network_config/data/tripsByCity _orig.js