Smart City Model Service Requirements

Author: Eric Gieseke

Date: 9/15/2020

Introduction

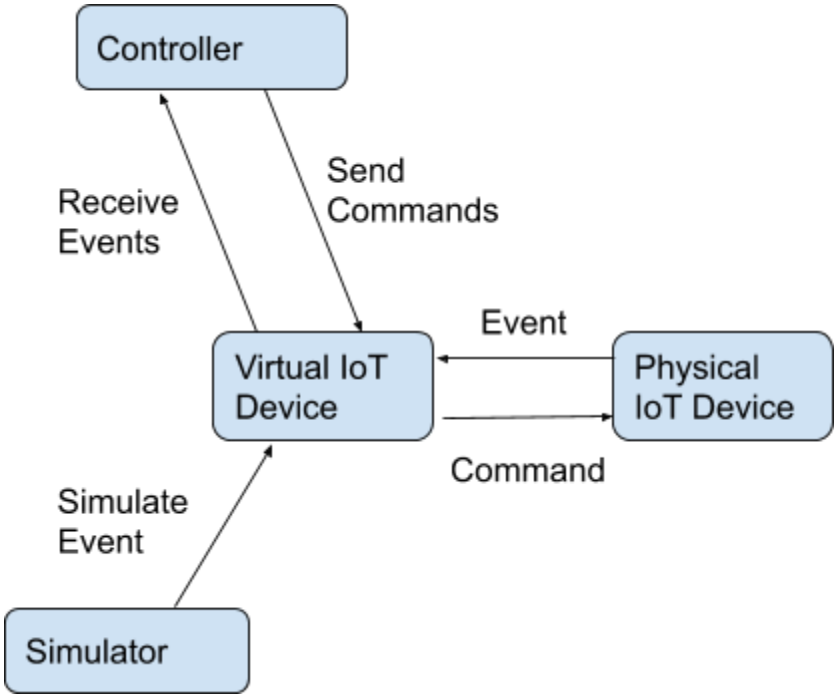
This document provides the requirements for the Smart City Model Service.

Overview

The Smart City Model Service is responsible for maintaining the state of Internet of Things (IoT) devices within the Smart City. The IoT Devices are designed to support the needs of city residents and guests. All IoT devices contain sensors that are able to collect and share data. Sensors include cameras, microphones, CO2 sensors, and thermometers. IoT devices are also able to generate sound output through speakers. IoT devices can be controlled through commands that they receive from an external Controller. The Controller is also able to receive IoT events.

IoT devices managed by the Smart City system include Street Lights, Parking Meters, Street Signs, Information Kiosk, Robots, and Vehicles. All IoT Devices contain a GPS sensor to determine their location which can be queried. Mobile IoT devices generate location events when their position changes.

Physical IoT Devices are managed using Virtual IoT Devices that represent the state of the associated physical device. The Virtual IoT device provides a proxy to the physical device. The following diagram shows the relationship between the virtual and physical IoT devices. It also shows how a Controller receives events and sends commands from and to the Physical IoT device via the Virtual IoT device.



For testing, a Simulator generates simulated events that appear to have come from the Physical IoT device. The Simulator will be used for testing the Smart City system.

Product API Requirements

This section defines the requirements for the Smart City Model Service.

The Smart City Model Service is primarily responsible for managing the state of the City domain objects including:

- City
- People
    - Resident
    - Visitor
- Iot Devices
    - Street Sign
    - Information Kiosk
    - Street Light
    - Robot
    - Parking Space
    - Vehicle
        - Bus

- Car

City

The City is used to model a city instance. Note that the Smart City system is a cloud based service and must be able to manage multiple Cities. A City has the following attributes:

- Globally unique identifier (e.g. city-1)
- Name (e.g. “Cambridge, MA”)
- Multiple people, either residents or visitors
- Multiple IoT Devices
- A blockchain account for receiving and sending money
- Location (lat, long)
- Radius which specifies the area encompassed by the city.

Person

Persons model the people that live in the city. A Person can be either a Resident or Visitor. Residents are well known persons, where visitors are anonymous. Both Residents and Visitors are assigned a unique person id.

Attributes of Residents include:

- Globally unique id
- Biometric Id
- Name of resident
- The phone number of the resident
- The Role of the resident (adult, child, or public administrator)
- Blockchain Account Address
- Location (lat/long)

Attributes of Visitor include:

- Globally unique id
- Biometric Id
- Location (lat/long)

Iot Devices

IoT Devices are the internet connected components of the Smart City. All IoT devices have the following attributes:

- A globally unique id
- Location(lat/long)
- Current status (ready, offline)
- Enabled (on/off)
- And the latest event emitted from the device

All IoT devices have the following input sensors:

- Microphone
- Camera
- Thermometer
- CO2 Meter

The Sensors generate events that are processed by the Virtual IoT Devices. Events have a type, action, and an optional subject. For example, the microphone may generate an event {microphone,“where is the nearest bus stop?”, resident:alice}. Note that the microphone is able to convert speech to text. Or the Camera may generate the event {camera,”person walking”, resident:bob}. Note that the microphone and camera sensors use AI to automatically identify the subject person. The CO2 Sensor may generate the following event {CO2, “400ppm”}, similarly, the Thermometer may report the current ambient temperature {thermometer, “88F”}, or the temperature of an individual {thermometer, “98.6F”, “jane”}.

In addition to input sensors, all IoT devices include a speaker for generating output speech, allowing the IoT devices to interact with Residents and Guests using natural language.

The following section describes the types of IoT devices:

Street Sign

A street sign is an IoT device that provides information for vehicles. It is able to alter the text displayed on the sign. For example, it can dynamically adjust the speed limit, or warn about an accident ahead.

Information Kiosk

The Information Kiosk helps residents and visitors. It is able to interact with Persons, though speech and displaying images. For example the Kiosk can display a map and help provide directions. The Kiosk can also support purchasing tickets for concerts and other events.

Street Light

The Street Light is an IoT device for illuminating the city. The Street Light is able to adjust its brightness.

Robot

Robots act as public servants. Robots are mobile and can respond to commands from Residents and Visitors. For example, helping to carry groceries. They can also asset in emergencies, for example putting out a fire.

Parking Space

A Parking Space is an IoT device able to detect the presence of a vehicle. A parking space has an hourly rate which is charged to the account associated with the vehicle.

Vehicle

Vehicles are mobile IoT Devices that are used for giving rides to Residents and Visitors. Vehicles can be either a Bus or a Car. Vehicles have a maximum rider capacity. Both Cars and Busses are autonomous. Riding in a Bus or Car is free for Visitors, but requires a fee for Residents.

Bus

Car

Smart City Model Service

The Smart City Model Service provides a top level Service interface for provisioning cities. It also supports controlling the City's IoT devices. Any external entity that wants to interact with the Smart City Model Service, must access it through the public API of the Model Service.

The Model Service provides a service interface for managing the state of the Cities.

The API supports commands for

- Defining the City configuration
- Showing the City configuration
- Updating the City configuration
- Creating/Simulating sensor events
- Sending command messages to IoT Devices
- Accessing IoT State and events
- Monitoring and supporting Residents and Visitors

All API methods should include an auth_token parameter that will be used later to support access control.

Command API

The Smart City Model Service supports a Command Line Interface (CLI) for configuring Cities and generating simulated sensor events. The commands can be listed in a file to provide a configuration script. The CLI should use the service interface to implement the commands.

Command Syntax:

City Commands

Define a city
```
define city <city_id> name <name> account <address> lat <float> long <float>
radius <float>
```

Show the details of a city. Print out the details of the city including the id, name, account, location, people, and IoT devices.
```
show city <city_id>
```

Device Commands

Define a street sign
```
define street-sign <city_id>:<device_id> lat <float> long <float> enabled
(true|false) text <text>
```

update a street sign
```
update street-sign <city_id>:<device_id> [enabled (true|false)] [text <text>]
```

Define an information kiosk
```
define info-kiosk <city_id>:<device_id> lat <float> long <float> enabled
(true|false) image <uri>
```

Update an information kiosk
```
update info-kiosk <city_id>:<device_id> [enabled (true|false)] [image <uri>]
```

Define a street light
```
define street-light <city_id>:<device_id> lat <float> long <float> enabled
(true|false) brightness <int>
```

Update a street light
```
update street-light <city_id>:<device_id> [enabled (true|false)] [brightness
<int>]
```

Define a parking space
```
define parking-space <city_id>:<device_id> lat <float> long <float> enabled
(true|false) rate <int>
```

Update a parking space
```
update parking-space <city_id>:<device_id> [enabled (true|false)] [rate
<int>]
```

Define a robot

```
define robot <city_id>:<device_id> lat <float> long <float> enabled
(true|false) activity <string>

# Update a robot
update robot <city_id>:<device_id> [lat <float> long <float>] [enabled
(true|false)] [activity <string>]

# Define a vehicle
define vehicle <city_id>:<device_id> lat <float> long <float> enabled
(true|false) type (bus|car) activity <string> capacity <int> fee <int>

# Update a vehicle
update vehicle <city_id>:<device_id> [lat <float> long <float>] [enabled
(true|false)] [activity <string>] [fee <int>]

# Show the details of a device, if device id is omitted, show details for all devices within the city
show device <city_id>[:<device_id>]

# Simulate a device sensor event
create sensor-event <city_id>[:<device_id>] type
(microphone|camera|thermometer|co2meter) value <string> [subject <person_id>]

# Send a device output
create sensor-output <city_id>[:<device_id>] type (speaker) value <string>
```

Person Commands

```
# Define a new Resident
define resident <person_id> name <name> bio-metric <string> phone
<phone_number> role (adult|child|administrator) lat <lat> long <long> account
<account_address>

# Update a Resident
update resident <person_id> [name <name>] [bio-metric <string>] [phone
<phone_number>] [role (adult|child|administrator)] [lat <lat> long <long>]
[account <account_address>]

# Define a new Visitor
define visitor <person_id> bio-metric <string> lat <lat> long <long>

# Update a Visitor
update visitor <person_id> [bio-metric <string>] [lat <lat> long <long>]

# Show the details of the person
```

```
show person <person_id>
```

Note: IoT events and commands will be defined as part of the city controller module in assignment 3.