Research online Publishing
www.ronpub.com

Open Access

*Open Journal of XXX (OJXX)*
*Volume X, Issue X, 20XX*
http://www.ronpub.com/ojxx
**ISSN XXXX-XXXX**

# An Interoperable Parking Data Standard for IoT Data Marketplaces: Experiences from developing a City-scale Real-time Parking Application

Gowri Sankar Ramachandran[A], Jeremy Stout[B], Joyce Edson[B], Bhaskar Krishnamachari[A]

[A] Viterbi School of Engineering, University of Southern California, Los Angeles, USA. {gsramach, bkrishna}@usc.edu
[B] Information Technology Agency, City of Los Angeles, Los Angeles, USA. {jeremy.stout, joyce.edson}@lacity.org

## Abstract

*Data marketplaces offer a viable solution to build large city-scale Internet of Things (IoT) applications. Contemporary data marketplaces such as Intelligent IoT Integrator (I3), Terbine, and Streamr present a middleware platform to help the data owners to sell their data to the application developers. However, such platforms suffer from adoption issues because of the heterogeneous nature of the data. On the one hand, the IoT devices and the software used by the device owners follow either a custom data standard or a proprietary industrial standard. On the other hand, the application developers buying data from multiple device owners expect the data to follow one common standard to improve interoperability. Therefore, a common data standard is desired to enable interoperable data exchange through a data marketplace while promoting adoption. In this paper, we present our experiences from developing a city-scale real-time parking application for a smart city. Besides, we also introduce a new data standard for smart parking.*

## Type of Paper and Keywords

Regular research paper: *IoT, Scalability, Smart City, Multi-Stakeholder, Data Economy*

## 1 Introduction

Managing city-scale IoT applications is challenging because of its management complexity. Data marketplaces are being considered as a solution for connecting hundreds of IoT devices with the application developers. Intelligent IoT Integrator (I3), Terbine, and Streamr are examples of data marketplaces. Such marketplace solutions offer a middleware platform to let the device owners sell their data to the application developers. This application development model provides a promising solution for building large city-scale IoT applications.

Such IoT marketplaces make data available to the application developers from a wide range of deployments, including parking sensors, weather stations, and solar monitoring systems. In this model, the data standard followed by the IoT devices and the device owners largely depend on the hardware, software, and other constraints, including privacy. Consider an application developer interested in building a parking monitoring system for a city using the data marketplace.

For this application, the application developer has to buy the parking data from multiple data providers, including the transportation department of cities, counties, and towns and various private garage owners. If each parking provider follows a custom and property data format, then the application developers have to convert the data from different sources into a single consistent format to aid their application, which leads to the following requirement:

1. A data marketplace must define data standards for each application to help the device owners and the application developers to easily sell data and build applications, respectively.

Besides, the lack of data standard may prevent the application developers from buying data from the marketplace, which, in turn, would reduce the revenue for the device and data owners. Therefore, data standards are one of the critical requirements to increase adoption while achieving economic sustainability.

Contemporary literature on data standards and IoT interoperability argues the need for a common standard [1, 5, 2]. Existing efforts in this space either focus on enabling interoperability between messaging protocols such as MQTT and CoAP [1, 5] or emphasize the need for interoperability at networking and MAC layers [2]. Other approaches for addressing data standardization include semantic interoperability [6, 7, 10]. Such methods lead to the standardization of networking and messaging protocols, but the data standardization remains an application-specific problem.

In this work, we show the interoperability challenges in developing a real-time parking application for a smart-city involving data marketplaces. In particular, we illustrate through multiple parking deployment scenarios why a new parking standard is desired to interconnect heterogeneous and real-time parking feeds to a data marketplace. Based on the review of the real-world parking feeds, we propose a new parking data standard for city-scale IoT applications.

The rest of the paper is structured as follows: **to be updated in the end**

## 2 Motivation

### 2.1 Parking Application

The vehicle population is continuously increasing in metropolitan cities, which also increases the demand for parking spaces. Multiple community members, including the government transportation agency, garage owners, and other private organizations, address the parking demands of the vehicle owners. However, studies suggest that the vehicle drivers are spending tens of hours searching for parking in each year [1]. Another research indicates that searching for parking costs $73 Billion for americans[2]. Minimizing the searching time have the potential to reduce fuel usage and cost, while immensely reducing the drivers' stress. Gathering real-time parking information from all the parking providers in the city and making that data available to drivers in real-time is essential to ease the driver's parking experience.

### 2.2 The role of a Data Marketplace

IoT data marketplaces are developed to let the device and data owners in the community sell their data to the application developers. Examples of IoT data marketplaces include I3, Terbine.io, and Streamr. Such data marketplaces enable the cities to develop large-scale city-wide IoT applications by leveraging the data sources provided by the community members. Following a marketplace-based application model, the city administration and the government agencies need not deploy and manage hundreds of IoT devices throughout the city for gathering sensor data. Instead, the community members deploy, manage, and make their IoT devices and their sensor data available to the city and the other application developers because the data marketplace provides a financial incentive for the sellers.

In the next section, we describe how a real-time parking application can be developed by using a data marketplace.

## 3 A Real-time Parking Application Using an IoT Data Marketplace

The City of Los Angeles' Information Technology Agency (ITA) is considering the marketplace-driven application to help the city and the community members make an informed decision. A real-time parking application is being developed in partnership with the University of Southern California and several industrial, government, and academic partners. In this section, we will present the architecture of this application.

### 3.1 Architecture of Marketplace-based Parking Application

Figure 1 shows the architecture of the real-time parking application that is currently under development at the City of Los Angeles. The key stakeholders and their roles in this application are described below.

[1] https://www.usatoday.com/story/money/2017/07/12/parking-pain-causes-financial-and-personal-strain/467637001/

[2] https://inrix.com/press-releases/parking-pain-us/

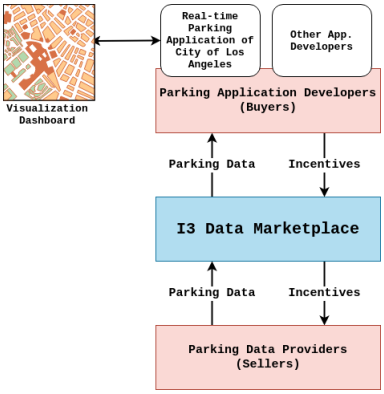


**Figure 1: The architecture of marketplace-based parking application currently under development at the City of Los Angeles.**

### 3.1.1 Parking Data Providers

These are owners and managers of public and private parking establishments. Most of the parking sites in metropolitan cities have a system in place to gather data about the parking availability in real-time. Currently, the parking information is mainly posted at the entrances of each parking site, although, the parking systems let the garage managers to share this information with other entities including the city administrators. Through an incentive-oriented application development model, we encourage the independent garage owners and managers to share their data with other application developers via the data marketplace, as shown in Figure 1.

### 3.1.2 I3 Data Marketplace

I3 [9, 11] is the open-source data marketplace developed at the University of Southern California, in collaboration with industrial and academic partners including the City of Los Angeles. In Figure 1, I3 marketplace middleware is used to bridge the data providers with the application developers. The key functionalities of the marketplace middleware includes user and device management, authentication and access control, ratings, data exchanging protocol, among other things. For more information, we refer the reader to I3 [9, 11]. Besides, an open-source software is also made available to the researchers and marketplace enthusiasts at the following link: `https://github.com/ANRGUSC/I3-Core`. For readers interested in understanding the operations through a demonstration, we have released a video here: `https://youtu.be/qFee7mlhriE`.

In our marketplace-based parking application, the City of Los Angeles and the University of Southern California are partnering with local parking establishments to feed parking availability data streams to the data marketplace in real-time.

| Neighborhood | Parking Type | Fields |
|---|---|---|
| Downtown Los Angeles (Parking Occupancy) | Street Parking | Spaceid, Eventtime, Occupancystate |
| Downtown Los Angeles (Parking Inventory for the above data stream) | Street Parking | Spaceid, Blockface, Metertype, Ratetype, Raterange, Timelimit, Parkingpolicy, Latlng |
| Los Angeles International Airport (LAX) | Multi-level Parking Structure | Lotdescription, Lot 1 Occupancy, Parkingid, Parkingname, Totalparkingspaces, Occupied, Freespaces, Fullcapacity, Color, Dataexportdatetime, Long (Longitude), Lat (Latitude) |
| Parking Deployment at Los Angeles | Multi-level Parking Structure | lotName, lotID, totalSpots, availableSpots, occupiedSpots, percentOccupied, percentAvailable, occupancyLevel, availabilityLevel, parkingPolicies, spotAvailability |

**Table 1: Examples of data formats used by the parking providers.**

### 3.1.3 Parking Application Developers

These are community members including the government agencies, private organizations, and other individuals interested in building an IoT application using the data sources available in the data marketplace. Figure 1 shows how the application developers are receiving parking data from the I3 data marketplace by making a payment. There can be tens to hundreds of application developers at the north-end of the data marketplace focusing on various IoT applications. In our parking application use case, the City of Los Angeles is

in the process of creating a real-time parking application using the I3 data marketplace.

Figure 2 shows the visualization dashboard of the parking that is currently under development. The real-time parking information is currently available for 7 different neighbourhoods including the Downtown Los Angeles, parts of Hollywood, Los Angeles International Airport, and Long Beach. This parking application shows parking availability on streets and parking garages. The community members can check the parking availability in real-time based on the color coding. For a particular parking garage or a street, if the number of free parking spots exceed 50%, then it is denoted in yellow color. Otherwise, the map presents the results in red color. Besides, the users can click each color coded block to see how many parking spots are currently available including its total parking capacity. At the time of writing this paper, the parking application was hosted online here: `https://findmeaspot.lacity.org/`.

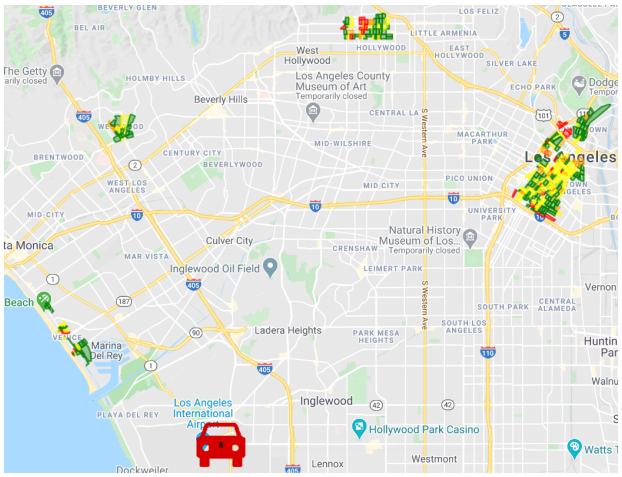


**Figure 2: The Visualization Dashboard of Real-time Parking Application developed by the City of Los Angeles.**

### 3.1.4 Interoperability Challenges

Although the architecture of the parking application presented in Figure 1 provides a platform the City of LA to gather parking feeds from various neighbourhood, there were a number of interoperability challenges, which prevents the City of LA from taking feeds via the I3 data marketplace platform. At the time of writing this paper, the parking application was directly fetching data from various parking data providers through a custom application. In this section, we will describe the interoperability challenges and why a custom application is not sustainable for the City of Los Angeles.

**Protocol Inconsistency:** All the parking data providers in the city use REST-based services for sharing their parking data. However, the I3 data marketplace that we use in this application employ MQTT, which is a publish-subscribe protocol. Protocols following request-reply messaging model is not scalable for data marketplaces, which is the reason behind the selection of MQTT as a messaging protocol for our I3 data marketplace. The parking data providers either have to develop a custom software to convert their REST API feed into a MQTT stream.

**Heterogeneous Data Format:** Each parking data provider follow a different data format, therefore, the application developers cannot consume the data without writing a custom data parser for each parking provider. Table 1 provides examples of parking data formats from real-world deployments.

**Continuous Management of Custom Software:** To deal with protocol and data format inconsistencies, a custom software can be developed. However, such a software has to be employed at the data provider's end. Running custom software for each data provider at the marketplace middleware is not scalable, and it leads to continuous development and management of software for each protocol and data format, which is not sustainable. Alternatively, the application developers could also deploy a custom software as part of their application, but this model creates a friction, and may discourage the application developers from adopting the marketplace-based application model.

These challenges show that the parking monitoring infrastructure deployed and managed by various government and private agencies are not interoperable, which hinders their effectiveness and utility. Note that the parking deployments have limited use, because it is only used to display the parking availability information at the entrances of the parking garages and nearby streets. Therefore, the vehicle drivers are still required to drive close to the digital displays close to the parking garages to check the parking availability. Creating a city-wide real-time parking application therefore offer a promising alternative, but it require a set of common standards.

In this paper, we review the parking data standards followed by real-world parking deployments and present a novel parking data standard to mitigate the data interoperability challenges at data marketplaces and other large-scale city-wide parking applications.

### 3.2 Design Requirements

Table 1 shows the different parking data formats currently employed by parking management systems. In particular, the table highlights the following issues:

- **Multiple parking types:** Multiple parking modalities are presented in a city. These range from street-side parking, single-level parking garages, to multi-level parking structures.
- **Diverse reporting model for parking availability:** Some parking establishments provide individual spot status using a Boolean variable that switches between *Occupied* and *Vacant*. In the case of multi-level parking structures, the information about the *total spots* and *total availability* is reported, and some deployments could also provide parking availability at each level.
- **Inconsistent Metadata:** Building a map-based visualization or to understand the freshness of the data, it is important to receive information including the GPS coordinates and timestamp associated with the last update. In the case of street-side parking, an information about specific parking spots. Some parking deployments provide both the metadata and the parking availability information through a single API. But, in some cases, multiple APIs are required to interpret the parking data for a particular location. As shown in Table 1, the parking availability data is reported using spaceids, and the application developer is required to issue an additional API to gather location information associated with the spaceid.

These issues must be taken into account when creating a new parking data standard. We list down the requirements below:

- **R1** The parking data format should cover all types of parking infrastructures.
- **R2** All the relevant metadata should be part of the parking data standard, and it should not require multiple data queries or messages to interpret the parking data.
- **R3** The spatial data should be embedded within the standard to help the application developers to create map-based visualizations.
- **R4** The data should follow a programmer-friendly data schema.

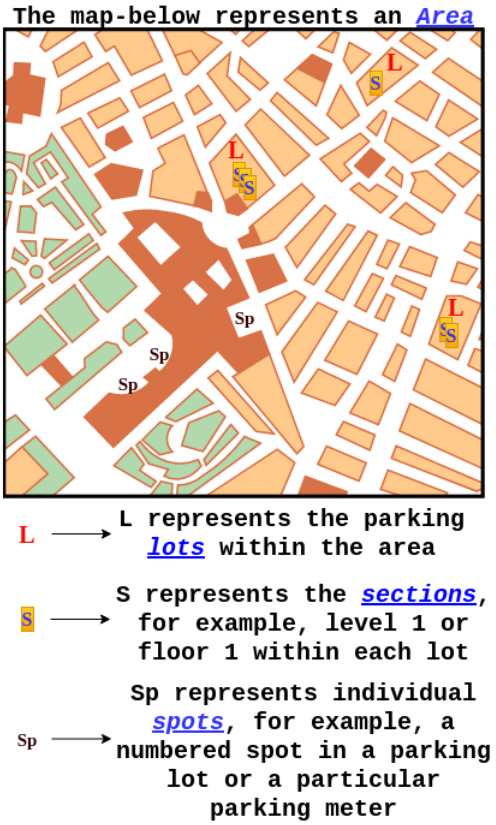


**Figure 3: Hierarchical layers followed by our parking standard.**

## 4 An Interoperable Data Standard for City-scale Real-time Parking Application

We propose a new parking data standard satisfying the requirements elucidated in Section 3.2. The key features of our parking data standard are discussed below:

### 4.1 Capturing Spatial Relationships through a Hierarchical Layering Schema

Figure 3 shows the hierarchical layering scheme followed by our parking standard.

Within a city, there are multiple areas or neighbourhood. For example, the Los Angeles International Airport (LAX), Downtown Los Angeles, and an university campus are considered as areas. In our parking standard, an *area* covers one particular neighbourhood. Formally, we can denote the area as $\mathcal{A}$. Within a city, there can be $n$ areas, which can be represented as $a_1, a_2, ..., a_n$.

Within each area, $a_i$, there could be multiple parking

lots. For example, Lot 6 within LAX airport area or City Center Parking in Downtown Los Angeles area. In our parking standard, a *lot* covers a particular parking structure, which has tens to hundreds of parking spaces across one or more levels (or floors). Formally, we can represent lots as $\mathcal{L}$. Within each area, $a_i$, there can be $n$ parking lots, which can be represented as $l_1, l_2, ..., l_n$.

Within each lot, $l_i$, there could be multiple sections (or floors). For example, section 2 (denotes level 2 or floor 2) of Lot 6 within LAX airport area or section 5 (denotes level 5 or floor 5) of City Center Parking in Downtown Los Angeles area. In our parking standard, a *section* covers a particular segment or a level of a parking lot, wherein each section may have tens of parking spaces. Formally, we can represent sections as $\mathcal{S}$. Within each lot, $l_i$, there can be $n$ sections, which can be represented as $s_1, s_2, ..., s_n$.

Within each lot, $s_i$, there could be multiple individual parking spots. For example, the parking spot 12 in section 2 (denotes level 2 or floor 2) of Lot 6 within LAX airport area or the parking spot 28 in section 5 (denotes level 5 or floor 5) of City Center Parking in Downtown Los Angeles area. In our parking standard, a *spot* refers to individual parking spot, which is the lowest granularity level. Formally, we can represent spots as $\mathcal{P}$. Within each section, $s_i$, there can be $n$ individual spots, which can be represented as $p_1, p_2, ..., p_n$.

**How is this standard covering the areas with only street-side parking or single-level parking structure?** Although our hierarchical parking standard covers the different types of parking establishments, not all areas in a city may have multi-level parking lots with multiple sections. Some areas may only have street-side parking or a single-level parking structure. In our parking standard, each higher level of the hierarchy could either stand-alone or be itself a collection of lower levels. The following variations are valid data formats in our standard:

- An Area can be a collection of lots (e.g. LAX) or collection of spots (downtown), or may be just a stand-alone unit (no further subdivision).
- A lot can be a collection of sections, or collection of spots, or stand-alone.
- A section can be collection of spots or stand-alone.
- A spot is always stand-alone.

We further explain the above variations with two practical examples in Section 5.

## 4.2 Attributes

In this section, we present the data attributes of our parking data standard:

**Area-specific Attributes:** Table 2 presents the area-specific attributes used in our parking standard. The key represents the fields that should be included in the area segment of the data, and the attribute values and the data types for each attribute are also presented to help the parking application developers. Most of the fields are self-explanatory, except for AreaGeometry, which captures the shape of the area. For each area, we could draw a polygon or other geometric shapes by tracing a set of GPS coordinates. For example, a square-shaped area can be represented with four GPS coordinates.

| **Attribute Key** | **Attribute Value** | **Data Type for Attribute Value** |
|---|---|---|
| ***Area-specific Attributes*** | | |
| Type | Area | String |
| OwnerInfo | Parking provider info | String |
| AreaID | Alphanumeric identifier | String |
| AreaName | Name of the area | String |
| AreaLatLong | Latitude and Longitude | Key-value Store |
| AreaGeometry | Spatial coordinates | Key-value Store |
| Timestamp | Last update timestamp | String (YYYY-MM-DDTHH:MM:SS.SSS) |
| TotalSpots | Total number of spots in the area | Integer |
| OccupiedSpots | Total number of occupied spots | Integer |

**Table 2: Area-specific Attributes**

**Lot-specific Attributes:** The lot-specific attributes are similar to the area-specific attributes, in terms of the keys and values. It contains the following fields: Type: Lot, OwnerInfo, LotID, LotName, LotLatLong, LotGeometry, TimeStamp, TotalSpots, and OccupiedSpots. A table is not created for lot-specific attributes to avoid redundancy. Except for the *Type* field, all the other items are similar to area-specific attributes.

**Section-specific Attributes:** The section-specific attributes are also follow the similar pattern. It contains the following fields: Type: Section, OwnerInfo, SectionID, SectionName SectionLatLong, SectionGeometry, TimeStamp, TotalSpots, and OccupiedSpots. Here, the *Type* field should contain the

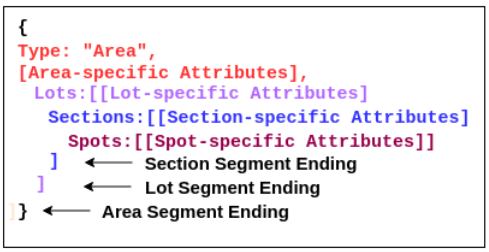


**Figure 4: The high-level JSON Schema of our Parking Data Format.**

value "Lot".

**Spot-specific Attributes:** The spot-specific attributes have the the following fields: Type: Spot, OwnerInfo, SpotID, SpotName, SpotLatLong, SpotGeometry, TimeStamp IsOccupied (True or False), SpotPolicy. Unlike other segments in the data format, the spot segment maintains a *Boolean* attribute called "IsOccupied", which is used to identify the status of a single parking spot. Additionally, there is also an attribute called "SpotPolicy", which is introduced to specify options such as Unrestricted, HandicapOnly or PermitOnly, and it could be further extended further if needed.

All these attributes are maintained in a JSON document [3]. We chose JSON because it is easy to handle JSON documents in popular programming languages such as Python, Java, and JavaScript.

### 4.3 Specifications for Geometry Attribute

Figure 5 shows the difference between *Point* and *Polygon* shapes. This attribute follows the *GeoJSON* data format [4]. We will show an example of Geometry attribute in Section 5.

### 4.4 Parking Payment Policies

An optional attribute is introduced to represent the parking payment policies, which is shown in Table 3. A "PolicyType" field is added to identify the pricing model for a given parking segment (area, lot, section, or spot). And, the timing field is used to represent the timing limitations, while the support for prepaid payment is expressed through a Boolean value.

### 4.5 Parking Reservation Policies

Some parking vendors may also provide support for reservation of parking spaces. We include an attribute in our parking standard to handle such circumstances,

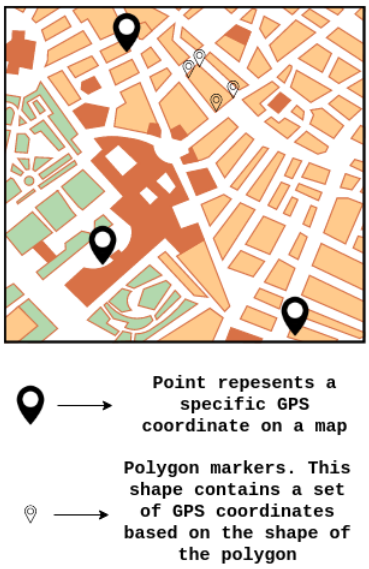


**Figure 5: Geometry Attribute: Point vs Polygon.**

which is shown in Table 4.Note that that details of future parking availability and how to make reservations are relegated to the URL and not specified in this standard. It may of interest to standardize those in the future, but it is out of scope here.

### 4.6 Security and Integrity of Parking Data

In order to gain a guarantee that the data comes from a valid source (source authentication) and has not been tampered with (integrity), a digital signature could be included in the JSON. The whole Parking JSON file could be sent as payload of a JWT [8] (Java Web Token), which adds a header and signature field (using JWS (Java Web Signature)). The header can include the certificate and cryptographic algorithms used for the signature.

### 4.7 Requirements Evaluation

Section 3.2 elicited the design requirements for a parking standard. Here, we show how our proposed parking standard satisfy those requirements:

- **R1** is satisfied through the use of hierarchical layering schema, which covers all the way from an entire area to an individual parking spot.
- Our parking standard includes name, identifier, GPS coordinates, geometry, and timestamp for area, lots, sections, and spots. Therefore, an application developer can process the parking using a single data feed, which fulfils **R2**.

| Attribute Key | Attribute Value | Data Type for Attribute Value |
|---|---|---|
| **Key: PaymentPolicy, Values are as follows:** | | |
| PolicyType | Free, FlatRate, UnitRate, TwoPhaseFlat (i.e. initially free for some time, then flat fee), TwoPhaseUnitRate (i.e. initially free for some time, then charged by the minute/hour) | String |
| Timing | MaxTime, TimeUnit (for UnitRate), FlatRatePrice or RatePerTime, InitialTime (for TwoPhase) | String |
| IsPrepaid | True or False | Boolean |

**Table 3: Optional Attribute for Payment Policies**

| Attribute Key | Attribute Value | Data Type for Attribute Value |
|---|---|---|
| **Key: ReservationPolicy, Values are as follows:** | | |
| IsReservable | True or False | Boolean |
| Max Reservation Time | Time in Minutes (How far in advance it can be reserved?) | Integer |
| ReservationURL | URL | String |

**Table 4: Optional Attribute for Reservation Policies**

- When visualizing parking data on a map, it is important to process information about the geometrical shape. Our parking standard includes geometry for area, lots, sections, and spots, which satisfies **R3**.
- Through the use of JSON document, we satisfy requirement **R4**.

## 5 Practical Examples

Example 1 shows the JSON document for one of the USC campuses. Due to the space limitation, we represent a partial data for two parking lots. But, a complete JSON document for this location would have multiple parking lots with tens of sections and hundreds of invidual spots. Besides, it is important to note how the Geometry attribute is used to carry the spatial coordinates that are associated with the parking data. In this case, the areaGeometry attribute can be used to draw a polygon on the map to denote the area. For each parking spot, coordinates are provided to draw a rectangle around each parking spot.


```
{
  "Type": "Area",
  "Attributes":[
  {"OwnerInfo":"USC"},
  {"AreaID":"usc5428"},
  {"AreaName":"USC UPC Campus"},
  {"AreaLatLong":[-118.39,33.94]},
  {"AreaGeometry":{
                "type": "Polygon",
                "coordinates": [
                    [
                        [
                            -118.39,
                            33.94
                        ],
                        [
                            -118.40,
                            33.94
                        ],
                        [
                            -118.40,
                            33.94
                        ],
                        [
                            -118.39,
                            33.94
                        ],
                        [
                            -118.39,
                            33.94
                        ]
                    ]
                ]
  }
  },
  {"Timestamp":"2019-12-07T21:22:48.1
     20"},
  {"TotalSpots":2023},
  {"OccupiedSpots":1949}
  ],
  "Lots":[
  {
    "Type":"Lot",
```

```
    "OwnerInfo":"USC",
    "LotID":"Lot1",
    "LotName":"LotDowney",
    "LotLatLong":[-118.39,33.94],
    "LotGeometry":{
      "Notes": "Not shown in this
          example due to space
          restrictions. But it will
          follow the format presented
          in previous area, lot, and
          section segments"
             },
    "Timestamp":"2019-12-07T21:22:48.
       120",
    "TotalSpots":455,
    "OccupiedSpots":"324",
    "Sections":[
      {
      "Type":"Section",
      "OwnerInfo":"USC",
      "SectionID":"usclot1floor1",
      "SectionName":"698floor1",
      "SectionLatLong":[-118.39,33.94
          ],
      "SectionGeometry":{
                "Notes": "Not shown
                    in this example
                    due to space
                    restrictions. But
                    it will follow the
                     format presented
                    in previous area,
                    lot, and section
                    segments"
             },
    "Timestamp":"2019-12-07T21:22:48.
       120",
    "TotalSpots":102,
    "OccupiedSpots":78
    }
    ]
    },
    {
      "Type":"Lot",
      "OwnerInfo":"USC",
      "LotID":"Lot2",
      "LotName":"LotShrine",
      "LotLatLong":[-118.39,33.94],
      "LotGeometry":{
                "Notes": "Not shown
                    in this example
                    due to space
                    restrictions. But
                    it will follow the
                     format presented
                    in previous area,
                    lot, and section
                    segments"
             },
    "Timestamp":"2019-12-07T21:22:48.
       120",
    "TotalSpots":655,
    "OccupiedSpots":"344",
    "Sections":[
      {
      "Type":"Section",
      "OwnerInfo":"USC",
      "SectionID":"usclot2floor1",
      "SectionName":"438floor1",
      "SectionLatLong":[-118.39,33.94
          ],
      "SectionGeometry":{
                "Notes": "Not shown
                    in this example
                    due to space
                    restrictions. But
                    it will follow the
                     format presented
                    in previous area,
                    lot, and section
                    segments"
             },
    "Timestamp":"2019-12-07T21:22:48.
       120",
    "TotalSpots":148,
    "OccupiedSpots":78,
    "Spots":[
    {
      "Type":"Spot",
      "OwnerInfo":"LADOT",
      "SpotID":"defgh456",
      "SpotName":"DT124",
      "SpotLatLong":[-118.39,33.94],
      "SpotGeometry":{
                "Notes": "Not shown
                    in this example
                    due to space
                    restrictions. But
                    it will follow the
                     format presented
                    in previous area,
                    lot, and section
                    segments"
             },
    "Timestamp":"2019-12-07T21:22:48.
       120",
    "IsOccupied":"True",
    "SpotPolicy":"HandcicapOnly"
```

```
        }
        ]
        }
        ]
        }
        ]
}
```

Example 1: JSON Document for USC Campus with two parking lots. Each parking lot has one section, wherein one of the parking lots include a single spot-specific data. This example is created to show how area, lots, sections, and spots would be represented following our parking format.

## 6 Conclusion and Future Work


### Acknowledgements

This work is supported by the USC Viterbi Center for Cyber-Physical Systems and the Internet of Things (CCI).


### References


[1] B. Ahlgren, M. Hidell, and E. H. Ngai, "Internet of things for smart cities: Interoperability and open data," *IEEE Internet Computing*, vol. 20, no. 06, pp. 52–56, nov 2016.

[2] G. Aloi, G. Caliciuri, G. Fortino, R. Gravina, P. Pace, W. Russo, and C. Savaglio, "A mobile multi-technology gateway to enable iot interoperability," in *2016 IEEE First International Conference on Internet-of-Things Design and Implementation (IoTDI)*, 2016, pp. 259–264.

[3] P. Bryan and M. Nottingham, "Javascript object notation (json) patch," *RFC 6902 (Proposed Standard)*, 2013.

[4] H. Butler, M. Daly, A. Doyle, S. Gillies, S. Hagen, T. Schaub *et al.*, "The geojson format," *Internet Engineering Task Force (IETF)*, 2016.

[5] P. Desai, A. Sheth, and P. Anantharam, "Semantic gateway as a service architecture for iot interoperability," in *2015 IEEE International Conference on Mobile Services*, 2015, pp. 313–319.

[6] M. Ganzha, M. Paprzycki, W. Pawłowski, P. Szmeja, and K. Wasielewska, "Semantic interoperability in the internet of things: An overview from the inter-iot perspective," *Journal of Network and Computer Applications*, vol. 81, pp. 111 – 124, 2017. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1084804516301618

[7] G. M. Honti and J. Abonyi, "A review of semantic sensor technologies in internet of things architectures," *Complexity*, vol. 2019, 2019.

[8] M. Jones, J. Bradley, N. Sakimura, and J. W. Token, "Rfc 7519," *JSON Web Token*, p. 111, 2015.

[9] B. Krishnamachari, J. Power, S. H. Kim, and C. Shahabi, "I3: an IoT marketplace for smart communities," in *Proceedings of the 16th Annual International Conference on Mobile Systems, Applications, and Services*. ACM, 2018, pp. 498–499.

[10] M. Noura, M. Atiquzzaman, and M. Gaedke, "Interoperability in internet of things: Taxonomies and open challenges," *Mobile Networks and Applications*, vol. 24, no. 3, pp. 796–809, 2019.

[11] X. Zhao, K. Karyakulam Sajan, G. Ramachandran, and B. Krishnamachari, "Demo abstract: The intelligent iot integrator data marketplace — version 1," in *Proceedings of the 5th ACM/IEEE Conference on Internet of Things Design and Implementation*, ser. IoTDI '20, 2020.


### Author Biographies

Gowri Ramachandran is a postdoctoral researcher at the Center for Cyber Physical Systems and the Internet-of Things (CCI) at the University of Southern California. He received his Ph.D. from imec-DistriNet, KU Leuven, Belgium. His research interests include Internet-of-Things (IoT), smart cities, and blockchain.

Bhaskar Krishnamachari received his Ph.D. degree from Cornell University, Ithaca, NY, USA, in 2002. He is currently a Professor with the Department of Electrical and Computer Engineering, and Director of the Center for Cyber-Physical Systems and the Internet of Things. His research interests include the design and analysis of algorithms and protocols for the Internet of Things, Wireless Networks, and Distributed Systems.