SPOT PARKING API

Open Data Dynamic Data

Version 1.4 – Document Version 1.0 02 October 2020

Spot Parking Pty Ltd - www.spotparking.com.au

Trademarks

Disclaimer

Document Lifetime

Spot Parking Pty Ltd may occasionally update documentation between releases of the related software. Consequently, if this document was not provided recently, it may not contain the most up-to-date information. Please email developers@spotparking.com for the most current information.

Where to get help

Spot Parking support, product, and licensing information can be obtained as follows.

Product information — For general information regarding Spot Parking products, licensing, and service, go to the Spot Parking website at:

https://www.spotparking.com.au

Technical support — For technical support, please email opendatapi@spotparking.com.au.

Your comments

Your suggestions will help us continue to improve the accuracy, organization, and overall quality of the user publications. Please send your opinion of this document to:

opendataapi@spotparking.com.au

If you have issues, comments, or questions about specific information or procedures, please include the title and, if available, the revision, the page numbers, and any other details that will help us locate the subject that you are addressing.

Preface

Intended Audience

This guide is part of the Spot Parking Open Data API specifications documentation set. It is intended for use by developers as a reference for integrating Spot Parking's parking zone information with existing capabilities.

Readers should be familiar with the following: RESTful APIs, JSON

Style Conventions

The following style conventions are used in this document:

Bold

Names of commands, options, programs, processes, services, and utilities

Names of interface elements (such windows, dialog boxes, buttons, fields, and menus)

Interface elements the user selects, clicks, presses, or types

Italic

Publication titles referenced in text

Emphasis (for example a new term)

Variables

`Courier`

System output, such as an error message or script

URLs, complete paths, filenames, prompts, and syntax

Courier italic

Variables on command line

User input variables

< >	Angle brackets enclose parameter or variable values supplied by the user
[ ]	Square brackets enclose optional values
|	Vertical bar indicates alternate selections - the bar means "or"
{ }	Braces indicate content that you must specify (that is, x or y or z)

Table of Contents

Document History

Paper copies are valid only on the day they are printed. Contact Spot Parking if you are in any doubt about the accuracy of this document.

Revision History

This document has been revised by:

Revision Number	Revision Date	Summary of Changes	Author
v1.0	22-10-2020	Initial Revision	Scott Taylor

Reference Documents

Please see the following documents for more information:

Document Name	Version	Author
Open Data API Overview	V1.01	Spot Parking
Geohash Wikipedia Entry	As per Wikipedia https://en.wikipedia.org/wiki/Geohash	Community authored

1. Overview

The Spot Parking platform gives you a group of APIs along with client libraries, language-specific examples, and documentation to help you develop applications that integrate with Spot Parking.

The Dynamic Data API provides capabilities to determine additional information in relation to a parking asset that is dynamic in nature – eg. will change frequently on a weekly basis. An example of this type of data includes those necessary to understand (near)real-time occupancy. It caters for a variety of different query and data types to suit typical use-cases.

Throughout the document, recommended approaches for how the API should be utilized are specified, and it is highly encouraged that these are adopted.

To use this API, you must be provided with an **API Token**. If you haven't been provided with an API Token, please contact opendataapi@spotparking.com.au for information on how to obtain one.

1.1. Conventions

We use the following conventions in this document:

- Responses are listed under 'Responses' for each method.
- Responses are in JSON format.
- Request parameters are mandatory unless explicitly marked as Optional.

1.2. Current Version

The Dynamic Data API will continue to evolve, and changes to this API are managed through a version management scheme. Versioning access is maintained explicitly via the URL path structure, and not within HTTP Request-Headers. Spot Parking will endeavor to maintaining previous versions of the API ongoing unless formal advanced notice is provided for its decommissioning.

1.3. Schema Summary

A full explanation of the data components and their relationships can be found in the section titled **Data Structures and Relationships**.

All API access is over HTTPS using an authenticated token, and accessed from the https://data-collection-api.spotparking.com.au base URL path.

All data is sent and received in JSON.

All GPS coordinates are provided using WGS 84 coordinate system projections.

1.4. HTTP Requests

API requests must be written as HTTPS requests, and include the following components:

- **HTTP Method**:
- **URL**: As specified in specific API specification (case is important)
- **HTTP Headers**: Authentication and encoding headers are expected.
- **Request Body**: As specified in specific API specification (case is important)

1.5. HTTP Methods

Dynamic Data Simple API requests supports GET method only.

The Dynamic Data Complex API requests supports POST method only. This is due to the need to provide query parameters in JSON object notation within the HTTP Request Body. Utilizing POST method ensures compatibility with any third-party client HTTP libraries.

1.6. Authentication

Authentication is achieved via the use of a Token, provided to you by Spot Parking. The Token must be passed for all API requests within the HTTP Headers. Invalid or missing tokens will result in a HTTP Status Code 401 Unauthorised response.

1.7. URL Format

Describe the format of the URL.

The API URL uses the following format:

```
<protocol>://<host>:<port>/1.4/<MethodName>
```

Example:

```
POST https://data-collection-api.spotparking.com.au/1.4/dynamic/complex
```

2. Data Structures and Relationships

2.1. Schema Overview

Dynamic data is stored within the Spot Parking platform as a simple key/value store associated with a resource. A resource is a specific parking asset as defined within the platform and is treated independently of zones.

One or more zones may refer to the same resource – for example in a parking lot (as depicted in Figure 1) where there are many different configuration of bays (zones), all of these could refer to the same overall lot resource to determine occupancy information.

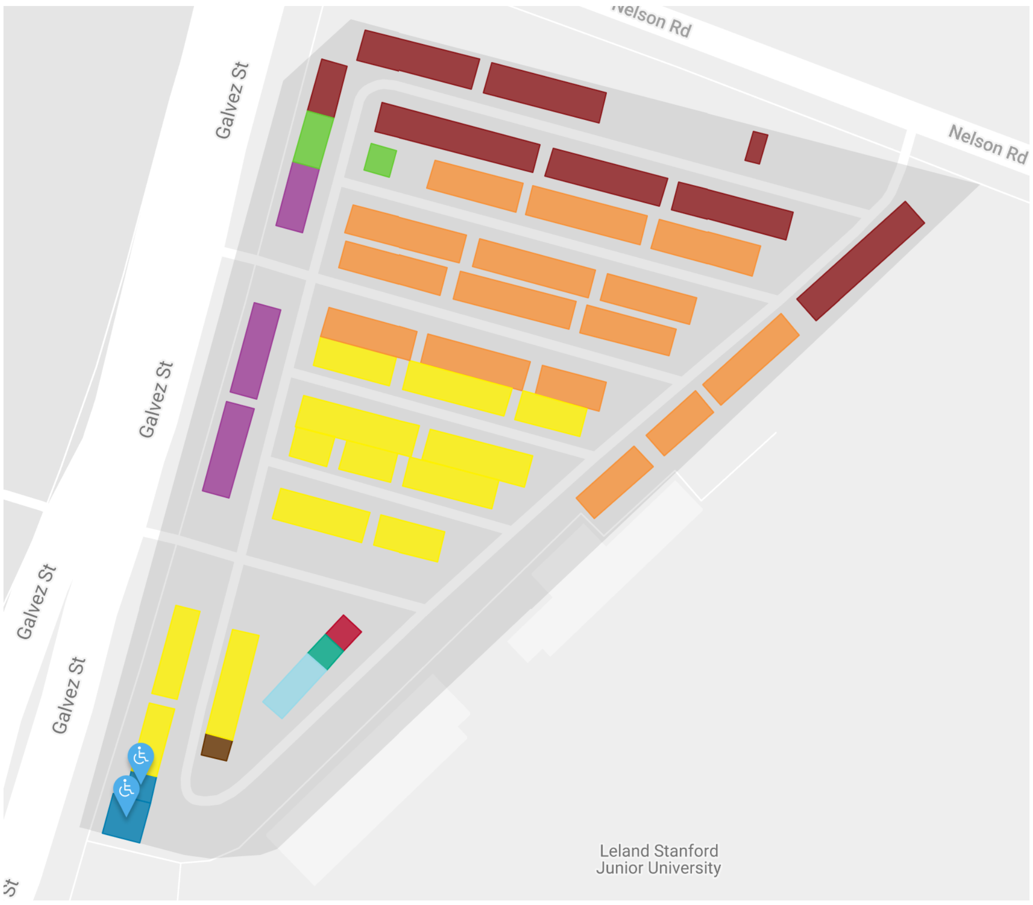

Figure 1 - Multiple zones within a lot

2.2. Key/Value Store Responses

As the key/value store can contain a variety of different types, the value returned consists of both the type (eg. Number, String, Array) and the value itself. The following describes the schematic layout of the response, along with examples to illustrate potential implementations of this schema.

```
interface DynamicDataResponse<key, T> {
        [key]:  ValueResponse<T> | ValueResponseAsArray<T>;
}
```

```
interface KeyValueResponse {
       value:  ValueResponse<T> | ValueResponseAsArray<T>;
}

interface ValueTypeDef {
       type:   ValueType;
}

interface ValueResponse<T> extends ValueTypeDef {
       value:  Value<T>;
}

interface ValueResponseAsArray<T> extends ValueTypeDef {
       values: Array<T>;
}

enum ValueType {
       Number = "Number",
       Date = "Date",
       Boolean = "Boolean",
       String = "String",
       Link = "Link",
       Array = "Array",
}

type Value<T as BooleanValue|StringValue|DateValue|LinkValue|ArrayValue>

type Array<T as BooleanValue|StringValue|DateValue|LinkValue|ArrayValue>

type BooleanValue = Boolean;

type StringValue = String;

type DateValue = String; // ISO Formatted

type LinkValue = Link;

type ArrayValue<T> = Array<T>;


interface Link {
       label: String;
       url:   String;
}
```

Example implementations of the `DynamicDataResponse` schema for different types:

```
{
     "occupancyRate": {
         "type": "Number",
         "value": 0.63
     }
 }

 {
     "occupancyLastUpdated": {
         "type": "Date",
         "value": "2020-09-23T00:36:03.651Z"
```

```
    }
}

{
    "links": {
        "type": "Array",
        "values": [
            {
                "type": "Link",
                "value": {
                    "label": "Ticket purchase",
                    "url": "https://parkandpay.com/checkout?zone=329843"
                }
            },
            {
                "type": "Link",
                "value": {
                    "label": "More info",
                    "url": "https://website.com/page"
                }
            }
        ]
    }
}
```

2.3. Type Enum Definitions

The following section describes the type enum definitions used within Dynamic Data API response schema.

2.3.1. Value Types

The `ValueType` describes the type of data contained within the `value|values` field of the `ValueResponse|ValueResponseAsArray` structures respectively.

Value Identifier	Description
Number	The field value is a number
Boolean	The field value is a boolean
String	The field value is a string
Date	The field value is a date formatted as a string in ISO format
Link	The field value is a `Link` structure used for specifying URLs.

Array	The field value is an array of values.

2.4. Data Structures

The following section describes the individual data structures used within the Dynamic Data API responses.

2.4.1. QueryRequest

```
interface QueryRequest {
      [resourceId1]:       ResourceBundle;
      [resourceId2]:       ResourceBundle;
      [resourceId..n]:     ResourceBundle;
}
```

2.4.2. QueryResponse

```
interface QueryResponse {
      [resourceId1]:       ResourceBundle;
      [resourceId2]:       ResourceBundle;
      [resourceId..n]:     ResourceBundle;
}
```

The `QueryResponse` structure consists of a dictionary of one or more resource bundles, each of which contains dynamic data related to the resource. The key of the dictionary is the `resourceId` for easy programmatic access to resource-specific information.

The following is a JSON example of a `RequestReponse` response containing occupancy information for two separate resources.

```
{
      "8f111740-6b83-4dc2-856b-d802e59084fb": {
            "occupancy": [
                  {
                        "occupancyRate": {
                              "type": "Number",
                              "value": 0.64
                        }
                  }
            ]
      },
      "a5f7e943-fbe8-4427-ae5e-7d5d3eff5fdf": {
            "occupancy": [
                  {
                        "occupancyRate": {
                              "type": "Number",
                              "value": 0.73
                        }
                }
            ]
        }
}
```

2.4.3. ResourceRequest

```
interface ResourceRequest {
      [containerLabel1]:        Array<String>;
      [containerLabel2]:        Array<String>;
      [containerLabel..n]:      Array<String>;
      <_referenceVariable1>?:   String;
      <_referenceVariable2>?:   String;
      <_referenceVariable..n>?: String;
}
```

The `ResourceRequest` structure consists of a dictionary of one or more containers of dynamic data fields to be retrieved. The key of the dictionary is the `containerLabel` for easy programmatic access to a set of related information within a resource bundle and is free for the calling application to determine (must not contain an underscore "_" prefix).

In addition, the calling API request can pass special reference variables (identified by an underscore _ prefix), which will be placed in the `ResourceBundle` associated with the resource. This is used to allow easy matching during post-processing of the resource bundle to other assets such as zones.

The following is an JSON example of a `ResourceRequest` request. The calling API has requested dynamic data for two separate resources. The first resource only requests the occupancy rate, the second requires the occupancy last updated field as well. The container label is defined as "occupancy" but could be any label of choice. Also passed are reference variables called "_zoneId" which provide zoneId information for each resource.

```
{
        "8f111740-6b83-4dc2-856b-d802e59084fb": {
                "_zoneId": "5428101f-f4f4-45ec-958c-369fce9d6f39",
                "occupancy": [
                        "occupancyRate"
                ]
        },
        "a5f7e943-fbe8-4427-ae5e-7d5d3eff5fdf": {
                "_zoneId": "66158f96-38ab-4476-98d4-e925990b80ad",
                "occupancy": [
                        "occupancyRate",
                        "occupancyLastUpdate"
                ]
        }
}
```

2.4.4. ResourceBundle

```
interface ResourceBundle {
      [containerLabel1]:        Array<DynamicDataResponse>;
      [containerLabel2]:        Array<DynamicDataResponse>;
      [containerLabel..n]:      Array<DynamicDataResponse>;
      <_referenceVariable1>?:   String;
      <_referenceVariable2>?:   String;
      <_referenceVariable..n>?: String;
}
```

The `ResourceBundle` structure consists of a dictionary of one or more containers of dynamic data. The key of the dictionary is the `containerLabel` for easy programmatic access to a set of related information within a resource bundle. The container label is supplied by the calling application via the API Request parameters.

In addition, the calling API request can pass special reference variables (identified by an underscore _ prefix), which will be placed in the bundle associated with the resource. This is used to allow easy matching during post-processing of the resource bundle to other assets such as zones.

The following is an JSON example of a `ResourceBundle` response. The calling API has requested a container label of "occupancy" to contain the requested dynamic data associated with occupancy information and has also passed a reference variable called "_zoneId" which has the referenceId for the associated zoneId.

```
{
      "8f111740-6b83-4dc2-856b-d802e59084fb": {
            "_zoneId": "5428101f-f4f4-45ec-958c-369fce9d6f39",
            "occupancy": [
                  {
                        "occupancyRate": {
                              "type": "Number",
                              "value": 0.64
                        }
                  },
                  {
                        "occupancyLastUpdate": {
                              "type": "Date",
                              "value": "2020-09-23T00:35:02.955Z"
                        }
                  }
            ]
      }

}
```

3. API Reference Documentation

3.1. Overview

The Dynamic Data API is a API resource providing two methods for obtaining dynamic data. One of these suits the quick access of dynamic data for a singular resource with limited container structuring, and the other allows for many resources with flexible structuring into containers.

Method	Purpose
Simple Query	Allows a limited query pertaining to one resource only. It is referred to within this document as a **simple** query.
Complex Query	Allows for a more flexible query pertaining to one or more resources. Dynamic data can be structured in one or more containers. It is referred to within this document as a **complex** query.

4. Simple Query

Given a set of query parameters, returns dynamic data information for a singular resource under one container.

4.1. Resource Information

The Dynamic Data API Simple Query resource information is as follows:

Method	Purpose
Response formats	JSON (application/json)
Requires authentication?	Yes (X-API-Token Header)
Rate limited?	No
Requests	N/A

4.2. Request

The Dynamic Data API Simple Query resource request information is as follows:

Method	URL
GET	https://data-collection-api.spotparking.com.au/1.4/ dynamic

Note: Please take consideration of case in all API calls.

4.3. Headers

The Dynamic Data API Simple Query requires the following HTTP Header information to be passed within the request in order to function:

Header	Description	Example / Setting
X-API-Token	Authentication Token (provided by Spot Parking)	Example: fHoX5l4bo22Xvv7n5dQDaFf7p

4.1. Parameters

The Dynamic Data API Simple Query expects all parameters to be passed as querystring. The following parameters are acceptable or expected:

Name	Type	Description	Required
resourceId	String	The identifier of the associated resource that the dynamic data requested belongs to	Mandatory
keys	Strings separated with commas	A list of one or more dynamic data fields to be returned, separated by commas.	Mandatory
container	String	The `containerLabel` to group returned dynamic data information into. If container is not specified, then the response will default to a label called 'default'	Optional
_<field>	String	Any querystring parameters prefixed with an underscore will be returned in the response within the `ResourceBundle` structure. This is useful to maintain references to other assets such as zones when post-processing the response.	Optional

4.1.1. Request Parameter Examples

The following examples demonstrates how parameters can be provided to perform certain types of queries.

```
?resourceId=22fa1e5c-9a67-40a1-9c5e-ceee65115763&keys=occupancyRate,occupancyLastUpdate
```

Retrieves the occupancyRate and occupancyLastUpdate dynamic data for the resource identified by `22fa1e5c-9a67-40a1-9c5e-ceee65115763`. Information will be returned in a container labeled "default".

```
?resourceId=22fa1e5c-9a67-40a1-9c5e-ceee65115763&keys=poiLinks&container=info
&_zoneId=5428101f-f4f4-45ec-958c-369fce9d6f39
```

Retrieves the dynamic data for the resource identified by `22fa1e5c-9a67-40a1-9c5e-ceee65115763`. Information will be returned in a container labeled "info" and the _zoneId field will also be placed into the container response.

4.2. Response

The Dynamic Data API Simple Query provides a response in JSON format. The structure of the response is as follows:

```
interface SimpleQueryResponse {
     data: QueryResponse;
}
```

The Dynamic Data API Simple Query will only contain dynamic data for one resource, and with one container consolidating all the dynamic data requested.

4.3. Example

```
GET /1.4/dynamic?resourceId=a5f7e943-fbe8-4427-ae5e-
7d5d3eff5fdf&keys=occupancyRate,occupancyLastUpdate&_zoneId=5428101f-f4f4-45ec-958c-
369fce9d6f39&container=info HTTP/1.1
Host: data-collection-api.spotparking.com.au
User-Agent: curl/7.64.1
Accept: */*
X-API-Token: <API_Token>


HTTP/1.1 200 OK
Access-Control-Allow-Headers: Origin, X-Requested-With, Content-Type, Accept, X-API-Token
Access-Control-Allow-Methods: GET, POST, PUT, DELETE
Access-Control-Allow-Origin: *
Content-Type: application/json; charset=utf-8
Date: Wed, 23 Sep 2020 06:13:12 GMT
ETag: W/"10a-zwIFuJ92uVKUHmYWdIVme46Bqi0"
X-Powered-By: Express
Content-Length: 266
Connection: keep-alive
```

```json
{
    "data": [
        {
            "resourceId": "a5f7e943-fbe8-4427-ae5e-7d5d3eff5fdf",
            "key": "occupancyRate",
            "value": {
                "type": "Number",
                "value": 0.33
            }
        },
        {
            "resourceId": "a5f7e943-fbe8-4427-ae5e-7d5d3eff5fdf",
            "key": "occupancyLastUpdate",
            "value": {
                "type": "Date",
                "value": "2020-09-23T06:13:03.632Z"
            }
        }
    ]
}
```

5. Complex Query

Given a set of query parameters, returns dynamic data information for one or more resource under one or more containers.

5.1. Resource Information

The Dynamic Data API Complex Query resource information is as follows:

Method	Purpose
Response formats	JSON (application/json)
Requires authentication?	Yes (X-API-Token Header)
Rate limited?	No
Requests	N/A

5.2. Request

The Dynamic Data API Complex Query resource request information is as follows:

Method	URL
POST	https://data-collection-api.spotparking.com.au/1.4/ dynamic/complex

Note: Please take consideration of case in all API calls.

5.3. Headers

The Dynamic Data API Complex Query requires the following HTTP Header information to be passed within the request in order to function:

Header	Description	Example / Setting
X-API-Token	Authentication Token (provided by Spot Parking)	Example: fHoX5l4bo22Xvv7n5dQDaFf7p
Content-Type	MIME Type of JSON	application/json

5.4. Parameters

The Dynamic Data API Complex Query resource expects all parameters to be passed within a JSON object structure passed via the Request Body. The following parameters are acceptable or expected:

Name	Type	Description	Required
requiredData	Structure	Contains a `QueryRequest` structure containing the information required	Mandatory

5.4.1. Request Parameter Examples

The following examples demonstrates how parameters can be provided to perform certain types of queries.

```
{
      "requiredData": {
            "8f111740-6b83-4dc2-856b-d802e59084fb": {
                  "_zoneId": "5428101f-f4f4-45ec-958c-369fce9d6f39",
                  "occupancy": [
                        "occupancyRate"
                  ]
            },
            "a5f7e943-fbe8-4427-ae5e-7d5d3eff5fdf": {
                  "_zoneId": "66158f96-38ab-4476-98d4-e925990b80ad",
                  "occupancy": [
                        "occupancyRate",
                        "occupancyLastUpdate"
                  ]
            }
      }
}
```

5.5. Response

The Dynamic Data API Complex Query provides a response in JSON format. The structure of the response is as follows:

```
interface ComplexQueryResponse {
     data: QueryResponse;
}
```

Unlike the Dynamic Data API Simple Query, a Complex query can contain dynamic data for multiple resources, and with multiple containers consolidating all the dynamic data requested.

5.6. Example

```
POST /1.4/dynamic/complex HTTP/1.1
Host: data-collection-api.spotparking.com.au
User-Agent: curl/7.64.1
Accept: */*
X-API-Token: <API_Token>
Content-Type: application/json
Body:
{
    "requiredData": {
        "8f111740-6b83-4dc2-856b-d802e59084fb": {
            "_zoneId": "5428101f-f4f4-45ec-958c-369fce9d6f39",
            "occupancy": [
                "occupancyRate"
            ]
        },
        "a5f7e943-fbe8-4427-ae5e-7d5d3eff5fdf": {
            "_zoneId": "66158f96-38ab-4476-98d4-e925990b80ad",
            "occupancy": [
                "occupancyRate",
                "occupancyLastUpdate"
            ]
        }
    }
}
```

```
HTTP/1.1 200 OK
Access-Control-Allow-Headers: Origin, X-Requested-With, Content-Type, Accept, X-API-Token
Access-Control-Allow-Methods: GET, POST, PUT, DELETE
Access-Control-Allow-Origin: *
Content-Type: application/json; charset=utf-8
Date: Wed, 23 Sep 2020 06:13:12 GMT
ETag: W/"10a-zwIFuJ92uVKUHmYWdIVme46Bqi0"
X-Powered-By: Express
Content-Length: 266
Connection: keep-alive
```

```
{
    "data": {
        "8f111740-6b83-4dc2-856b-d802e59084fb": {
            "_zoneId": "5428101f-f4f4-45ec-958c-369fce9d6f39",
            "occupancy": [
                {
                    "occupancyRate": {
                        "type": "Number",
                        "value": 0.29
                    }
                }
            ]
        },
        "a5f7e943-fbe8-4427-ae5e-7d5d3eff5fdf": {
            "_zoneId": "66158f96-38ab-4476-98d4-e925990b80ad",
            "occupancy": [
                {
                    "occupancyRate": {
                        "type": "Number",
                        "value": 0.34
                    }
                },
                {
                    "occupancyLastUpdate": {
                        "type": "Date",
                        "value": "2020-09-23T06:21:03.387Z"
                    }
                }
            ]
        }
    }
}
```

6. Status Codes

The API uses the following HTTP status codes. 2XX – Success; 4XX - Error in client; 5XX - Error in server.

Status Code	Description
200	OK
201	Created
202	Accepted (Request accepted, and queued for execution)
400	Bad request
401	Authentication failure
403	Forbidden
404	Resource not found
405	Method Not Allowed
409	Conflict
412	Precondition Failed
413	Request Entity Too Large
500	Internal Server Error
501	Not Implemented
503	Service Unavailable

7. Determining Available Dynamic Data Fields

Available dynamic data fields are indicated by the `dynamicData` field found in resources such as zones.

An example of a JSON Document for Zones zone definition with dynamic data defined is shown below:

```
{
      "id": "91abab08-28b2-43cc-bb07-0a8823cfa60a",
      "paths": { … },
      "assetType": "garage",
      "schedule": { … },
      "customName": "Peninsular Carpark",
      "dynamicData": {
            "occupancy": {
                  "keys": [
                        "occupancyRate",
                        "occupancyLastUpdate"
                  ],
                  "refreshRate": 120
            }
      },
      "resourceId": "8f111740-6b83-4dc2-856b-d802e59084fb"
}
```

The two important fields to consider here are `dynamicData` and `resourceId`.

Within the `dynamicData` field is a container with two dynamic data keys for occupancy, as well as a field called `refreshRate`. The `refreshRate` refers to the recommended refresh frequency for updating the field information in seconds. In this example, it is recommending an update every two minutes.

The `resourceId` field represents the resourceId to pass to the Dynamic Data Query APIs as defined.

Typically, a JSON Document or the ZoneGroups API call may result in multiple zones with the `dynamicData` field defined. It is highly recommended to utilise the **Complex Query** method and collect the dynamic data of all resources within one request, rather than generating multiple **Single Query** requests, placing unnecessary burden on the Spot Platform infrastructure.