SPOT PARKING API

Open Data API Overview

Version 1.0 – Document Version 1.01 02 October 2020

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

Trademarks

All trademarks or registered trademarks are the property of their respective owners.

Disclaimer

The information provided in this document is provided "as is" without warranty of any kind. Spot Parking Pty Ltd disclaims all warranties, either express or implied, including the warranties of merchantability and fitness for a particular purpose. In no event shall Spot Parking Pty Ltd be liable for any damages whatsoever including direct, indirect, incidental, consequential, loss of business profits or special damages, even if Spot Parking Pty Ltd or its suppliers have been advised of the possibility of such damages.

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 opendataapi@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 opendataapi@spotparking.com.au.

Your comments

Your suggestions will help us continue to improve the accuracy, organization, and overall quality of our 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 OpenData API specifications documentation set. It is intended for use by developers as an overview of Spot Parking's platform and associated capabilities.

Readers should be familiar with the following: RESTful APIs, Google Protocol Buffers

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	14-09-2020	Initial Revision	Scott Taylor
V1.01	01-10-2020	Small updates in case treatment and grammar.	Scott Taylor

Reference Documents

Please see the following documents for more information:

Document Name	Version	Author
Open Data JSON Document for Zones	V1.0	Spot Parking
Open Data ZoneGroups 1.4 API Reference Documentation	V1.05	Spot Parking
Open Data Dynamic Data 1.4 API Reference Documentation	V1.0	Spot Parking
Geohash Wikipedia Entry	As per Wikipedia https://en.wikipedia.org/wiki/Geohash	Community authored

1. Overview

Welcome to Spot Parking. The Spot Parking platform provides convenient access to business rules and metadata associated with a range of geocoded parking-related assets. These assets have been captured, interpreted and analysed according to the local jurisdictional laws and regulations that underpin their use, and made available digitally through a series of APIs and regular data extracts for rapid consumption for a range of applications.

The objective of the platform is two-fold:

- Provide a useful facade that removes the complexity of requiring localised knowledge of parking rules and regulations, irrespective of jurisdiction, vehicle or user contexts, and
- Deliver the information necessary to allow informed decisions, whether human or automated, about parking-related assets and their associated classifications and entitlements with the speed and efficiency demanded of modern applications.

In other words, the Spot Parking platform aims to be able to answer two fundamental questions quickly and accurately:

1. Can I park here?
2. And if so, under what conditions?

1.1. Utility of Information

Originally, Spot Parking foresaw a critical missing dataset component necessary for fully autonomous vehicle navigation - the need to understand the business rules associated with parking at a specific destination. How can software make intelligent decisions about the suitability of a parking location based upon the user needs (how long, how much), relevant conditions or entitlements (e.g. disabled permits, commercial vehicle) and local rules and regulations? How does software access this information in a universal way, not requiring access to a series of proprietary applications and data stores tied to specific locations or contexts?

Although supporting the autonomous vehicle future remains core to Spot Parking's vision, the usefulness of the platform to a broader range of applications became more apparent over time. Here are just a few examples where information provided by the Spot Parking platform can be applied:

- **transport modelling** - understanding the impact of clearways and on-street parking on major arterial road networks to facilitate scenario testing and traffic predictions.
- **ride-sharing services** - determining the legality and suitability for pick-up/drop-off locations, or extended rest.

- **mapping and navigation** - visually depicting and aiding in the navigation of suitable parking locations.
- **enforcement** - accurately determining the entitlements of a specific parking instance, assisting rangers in making correct decisions in the issuement of citations/fines.
- **logistics and route optimisation** - ascertaining commercial loading locations (and potential availability of) to optimise routes for deliveries.
- **city planners** - determining the profitability and practicality of parking assets across to optimise revenue and usefulness.

1.2. Information formats

Spot Parking provides access to its open-data licensed datasets in two different forms. One is a set of regularly published standalone JSON documents made easily accessible via a standardized URI format specification. The second leverages the Spot Platform API Gateway directly, utilising a set of flexible RESTful APIs to access a more exhaustive set of information in near real-time.

The following describes the main advantages and disadvantages for each approach:

Accessibility form	Advantages	Disadvantages
Standalone JSON	• Easily consumable by existing technologies and languages • Data structure can be easily viewed and understood	• Restricted view of available data • Not real-time (only updated weekly) • Data structure is not in compressed format and therefore may not be suitable to consume directly in client applications • Data is provided in large geographic blocks only. • No custom user profiling is available. • Does not cater for occupancy data
RESTful API	• Provides near real-time information on demand – will always be updated with the latest changes. • Data retrieval is optimized for speed and efficiency of access in most cases (up to 10x), making it far more suitable for client-based applications to access directly.	• Requires API authentication and initial onboarding from Spot Parking • Most of Spot Parking's APIs use **Google Protobuf** format exclusively – a more advanced level of integration is necessary to integrate directly using this standard.

	• The full range of data is available, including the handling of specific user contexts (such as disabled permits, area permits etc) and dynamic data (such as occupancy information)	

It is highly recommended to look at the standalone JSON format first, as it may likely suit what you are looking for, without the additional complexity to handle Google Protobuf formatted data.

1.3. Information Types

Originally, Spot Parking thought the most important component necessary for understanding parking-related business rules was individual parking sign information. However, it was quickly determined that parking signage by itself is insufficient to determine all the business rules that apply to any particular parking asset. For example, parking signs rarely describe what tariffs (if any) underpin the use of the parking asset, or what are the specific rules for users that use a motorbike, or perhaps have a disabled permit.

Other than parking signs, there are additional factors to consider:

- are there kerb or road markings that further restrict or enable the specific parking use of the asset?
- are there any jurisdiction concerns that need to be considered (regulations and by-laws at an area, city, state or federal level)?
- where there are multiple signs within a specific location, what are the associated relationships and priorisations between each of those signs? Where there is conflicting information between them, how does that get determined?
- are there additional signs (such as the definition of a generalized parking area, school zones or clearway/towaway signs) that may not appear at each specific instance where individual parking signs are located, but apply business rules specific to that area?

For both information formats, Spot Parking provides at a fundamental level two key pieces of information:

1) **Where** does this information apply?
2) **What** rules apply **when** at this location?

1.3.1. The 'Where'

The Spot Platform defines a location where common parking business rules apply as a **zone**. A zone could consist of a series of contiguous signs, all defining the same

conditions. For example, a series of 5 parking signs all stipulating 'No Stopping' along the side of a road is defined as one zone.

Parking assets come in a mixture of configurations, from non-marked on-street parking, individual parking lots through to multi-level carparks and garages. The type of parking asset involved will impact how their location is defined:

1) For **on-street parking** locations, these are typically defined as simple polygon paths, following the left or right side of the street in question.

2) For most other configurations, these consist of either a simple polygon or a series of polygons (defined as a complex object).

Metadata may exist that further describes the location in question (eg. type of facility it is, number of levels etc).

All GPS coordinates are provided using WGS 84 coordinate system projections so that they can be directly used in the typical mapping applications.

1.3.2. The 'What' and 'When'

Parking assets can be assigned a complex set of rules which dictate under what circumstances/conditions parking can (or not) occur. Spot Parking provides this information in a schedule format, indicating defined intervals of time when a business rule applies, and what attributes are associated with that applied business rule.

The schedule format allows for very flexible queries to occur, including the ability to see ahead or in the past, not just in the current time. You can, for example, understand whether it is possible to legally stay in one parking asset for an extended period, irrespective if the business rules change repeatably during that time.

Spot Parking provides for three weeks of scheduled data per parking asset, starting from a defined base date (the start of the current week, or the week prior). Schedules are regenerated every week on early Tuesday morning, thus at any one point you are guaranteed at least two weeks (the current week and the following week) of schedule information to work from.

Please note that schedules require regeneration weekly due to the need to accommodate:

- New changes introduced due to parking asset business rule changes (tariffs etc), and or events or temporary closures to assets.
- Schedule changes due to daylight saving transitions
- Changes in rules related to public holidays or school holiday periods

The platform removes the complexity of having to deal with these issues independently. It remains context-aware of specific conditions and exceptions based on jurisdictional or user contexts and will accommodate these within the schedule accordingly.

Ingestion or consumption of Spot Parking information requires a minimum weekly update (via published standalone JSON Documents). The use of the RESTful API interface negates this issue, as it will always have the latest, up-to-date information available.

1.3.3. Static versus Dynamic Data

The Spot Platform provides for both static and dynamic data (via RESTful API) queries.

Static data is provided directly within the standalone JSON Documents or via the ZoneGroups API. Static data is unlikely to change during the week, for example a zone set for 'No Stopping' at all times is probably safe to assume that it will remain so for the duration of the week.

Dynamic data, on the other hand, is very likely to change. Dynamic data includes things such as real-time occupancy information for garages and parking lots.

Dynamic data availability meta-data is incorporated within static data elements when it exists. For example, a zone referring to a garage may include references to a set of dynamic data attributes that are retrievable. These elements can then be requested directly by the **DynamicComplex** API call (see its specific documentation for details), usually at a frequency recommended within the meta-data itself.

1.4. Authentication

For RESTful API use, authentication is achieved via the use of a Token, provided to you by Spot Parking as part of the onboarding process. 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.

There is no authentication requirement necessary for accessing Spot Parking data via the standalone JSON Document format.

2. Geohashes

Spot Parking uses an open-source hashing algorithm to refer to geographic locations (known as a Geohash) when requesting information from the platform. A geohash is a convenient way of expressing a location (anywhere in the world) using a short alphanumeric string, with greater precision obtained with longer strings. There are varying formats of geohashes available, some open-source and others proprietary. Spot's underlying infrastructure supports the use of an open-source implementation (see https://www.movable-type.co.uk/scripts/geohash.html) to remain independent from map provider-focused solutions.

Geohashes are used for both retrieving standalone JSON documents as well as interfacing with the Spot Parking platform via RESTful APIs. For convenience, a helper API is available (see below) that can directly translate a latitude/longitude coordinate into its equivalent geohash form. However, the recommended approach is to leverage the open-source libraries available in many languages that can do this programmatically for you. There is more detailed information about the use of geohashes, and recommended libraries in the specific JSON Document and ZoneGroup API documentations.

For further background information, refer to the Wikipedia entry for 'Geohash' online.

2.1. Geohash Utility API Overview

The Geohash Utility API is a singular API resource to aid in the conversion of standard coordinates into the equivalent geohash form.

Method	Purpose
Coordinate To Geohash	Generates geohash information in relation to the provided coordinate

3. Coordinate To Geohash

Given a specified latitude and longitude, returns detailed information about the equivalent geohash, including its boundary distances, and associated neighboring geohashes.

3.1. Resource Information

The Geohash API Coordinate To Geohash resource information is as follows:

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

3.2. Request

The Geohash API Coordinate To Geohash resource request information is as follows:

Method	URL
GET	https://data-collection-api.spotparking.com.au/1.1/utils/coordinateToGeohash

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

3.3. Headers

The Geohash API Coordinate To Geohash resource 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 (as provided by Spot Parking)	Example: fHoX5l4bo22Xvv7n5dQDaFf7p

3.4.

3.4. Parameters

The Geohash API Coordinate To Geohash resource expects all parameters to be passed within the URI querystring. The following parameters are acceptable or expected:

Name	Type	Description	Required
latitude (or lat)	A numeral (with decimal places).	The latitude value for the associated geo-coordinate. Must be a number value between -90 and 90.	Mandatory
longitude (or lon) (or lng)	A numeral (with decimal places).	The longitude value for the associated geocoordinate. Must be a number value between -180 and 180.	Mandatory
precisions	One or more geohash precisions between 5 – 8. Multiple precisions can be specified in comma delimited form.	The number of characters (precision) of the returned geohashes. The utility will only accept precision requests of between 5 – 8 characters *Note defaults to all precisions between 5 – 8*	Optional

3.4.1. Request Parameter Examples

The following examples demonstrates how parameters can be provided to request different conversions.

```
/1.1/utils/coordinateToGeohash?lat=-33.798172&lng=151.286663&precisions=5,6
```

This request would return geohash information for the specified coordinate in both 5 and 6 precisions.

3.5. Response

The Geohash API Query provides a response in JSON format, according to the following structure:

```
{
  "<precision>":  {
    "geohash": "<converted_geohashes>",
    "neighbours": { <neighbours> },
```

```
    "distance": { <distance> }
  },
  "<precision>": …
}
```

An explanation of the fields as below:

Name	Type	Description
precision	A string	The string equivalent of the precision integer.
converted_geohash	A string	The geohash for the coordinate in the precision specified. For example: "r3gxdx" for a geohash with precision 6.
neighbours	A structure (dictionary) of associated geohash neighbours, with the key denoting the compass direction abbreviation.	The surrounding neighbouring geohashes for the converted geohash.
distance	A structure (dictionary) that denotes the distance (in metres) of the geohash area in both horizontal and vertical directions	Structure contains two attributes: "ew" denoting horizontal distance, and "ns" denoting vertical distance.

3.6. Example

The following is an example output of the request illustrated above:


```
{
    "5": {
        "geohash": "r3gxd",
        "neighbours": {
            "n": "r3gxf",
            "ne": "r3gxg",
            "e": "r3gxe",
            "se": "r3gx7",
```

```
            "s": "r3gx6",
            "sw": "r3gx3",
            "w": "r3gx9",
            "nw": "r3gxc"
        },
        "distance": {
            "ew": 4064.89507578896,
            "ns": 4892.576772359856
        }
    },
    "6": {
        "geohash": "r3gxdx",
        "neighbours": {
            "n": "r3gxf8",
            "ne": "r3gxfb",
            "e": "r3gxdz",
            "se": "r3gxdy",
            "s": "r3gxdw",
            "sw": "r3gxdq",
            "w": "r3gxdr",
            "nw": "r3gxf2"
        },
        "distance": {
            "ew": 1016.4530159109615,
            "ns": 611.5720965448935
        }
    }
}
```


3.7. 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
401	Authentication failure
422	Invalid parameters
500	Internal Server Error
503	Service Unavailable