SPOT PARKING API

Open Data JSON Document for Zones

Version 1.0 – Document Version 1.02 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 Open Data API specifications documentation set. It is intended for use by developers as a guide on how to utilise the standalone JSON datasets for parking-asset related business rules.

Readers should be familiar with the following: JSON parsing and notation

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

1. Overview 6
1.1. Current Version 6
1.2. Schema Specifications 6
2. Zones Overview 7
2.1. Document Structure 7
2.2. Zone Schema 7
3. Condition Attributes 21
4. JSON Document Access 24
4.1. Base URI 24
4.2. Dataset Catalog 24
4.3. Archival datasets 25
5. Geohashes 26
5.1. Overview 26

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	V1.0	Spot Parking
Open Data ZoneGroups 1.4 API Reference Documentation	V1.04	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

The Spot Parking platform provides a group of APIs along with client libraries, language-specific examples, and documentation to help you develop applications that integrate with Spot Parking. In addition, Spot Parking publishes a set of JSON Documents that provides for easy consumption by applications, but with a limited subset of data. This document describes the structure of these JSON Documents, their limitations and their recommended use.

The JSON Documents provide the capability to determine the local parking rules and entitlements within specific locations for a generic user profile. From these documents, applications can understand the different categories of parking available, and what available entitlements (if any) a user with a standard motor vehicle (eg. car) with no additional permits or permissions may have to park within that location (eg. business rules).

Note: The Spot Parking platform provides for a far greater level of customisation, but this capability is only available through the use of platform RESTful APIs. If you need to cater specifically for special use-cases such as determining entitlements for permit holders, or for special vehicle categories, then the use of RESTful APIs is required.

Unlike the RESTful API, access to the JSON Documents is available on the terms specified via the Open Data platform. A specific API Token is not required in order to access these documents.

1.1. Current Version

The JSON Document for Zones schema will continue to evolve, and changes to this Specification will be managed through a version management scheme. Where possible, Spot Parking will endeavour to maintain previous versions of this Specification ongoing unless formal advanced noticed is provided for its decommissioning.

1.2. Schema Specifications

This document will contain references to online resources that assist in the consumption of the JSON Documents. This includes a JSON-Schema (draft-06 format) document, and a series of client libraries (for different languages) that adhere to this standard. The use of these materials is strongly recommended, especially where auto-complete functionality is available in developer tooling.

Together with this document, a developer should be able to fully understand the structure of the JSON documents, alongside the associated meanings and potential use of the contained data.

2. Zones Overview

The JSON Documents for Zones contain a collection of zones. A zone is an individual location where a parking asset exists, with its own set of business rules. A zone could identify a part of on-street parking on the side of a street, a lot (known as an outline), a set of off-street parking bays, or a garage/multi-level.

Zone location information is specified as a series of coordinates, either as a polyline (line) or as one (simple) or more (complex) set of polygons (area).

Associated with each zone is a set of business rules, specified in a schedule format. The schedule defines a base date from when the associated business rules apply, and a series of calendar events/intervals that individually define what specific business rules apply when. JSON Documents will always contain three weeks of calendar events from the base date and are updated weekly.

A zone may contain references to dynamic data – this is a reference to datasets containing additional data that is likely to change during the week, and therefore requires additional access via a RESTful API. The type of data that is dynamic includes things like real-time occupancy rates etc. This data doesn't make sense to publish within the JSON Documents, but the availability of such data can be easily determined and then obtained if needed.

2.1. Document Structure

Each JSON document consists of a collection of **Zones** for a specific **Geohash.** A geohash is a way of easily identifying a location with a specific level of precision and is used extensively by the Spot Parking platform. The more characters in the geohash, the more precise the location is. See section **Geohashes** for further information.

Each JSON document is published containing zones within a geohash with 6 characters of precision (≤ 1.22km × 0.61km) in size.

2.2. Zone Schema

The following section outlines the structure of each individual zone. To illustrate the structure clearly, TypeScript notation is used throughout this document. If you are not familiar with Typescript, then there are plenty of resource online that can assist in your understanding. In addition, a table providing further detail and explanation of the fields is provided to further facilitate comprehension of the fields and their utility.

2.2.1. Zone

The **Zone** structure is the root container for all zone information.

```
interface Zone {
    id:              string;
    paths:           Paths;
    assetType:       AssetType;
    schedule:        Schedule;
    customName?:     string;
    photos?:         string[];
    photosBaseUrl?: string;
    dynamicData?:    DynamicData;
    resourceId?:     string;
}

enum AssetType {
    Bay = "bay",
    Garage = "garage",
    OnStreet = "on-street",
    Outline = "outline",
}
```

Field	Description
id	An unique identifier for the individual zone (in UUID v4 format). *Note that this id is not guaranteed to persist across every weekly refresh.*
paths	The definition of the location where this zone exists. See **Paths** section for detailed explanation of this field.
assetType	Defines what type of asset this zone is defining. **Bay** identifies a set of one or more contiguous bays with common set of business rules. Bays are mostly related to off-street parking. **Garage** identifies a garage/multi-level parking facility

	OnStreet identifies a subset of parking on a side of a street. **Outline** identifies the boundary of parking lot.
schedule	Contains the definition of the calendar that defines what business rules apply and when in relation to the zone. See **Schedule** section for detailed explanation of this field.
photos (optional)	If the zone has associated photos/imagery, the references for these images can be found as an array of URIs.
photosBaseUrl (optional)	If there are more than one photo specified, and if the photos share the same base URL, then the base URL is specified in this field.
dynamicData (optional)	If the zone has a set of associated dynamic data, then these are specified here. See **DynamicData** section for detailed explanation of this field.
resourceId (optional)	This field is used in combination with the dynamicData field to determine the resourceId associated with the dynamic data specification. It is possible that more than one zone refers to the same resourceId (for example individual bays within a lot may refer to a resourceId that identifies the lot, not the bay). *Note the resourceId does not refer to any zone id, it is a separate identifier used by the Spot Platform to identify individual parking assets.*

For further clarity as to the different types of parking assets available, refer to Figure 1. The **Garage** type (pink square) is clearly a parking asset on its own and highlighted as such. The **Outline** type (transparent grey) usually defines the outlines of off-street parking facilities where there exist one or more related bays. **Bays** usually define parking within off-street configurations (eg. no natural relationship to the kerb). **OnStreet** zones are assets located on the side of a street, alongside a kerb.

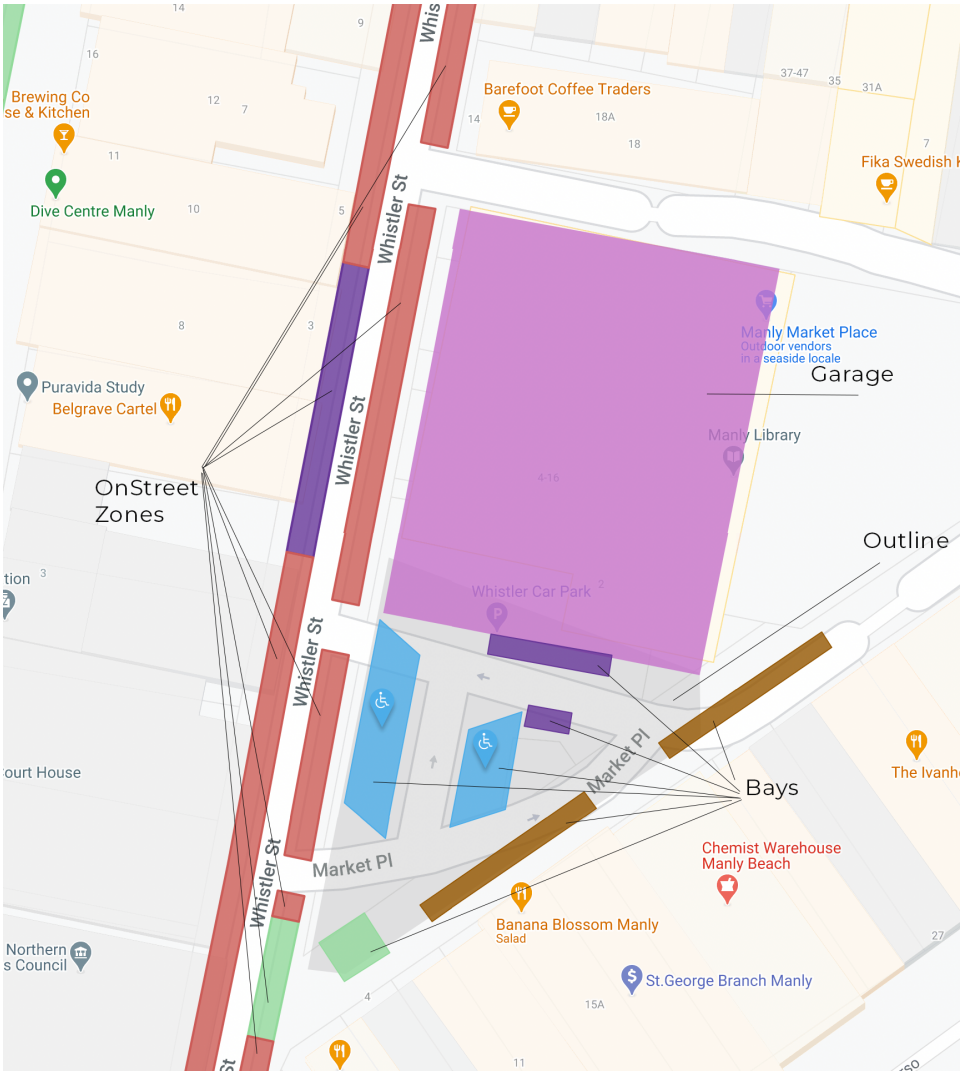


Figure 1 - Different types of Parking Assets

The main difference between **OnStreet** zones and **Bays** is in how their location information is provided. **OnStreet** zones are provided as polyline coordinates, and it is up to the individual application how best to render them (in Figure 1, a polygon object has been created directly from the polyline coordinates). In contrast, **Bays** are provided as already defined polygons. This is necessary as bays comes in a variety of sizes and configurations, and it would be impractical to represent these as polylines. An example of the variety of bays can be seen in the parking lot depicted in Figure 2. It consists of a mixture of angled and bevelled parking assets alongside standard parking bays of differing sizes.

Bays can also differ to **OnStreet** zones in that they may also have a child relationship to the **Outline** asset (eg. parking lot). This enables some commonality of features (such as tariffs) to be defined at the **Outline** level, and then shared with the related **Bay** assets. Please note that this is an advance feature which the Open Data dataset does not yet incorporate.

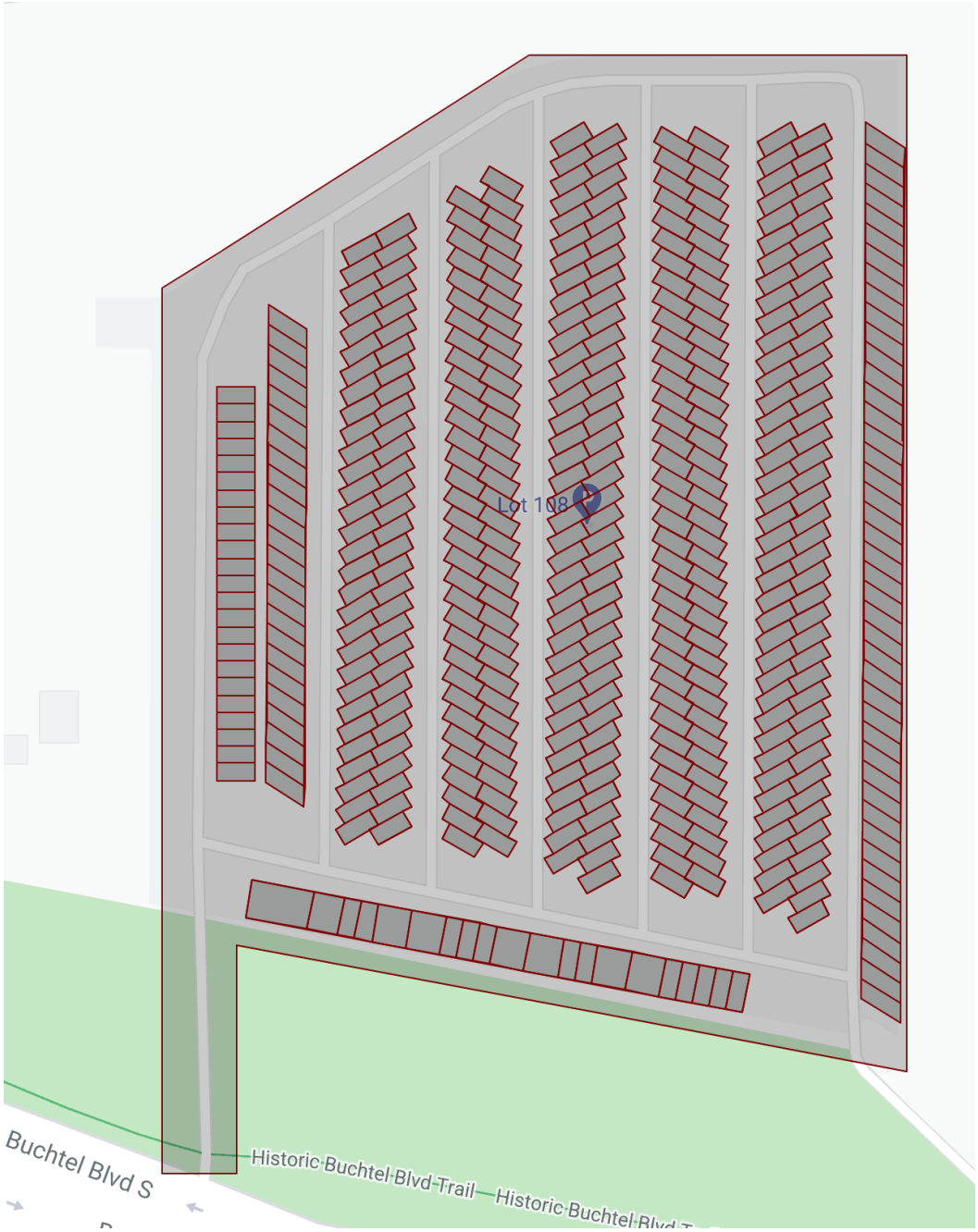


Figure 2 - Example of differing styles of Bay rendering

2.2.2. Paths

The **Paths** structure provides the location where a zone exist exists.

```
interface Paths {
    type:  PathType;
    paths: Path[];
```

```
    form:  Form;
}

enum PathType {
    Simple = "simple",
    Complex = "complex",
}

interface Path {
    latitude:  number;
    longitude: number;
}

enum Form {
    Polygon = "polygon",
    Polyline = "polyline",
}
```

Field	Description
type	Identifies the type of path involved. **Simple** paths exist with one object (either polyline or polygon). **Complex** paths consist of multiple polygons and are not supported yet within JSON Documents. *Currently there are no parking assets requiring complex paths for the available Open Data dataset.*
paths	An array of two or more coordinates that make up the definition of the zone location.
form	Identifies whether the coordinates specified in the **paths** field is of the form of a polyline, or a polygon.

2.2.3. Schedule

The **Schedule** structure defines a calendar of business rules. Use this structure to understand what category of parking this zone defines, when it applies, and what parking entitlements are available for a generic user profile.

```
interface Schedule {
    baseTimeStamp:     Date;
    interval:          number;
    totalIntervals:    number;
    intervalDstOffset: number;
    calendar:          Calendar[];
}
```

Field	Description
baseTimeStamp	Establishes the base date for which the calendar is based from. It is provided in standard UTC format. For example: "2020-09-13T14:00:00.000Z"
interval	Number of minutes within each interval. *Typically this is set at 1 (eg. 1 minute interval).*
totalIntervals	Number of interval units contained within the calendar. Used in combination with baseTimeStamp and interval fields to determine the total time range that the calendar defines. Eg. baseTimeStamp + (interval * totalIntervals) = endTimeStamp *Note usually set at 30240 to represent 3 weeks of data however this can differ if daylight saving transitions occur during the period.*
intervalDstOffset	If during the schedule period, a daylight saving transition (DST) occurs, this will show the respective offset (change) in intervals. For example, if interval = 1 and an hour is lost during the calendar period due to DST then this value will be set at -60. If an hour is gained, this will be set at 60.
calendar	An array of calendar events that define individual changes of business rules during the schedule period. See **Calendar** section for detailed explanation of this field.

2.2.4. Calendar

The **Calendar** structure defines an individual calendar event where a consistent parking business rule applies.

```
interface Calendar {
    start:          number;
    end:            number;
    startDstOffset: number;
    endDstOffset:   number;
    parkingStatus:  ParkingStatus;
    condition:      Condition;
    maxDuration?:   number;
    tariff?:        Tariff;
}

enum ParkingStatus {
    DropOffPickupOnly = "dropOffPickupOnly",
    NoParking = "noParking",
    Paid = "paid",
    Restricted = "restricted",
    Undefined = "undefined",
    Unrestricted = "unrestricted",
}
```

Field	Description
start	The starting interval when this business rule applies.
end	The ending interval when this business rule applies.
startDstOffset	If a daylight saving transition (DST) has occurred in an calendar event prior to this event, then this will show the respective offset (change) in intervals.
endDstOffset	If a daylight saving transaction (DST) has occurred either prior to this event, or during this event, then this will show the respective (offset) change in intervals. *Note if startDstOffset and endDstOffset contain different values, then the DST has occurred during the current event.*
parkingStatus	Defines the type of parking entitlement in place for the calendar event.

	DropOffPickupOnly – stopping (but no parking) is allowed with certain conditions during the calendar event **NoParking –** no parking or stopping is allowed during the calendar event **Paid** – parking is allowed for a fee with conditions during the calendar event **Restricted** – parking is allowed with conditions (eg. time-duration restricted) during the calendar event **Undefined** – no identifiable condition exists (ignore entitlement) **Unrestricted –** no restrictions during the calendar event *The **parkingStatus** field determines the 'Can I park here?' question*
condition	Defines the category of parking and any additional attributes associated with the business rule that defines the conditions for which parking can occur. There are many conditions possible. See **Condition** section for detailed explanation of this field.
maxDuration (optional)	Defines any duration limit (in minutes) imposed on parking. If this value is not defined, or equal to 0 then there are no time restrictions applied.
tariff (optional)	If the business rule involves a fee, then it is described here in this field. See **Tariff** section for detailed explanation of this field.

2.2.5. Condition

The **Condition** structure provides for the classification and if required the additional attributes that determine the business rule applied to a zone at a certain calendar event.

```
interface Condition {
    category: Category;
    <additionalAttributes>?
}

enum Category {
    BusLane = "BUS_LANE",
    BusZone = "BUS_ZONE",
    Clearway = "CLEARWAY",
    DisabledParking = "DISABLED_PARKING",
    Garage = "GARAGE",
    LoadingZone = "LOADING_ZONE",
```

```
    MailZone = "MAIL_ZONE",
    MeteredParking = "METERED_PARKING",
    MotorbikesOnly = "MOTORBIKES_ONLY",
    NoParking = "NO_PARKING",
    NoStopping = "NO_STOPPING",
    RestrictedParking = "RESTRICTED_PARKING",
    TaxiZone = "TAXI_ZONE",
    Undefined = "UNDEFINED",
    UnrestrictedParking = "UNRESTRICTED_PARKING",
    WorksZone = "WORKS_ZONE",
}
```

Field	Description
category	The classification of the parking business rule. The **Category** enumeration above only considers what is currently available for the Open Data set currently, but new categories will get defined on an ongoing basis.
<additionalAttributes> (optional)	There are wide range of additional attributes possible. An explanation of some of the more common ones can be found in the **Condition Attributes** section.

2.2.6. Tariff

The **Tariff** structure defines whether there are any fees associated with a business rule.

```
interface Tariff {
    displayCharge:          number;
    tariffType:             TariffType;
    chargeInterval?:        number;
    currency?:              Currency;
    displayChargeUnitSize?: number;
    minimumChargeUnit?:     number;
    steppedTariffs?:        SteppedTariff[];
    specialTariffs?:        SpecialTariff[];
```

```
    cappedCharge?:          number;
}

enum TariffType {
    Free = "free",
    Fixed = "fixed",
    ProRated = "proRated",
    ProRatedWithDailyCap = "proRatedWithDailyCap",
    SteppedInDuration = "steppedInDuration",
}
```

Field	Description
displayCharge	The value commonly displayed for fee information and is used as the base rate for meterage calculation.
tariffType	Defines how meterage is calculated.
chargeInterval (optional)	The length of time in minutes for which a new charge should be applied. A value of 1 indicates a unit size of 1 minute. A value of 5 indicates a unit size of 5 minutes.
currency (optional)	Currency abbreviation based on the ISO international standard – 4217. Example: 'AUD', 'USD', 'GBP', 'EUR'
displayChargeUnitSize (optional)	The number of charge intervals represented in the display charge. For example if chargeInterval = 1 (minute) and the displayCharge = $2 (per hour) then the displayChargeUnitSize = 60 (minutes)
minimumChargeUnit (optional)	The minimum number of charge_intervals that will be charged, regardless of stay duration.
steppedTariffs (optional)	An array of **SteppedTariff** conditions for when TariffType = 'SteppedInDuration'. See **SteppedTariff** section for detailed explanation of this field.
specialTariffs (optional)	An array of **SpecialTariff** conditions for when TariffType = 'SteppedInDuration'. See **SpecialTariff** section for detailed explanation of this field.
cappedCharge	For those TariffTypes with a capped component, this value represents the maximum capped charge.

2.2.7. Stepped Tariff

The **Stepped Tariff** structure describes a tariff that is imposed at a certain duration of stay. Usually found in garages where there is a series of defined fees based upon length of stay.

```
interface SteppedTariff {
    lessThanDuration?: number;
    tariffType:        TariffType;
    displayCharge:     number;
    currency?:         Currency;
    eventBased:        boolean;
    event?:            string;
}
```

Field	Description
lessThanDuration (optional)	The maximum number of minutes (in duration) for which this tariff takes effect. If not specified, this is then treated as the last stepped rule.
tariffType	Defines how meterage is calculated.
displayCharge	The value commonly displayed for fee information and is used as the base rate for meterage calculation.
currency (optional)	Currency abbreviation based on the ISO international standard – 4217. Example: 'AUD', 'USD', 'GBP', 'EUR'
eventBased	Whether the this tariff applies based upon a special event (such as ticket validated, or lost ticket)
event (optional)	The name of the event for which this applies

2.2.8. Special Tariff

The **Special Tariff** structure describes a tariff that has both an entry and exit condition (such as an Early Bird special).

```
interface SpecialTariff {
```

```
    specialName:    string;
    entryBegin:     number;
    entryEnd:       number;
    exitBegin:      number;
    exitEnd:        number;
    currency?:      Currency;
    displayCharge:  number;
    tariffType:     TariffType;
}
```

Field	Description
specialName	An user-friendly term associated with the tariff, for example 'Early Bird'
entryBegin	The time in minutes from midnight for which entry must be from
entryEnd	The time in minutes from midnight for which entry must be within
exitBegin	The time in minutes from midnight for which exit must be from
exitEnd	The time in minutes from midnight for which exit must be within
tariffType	Defines how meterage is calculated.
displayCharge	The value commonly displayed for fee information and is used as the base rate for meterage calculation.
currency (optional)	Currency abbreviation based on the ISO international standard – 4217. Example: 'AUD', 'USD', 'GBP', 'EUR'

2.2.9. Dynamic Data

The **Dynamic Data** structure defines references to additional attributes that can be retrieved from the Spot Parking platform to provide real-time information. This is often used in the context of **live occupancy** information.

To retrieve this information, the application must be registered to use the **DynamicComplex** API (see respective documentation for details).

```
interface DynamicData {
```

```
    occupancy:? Occupancy;
}

interface Occupancy {
    keys:        string[];
    refreshRate: number;
}
```

Field	Description
occupancy (optional)	If available, provides the information needed to obtain the real-time occupancy for this asset.

Field	Description
keys	One or more keys that identify the information to obtain dynamically from the server.
refreshRate	The frequency in seconds recommended between subsequent requests for information. For example a value of 120 would suggest a request for update should be made every two minutes.

3. Condition Attributes

Apart from the category field, the condition can have a wide range of optional attributes. Here is a list of some typical attributes and their meaning.

Field	Type	Description
additionalChargeForOvernightStay	boolean	Whether an additional fee is applied for overnight stay.
authorisedAusgridVehiclesOnly	boolean	Whether authorized AusGrid vehicles are permitted to park within this zone.
authorisedVehiclesOnly	boolean	Whether authorized vehicles are permitted to park within this zone.
busLane	boolean	Whether the area constitutes a bus lane during this time.
carParkClosed	boolean	Whether the car park zone is officially closed during this time.
carShareZone	array<string>	A list of approved car share providers that can use this particular zone.
councilVehiclesAllowed	boolean	Whether council vehicles are permitted to park within this zone.
dayRatesSchedule	boolean	Whether the tariff provided is the day rate.
emergencyVehiclesAllowed	boolean	Whether emergency vehicles are permitted to park within this zone.
frontToKerb	boolean	Whether parking with the vehicle with its front nearest the kerb is permitted within the zone.
funeralCarAllowed	boolean	Whether funeral cars are permitted to park within the zone.
heightRestriction	number	The height restriction (in metres) of the parking facility.
layOverLimit	number	The number of minutes the permitted vehicle is allowed to stay within the zone area.
mailVehiclesAllowed	boolean	Whether mail vehicles are permitted to park within this zone.
maxVehicleLength	number	The maximum length in meters for vehicle to park in the zone.

nightsRatesSchedule	boolean	Whether the tariff provided is the night rate.
numberOfBays	number	The number of marked bays determined within the zone.
numberOfDriveways	number	The number of standard width driveways determined within the zone. This can be used to approximate available parking spaces in areas where bays are not marked.
operator	string	The operator of the garage or car park.
overnightRatesSchedule	boolean	Whether the tariff provided is the overnight rate.
parallelParking	boolean	Whether the zone provides for parallel parking.
parkingAngle	number	The degrees of the bay angle when the zone is in AngleParking parkingStyle.
parkingStyle	string	Whether the parking is angled or not. Potential values are AngleParking or Parallel.
permitHolderExcepted	array<string>	A list of permits that have special permissions within the zone.
policeVehiclesAllowed	boolean	Whether police vehicles are permitted to park in this zone.
ratesApplyOnlyIfEntryDuringThisTime	boolean	The displayed rates only apply if the vehicle has entered the facility during this time.
rearToKerb	boolean	Whether parking with the vehicle with its rear nearest the kerb is permitted within the zone.
schoolDay	boolean	Whether the time period constitutes a school day. This is useful to understand specific business rules that apply within the zone during school periods.
schoolZone	boolean	Whether the zone is defined as a school zone (as marked with the '40' Speed signs).
telstraVehiclesAllowed	boolean	Whether Telstra vehicles are permitted to park within this zone.
totalCarSharingSpaces	number	The number of allocated car sharing spaces within the garage.
totalDisabledSpaces	number	The number of allocated disabled spaces within the garage.
totalGarageSpaces	number	The number of allocated parking spaces within the garage.

totalMotorbikeSpaces	number	The number of allocated motorbike parking spaces within the garage.
totalParentWithPramSpaces	number	The number of allocated parent with pram parking spaces within the garage.
towaway	boolean	Whether the zone is within a towaway area.
waitLimit	number	The number of minutes a car can remain stationary when the parkingType is 'dropOffOrPickUp'.
weddingCarAllowed	boolean	Whether wedding cars are permitted to park within the zone.
weekdayRatesSchedule	boolean	Whether the tariff provided is the weekday rate.
weekendRatesSchedule	boolean	Whether the tariff provided is the weekend rate.

In additional, some attributes have a "%" prefix. This indicates that the attribute should be presented in the form provided after the prefix. This is usually used for descriptive purposes only.

Some examples of this type of attribute are illustrated below:

%Overnight Stays With Occupancy Prohibited between 6pm - 6am

%Overnight Stays With Occupancy Prohibited between 10pm - 6am

%Park for free when you use your valid Opal Card to travel on public transport (First 18 hrs). Non-Opal users - Weekday day rates apply

%Park for free when you use your valid Opal Card to travel on public transport (First 18 hrs). Non-Opal users - Weekend day rates apply

4. JSON Document Access

The JSON Documents for Zones dataset is published every week to a specified end-point. Applications can acquire these documents using the process described in this section.

4.1. Base URI

The dataset can be accessed utilizing the following base URI:

```
https://s3-ap-southeast-2.amazonaws.com/open-data.distribution.spotparking.com.au
```

Applications access dataset files by combining the base URL with additional filename paths.

```
eg. <base_uri>/<filename_path>
```

4.2. Dataset Catalog

The dataset catalog is a JSON document that describes what files the dataset contains. You can access it with the following filename path:

```
catalog.json
```

The catalog schema is defined as follows:

```
interface Catalog {
    publisher:          string;
    licensedBy:         string;
    publishDate:        Date;
    numberOfGeohashes:  number;
    geohashes:          Array<string>;
    filenameFormat:     string;
    userProfile:        any;
}
```

A description of the fields within the schema:

Field	Description
publisher	The publisher of the dataset.

licensedBy	The licensee of the dataset.
publishDate	The date when the dataset was published.
numberOfGeohashes	The number of individual geohash documents published within this dataset (not including the catalog itself).
geohashes	An array of each individual geohashes that has been published under this dataset.
filenameFormat	The filename path format for which these geohash JSON documents has been published. *Note the format is specified in handlebar format. Replace the handlebar variable with the geohash of interest.*
userProfile	This is the user profile which was utilized to generate the datasets.

From this information, applications can determine programmatically which geohashes have been published and are available for them to consume.

4.3. Archival datasets

The latest dataset will always be available at the base URI. In addition, an application can use archival datasets for historical information. The archival datasets operate in a similar fashion, apart from the use of a subcategory that is specified before the file path.

```
eg. <base_uri>/<subcategory>/<filename_path>
```

The subcategory format is defined as follows:

```
<published_year_in_full>_<published_month_numeric>_W<week_of_month>
```

For example, the subcategory format for the 4th week of September 2020 would look like the following:

```
2020-9-W4
```

The catalog document for the archival dataset can be found as the following filename path:

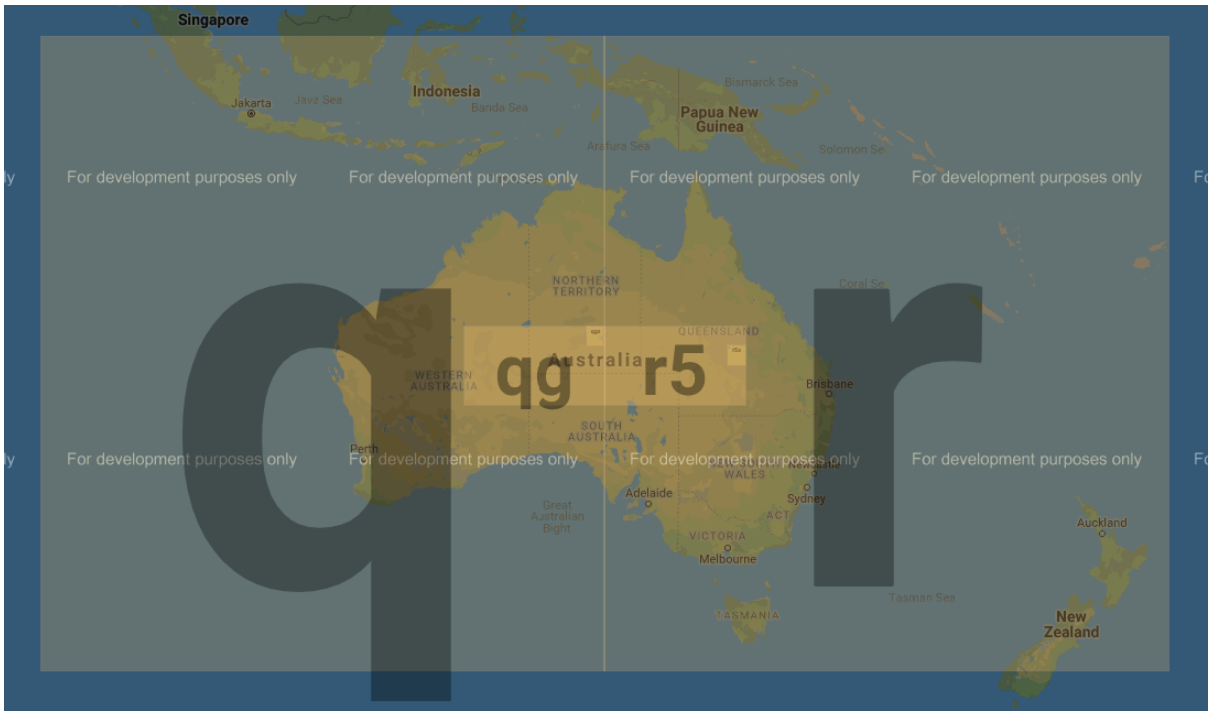
```
<subcategory>/catalog.json
```

5. Geohashes

5.1. Overview

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.

The illustration below clearly shows how geohashes works. In this case, the geohash reference 'r' refers to a coordinate somewhere within the r bounding box. By adding an additional character '5' further narrows down the size of the reference bounding box (by a factor of 1/32). Every additional character (known as a precision) further divides by a factor of 1/32). Ultimately a geohash reference with a precision of 12 (12 characters) can refer to an area roughly ≤ 3.72cm* x 1.86cm anywhere on earth (* due to the radial nature of the earth, this value decreases in size the closer the coordinate is to the pole regions).



JSON Documents for Zones separate documents based on geohashes of 6 precisions. This provides for the ability to retrieve information in a tiled format which suits most use-cases for its use. In addition, the RESTful APIs allow for precisions between 5 – 8 characters.

The respective size (dimensions) of a geohash with specific level of precisions is illustrated below:

Precision	Area Dimensions
5	≤ 4.89km × 4.89km
6 (JSON Document)	≤ 1.22km × 0.61km
7	≤ 153m × 153m
8	≤ 38.2m × 19.1m