SPOT PARKING API

Open Data Zone Groups

Version 1.4 – Document Version 1.05 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 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, 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	22-09-2020	Initial Revision	Scott Taylor
V1.05	02-10-2020	Reduced parameter support to simply usage for Open Data customers	Scott Taylor

Reference Documents

Please see the following documents for more information:

Document Name	Version	Author
Open Data API Overview	V1.01	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 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 Zone Groups API provides capabilities to determine local parking rules and entitlements within specified location(s). It caters for a variety of different query types to suit typical use-cases. It has been designed with performance-first principles – meaning it has sacrificed some initial ease-of-use for optimal efficiency. In particular, the API uses **Google Protobuf** (protobuf) responses exclusively, and the reader should be familiar with some of the basic concepts of this technology before proceeding (some recommendation references are available at the back of this document).

Throughout the document, recommended approaches for how the API should be utilized are specified, and it is highly encouraged that these are adopted. In typical usage, the Zone Groups API generates a rich, relatively large dataset of information, and it is necessary to consider how this data is best consumed, especially on resource-constrained devices.

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 Google Protocol Buffers (protobuf) format.
- Request parameters are mandatory unless explicitly marked as Optional.

1.2. Current Version

The Zone Groups 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

Due to the use of Google Protocol Buffers for responses, schemas are already formally documented within a `.proto` file. See section titled **Protobuf Specification** for a reproduction of this file.

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 in JSON format and received encoded in Google Protocol serialization format.

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**: Only POST is supported in Zone Query API requests
- **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

The Zone Query API 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>
```

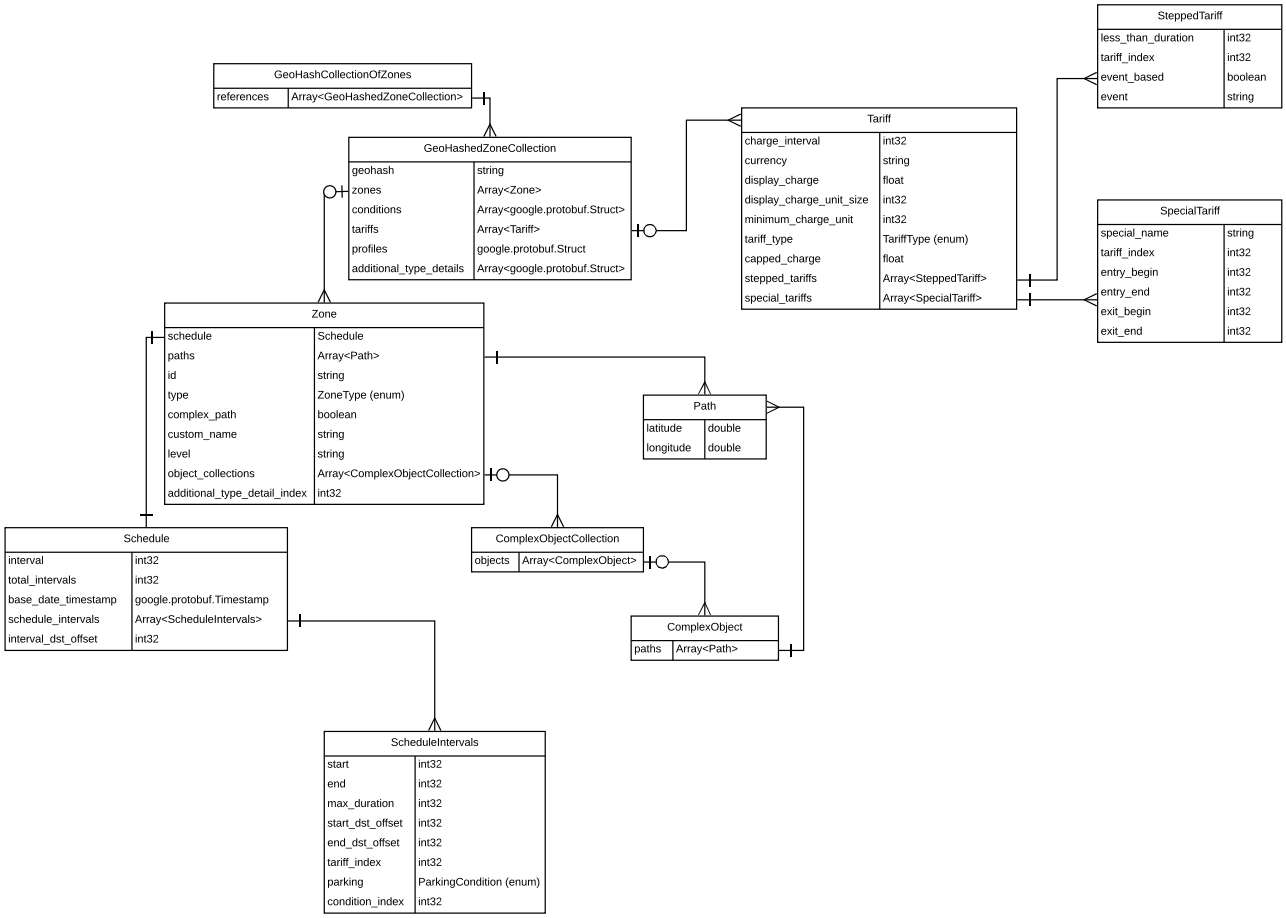
Example:

```
POST https://data-collection-api.spotparking.com.au/1.4/query/zoneGroups
```

2. Data Structures and Relationships

2.1. Schema Overview

A relationship diagram of the Zone Query API response output is as below. A full explanation of each component is contained in the section titled **Data Structures**.



2.2. Type Enum Definitions

The following section describes the type enum definitions used within the Zone Query API response schema.

2.2.1. Parking Condition

The parking condition describes the parking permission status for an individual parking zone based upon the supplied API request parameters. The same parking zone may have a different parking condition depending on the userProfile parameter provided within the request. For example, if the user specifies access to a disabled permit, then a disabled

space may have a different parking condition compared to a user who does not. The parking condition type does not specify the actual signage classification.

Value Identifier	Description
UNRESTRICTED	Non-metered Parking with no maximum time durations.
RESTRICTED	Non-metered Parking with a maximum time duration specified.
METERED	Metered Parking.
NO_PARKING	No parking permitted.
DROP_OFF_PICKUP_ONLY	Vehicle may be stationary only whilst dropping off or picking up passengers
INHERIT_TARIFF_FROM_PARENT (96)	Tariff is inherited from its parent. Used for example to apply general tariff structure for bays within garages and multi-levels
INHERIT_FROM_PARENT (97)	Business rule is inherited from its parent. Used for example to apply business rules for bays within garages and multi-levels
UNDEFINED (98)	No defined parking condition identified. (should be ignored)
INVALID (99)	Invalid parking condition identified. (should be ignored)

2.2.2. Zone Type

The zone type describes the nature of an individual zone reference. It has relevance for both client visualization and calculations.

Value Identifier	Description
NORMAL	On-street parking definition.
OUTLINE	Outline of a parking-related POI, such as a parking lot or garage.
BAY	Off-street parking definition.
BAY_POINT	Center location of an individual parking bay.
GARAGE	Garaged off-street parking facility outline
MULTI_LEVEL	Multi-level off-street parking facility outline

POI	Point of Interest
INVALID (99)	Invalid ZoneType detected. (should be ignored)

2.2.3. Tariff Type

The tariff type describes how tariff values should be calculated to determine any meterage.

Value Identifier	Description
PRO_RATED	Tariff is pro-rated without cap consideration.
PRO_RATED_WITH_DAILY_CAP	Tariff is pro-rated with a maximum daily cap.
PRO_RATED_WITH_PERIOD_CAP	Tariff is pro-rated with a schedule interval maximum cap.
FIXED	Tariff is a fixed fee regardless of stay.
FREE	No tariff.
STEPPED_IN_DURATION	Tariff structure is complex consisting of different tariff charges based on duration of stay.
SPECIAL_TARIFF	Tariffs such as special event tariffs (early bird specials etc).

2.3. Data Structures

The following section describes the individual data structures used within the Zone Groups API response.

2.3.1. GeoHashCollectionOfZones

The **GeoHashCollectionOfZones** is the master (root) container for all Zone Groups API responses. It contains a singular field containing a list of **GeoHashedZoneCollection** instances.

Field	Type	Description
references	Array<GeoHashedZoneCollection>	Contains a list of individual geohash zone collections.

2.3.2. GeoHashedZoneCollection

A **GeoHashedZoneCollection** contains all of the parking zone information for a particular geohash reference. It is best considered like as a miniature database, containing a list of

zones with associated lookup references to common conditions and tariffs. It also includes profile information that self-describes the various user profile information that may affect the data output for this particular geohash collection in subsequent requests.

Field	Type	Description
geohash	String. *See **Geohash** section for further information on Geohashes.*	The geohash for the collection of parking zone information.
zones	Array<Zone>	A list of 0..*n* parking zone definitions contained within the area referenced by the geohash.

conditions	Array<google.protobuf.Struct>
	
	A condition is a JSON object that describes the characteristics of the dominant business rule affecting the parking zone.
	
	There is only one mandatory attribute within the JSON object called 'category'. This refers to the business rule set. Additional attributes may also be present.
		A list of 0..n unique conditions found in parking zones within the area referenced by the geohash.
		
		Example condition object:
		`{`
		`"category": "NO_PARKING",`
		`"areaPermitExcepted": 20,`
		`"towaway": true`
		`}`

tariffs	Array<Tariff>	A list of 0..n unique tariff definitions found in parking zones within the area referenced by the geohash.

profiles	google.protobuf.Struct
	
	A profile is a JSON object that describes the types of userProfile attributes that could generate alternative zone information for this geohash.
	This can be seen as a self-describing field in that only those attributes that affect zone information output are documented.
		Example profile object:
		`{`
		`"disabledPermit": true,`
		`"areaPermit": ["20"],`
		`"_vehicleType": ["TAXI", "BUS"]`
		`}`
		Note: those attributes prefixed with an "_" indicate an attribute which should be considered mutually exclusive. ie. A user profile should only request one of the listed values, not multiple.

additionalTypeDetails	Array<google.protobuf.Struct> An additional type detail is a JSON object that provides additional metadata associated with the	A list of 0..n unique additional type details found in parking zones within the area referenced by the geohash.

	parking asset. This can provide additional information regarding relationships between different parking assets (eg. parent-child relationships) or custom naming information etc.	

Geohash References Only Designations

For some zone types (**outlines**), the API may return 2 GeohashedZoneCollection instances for the same geohash reference. In this case, the second geohash reference key is structured in the form `<geohash>_referenceOnly`. Because zone outlines could extend across multiple geohash areas, if a part of an outline is contained within the geohash area it is provided as a reference only. Geohash reference only designations are useful only for visualization needs (where an outline may be slightly visible within the current map bounds but should still be displayed).

2.3.3. Zone

A **Zone** contains all the information necessary to determine the specifics of an individual parking zone, including its geometry. A zone has various types, the most common being on-street parking (normal), parking POIs (outlines) or off-street parking bays (bays).

Field	Type	Description
schedule	Schedule	A zone has a schedule which dictates when what business rules apply at certain times. See **Schedule** section for more information.
paths	Array<Path>	A zone must have relevance only within a defined geographic area. For on-street parking zones, this is defined by a list of 2 or more consecutive coordinates that combine to construct a polyline. In the case of off-street parking zones, this is defined by a list of 2 or more consecutive coordinates that combine to construct a polygon. For outlines, if the zone consists of one simple polygon, this may be represented as a list of 2 or more consecutive coordinates that combine to construct a polygon. For outlines which consist of multiple

		polygons, the **object_collections** field is used instead.
id	String representing a GUID of the particular zone instance.	A unique identifier for parking zone information that can be used for referencing a particular zone instance within a collection of zones.
type	ZoneType (enum)	Identifies the type of zone information being represented (normal, outline, bay)
complex_path	Boolean	A flag indicating whether path information is contained within the **object_collections** field rather than in the **paths** field.
custom_name	String	If the zone has an unique reference (such as a name of a Garage) then it may be contained here.
level	String	If the zone information pertains to a specific level of a multiple-level parking facility, then it is identified here.
object_collections	Array<ComplexObjectCollection>	For zones that require complex rendering, path information is contained here. This is a list of 0..*n* ComplexObjectCollection, each of which represents a unique polygon object.
additionalTypeDetailIndex	Int32	Refers to additional type detail information contained within the **additionalTypeDetails** field of the parent GeoHashedZoneCollection.

2.3.4. Schedule

A **Schedule** represents what business rules apply when within a parking zone. Although the Spot Parking platform handles multiple schedules per parking zone (to accommodate different user profile parameters), only one is returned per zone within a Zone Query API response.

Field	Type	Description
interval	Int32	An interval defines the unit of time (in minutes) for each unit of the schedule. For example, an interval of 1 indicates time units

		are provided with a granularity of 1 minute. An interval of 5 would indicate 5 minutes. Spot currently generates schedules with an interval setting of 1.
total_intervals	Int32	The total number of intervals contained within the schedule. To determine the total number of minutes covered by a schedule, calculate: `interval * total_intervals = total_minutes`
base_date_timestamp	google.protobuf.Timestamp	The base date for where time intervals is calculated from. To calculate the specific time represented by a time interval, calculate (for example *start*): `base_date_timestamp + (start * interval) minutes = interval_start_time`
schedule_intervals	Array<ScheduleIntervals>	A list of 1..*n* defined interval ranges where certain business rules apply.
Interval_dst_offset	Int32	If a daylight savings change occurs during the duration of the schedule, then this field will indicate the offset in minutes. For example, if the schedule lost an hour, then this value will be -60. If the schedule gained an hour, this value will be 60.

2.3.5. Schedule Intervals

A **Schedule Intervals** instance contains information regarding a specific business rule setting for a defined time period (represented as a range of intervals).

Field	Type	Description
start	Int32	The start interval unit for when the business rules applies. To determine number of minutes from base date use the following calculation: `start * interval`
end	Int32	The end interval unit for when the business rules applies. To determine the number of minutes from base date use the following calculation: `end * interval`

max_duration	Int32	The maximum duration for a stay applying to the specified business rule in minutes. For example, for 2 hours, this would be represented as 120. If no maximum duration is specified, then this value will be 0.
start_dst_offset	Int32	Represents the offset of any DST change occurring before the start of this particular schedule interval, but after the schedule base date.
end_dst_offset	Int32	Represents the offset of any DST change occurring before the end of this particular schedule interval, but after the schedule base date. If start_dst_offset and end_dst_offset are different values, then the DST change has occurred during the schedule interval.
tariff_index	Int32	Refers to tariff information contained within the **tariffs** field of the parent GeoHashedZoneCollection instance.
parking	ParkingCondition (enum)	Represents the parking permission status associated with the business rule and the requested user profile.
condition_index	Int32	Refers to condition information contained within the **conditions** field of the parent GeoHashedZoneCollection.

To determine the ScheduleIntervals instance associated with a date and time of enquiry:

1) Determine the enquiry interval value

Enquiry interval = Math.Floor((Minutes between Enquiry Timestamp and Base Date Timestamp) / Schedule.interval)

2) Search for ScheduleIntervals instance

If (Enquiry interval >= ScheduleIntervals.start) && (Enquiry interval < ScheduleIntervals.end)

2.3.6. Path

The **Path** represents an atomic latitude and longitude coordinate. It is used in multiple consecutive pairs to identify geographical areas where parking zone information applies.

Field	Type	Description
latitude	Double	The latitude coordinate of a coordinate pair represented in decimal notation.
longitude	Double	The longitude coordinate of a coordinate pair represented in decimal notation.

2.3.7. Complex Object Collection

A **Complex Object Collection** instances represents a complex polygon object with one or more polygon paths. This is useful for zone outlines types which have inner holes within their main polygon.

Field	Type	Description
objects	Array<ComplexObject>	A list of **ComplexObject** instances. The first instance within the list represents the outer polygon layer. Additional instances within the list represent inner polygons that generate internal holes within the outer polygon layer.

2.3.8. Complex Object

A **Complex Object** represents a list of coordinates that together represent a polygon object.

Field	Type	Description
paths	Array<Path>	A list of **Path** instances that together represent a polygon of consecutive GPS coordinates.

2.3.9. Tariff

A **Tariff** instance represents rating information in relation to meterage.

Field	Type	Description
charge_interval	Int32	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	String	Currency abbreviation based on the ISO international standard – 4217. Example: 'AUD', 'USD', 'GBP', 'EUR'
display_charge	Float	The value commonly displayed for fee information and is used as the base rate for meterage calculation.
display_charge_unit_size	Int32	The number of charge intervals represented in the display charge. For example, if charge_interval = 1 (minute) and the display_charge = $2 (per hour) then the display_charge_unit_size = 60 (minutes).
minimum_charge_unit	Int32	The minimum number of charge intervals that will be charged, regardless of stay duration.
tariff_type	TariffType (enum)	Defines how meterage is calculated.
capped_charge	Float	For those TariffTypes with a capped component, this value represents the maximum capped charge.
stepped_tariffs	Array<SteppedTariff>	List of Stepped Tariff conditions for complex STEPPED_IN_DURATION tariffs.
special_tariffs	Array<SpecialTariff>	List of Special Tariff conditions for complex STEPPED_IN_DURATION tariffs.

For example, for an area with $4.50 per hour charged pro-rata by minute, the display_charge would be 4.5, display_charge_unit_size would be 60, and charge_interval would be 1.

2.3.10. Stepped Tariff Object

A **Stepped Tariff Object** represents a stepped tariff condition that makes up a complex STEPPED_IN_DURATION tariff

Field	Type	Description
less_than_duration	Int32	The maximum number of minutes (in duration) for which this tariff takes effect.
tariff_index	Int32	Refers to tariff information contained within the **tariffs** field of the parent GeoHashedZoneCollection instance.

event_based	Boolean	Whether this tariff applies based upon a special event (such as ticket validated).
event	String	The name of the event for which this applies.

2.3.1. Special Tariff Object

A **Special Tariff Object** represents a special tariff condition that makes up part of a complex STEPPED_IN_DURATION tariff. An example would be an early bird special. Special Tariffs may apply only if special conditions are met – eg. entry between a time range and exit between a time range.

Field	Type	Description
special_name	String	A user-friendly string associated with the tariff.
tariff_index	Int32	Refers to tariff information contained within the **tariffs** field of the parent GeoHashedZoneCollection instance.
entry_begin	Int32	The time in minutes from midnight for which entry must be from.
entry_end	Int32	The time in minutes from midnight for which entry must be within.
exit_begin	Int32	The time in minutes from midnight for which exit must be from.
exit_end	Int32	The time in minutes from midnight for which exit must be within.

3. API Reference Documentation

3.1. Overview

The Zone Groups API is a singular API resource that serves multiple purposes. Its flexibility is contained within its request payload, which is designed to support a multitude of complex queries.

Note that the term method and resource (and object) tend to be used interchangeable in API documentation.

Method	Purpose
Query	Generates a list of applicable parking zones

4. Query

Given a set of query parameters, returns detailed information about parking zones that meet the requested criteria.

4.1. Resource Information

The Zone Groups API Query resource information is as follows:

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

4.2. Request

The Zone Groups API Query resource request information is as follows:

Method	URL
POST	https://data-collection-api.spotparking.com.au/1.4/query/zoneGroups

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

4.3. Headers

The Zone Groups API Query 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 (provided by Spot Parking)	Example: fHoX5l4bo22Xvv7n5dQDaFf7p
Content-Type	Request Body Type Information	application/json

4.1. Parameters

The Zone Groups API 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
geohashes	An array of *Geohashes* (string format) to retrieve information for.	A list of geohashes to retrieve parking information for. All contained geohashes must be the same precision level. Level of precision supported is between 5 – 8 geohash characters. See section **Geohashes** for further background on geohashes.	Mandatory
userProfile	JSON structure containing user profile attributes.	Provides user context into the query. For example, to indicate access to a permit type (disabled, resident etc), type of vehicle license or vehicle type. (See subsection **Request Parameter Examples** for examples) The available profiles supported in queries are always passed back in Zone Groups API responses within the field *GeoHashedZoneCollection->profiles*	Optional (defaults to empty profile)

4.1.1. Request Parameter Examples

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

```
{
  "geohashes": ["r3gx27","r3gx26","r3gx23","r3gx2e","r3gx2d","r3gx29"],
}
```

The most typical usage. Retrieves **all parking zone** information within the areas defined by the group of specified geohashes.

By default, using geohashes with 6-character will provide the fastest system response as they are automatically pre-cached upon system generation.

```
{
  "geohashes": ["r3gx27","r3gx26","r3gx23","r3gx2e","r3gx2d","r3gx29"],
  "userProfile":{},
```

```
}
```

Retrieves all parking zone information within the areas defined by the group of specified geohashes. Note: the userProfile being passed is an empty object – this is equivalent to not providing a userProfile parameter.

```
{
  "geohashes": ["r3gx27”],
  "userProfile":{
    “disabledPermit”: true
  }
}
```

Retrieves all parking zone information within the area defined by the specified geohash. Note: the userProfile being passed has a parameter **disabledPermit** which is set to true. The entitlements of holders with disabled parking permits will be provided within the response. Available user profile parameters for that area are always provided back in every zoneGroup query response.

4.2. Response

The Zone Group API Query provides a response in Google Protocol Buffers format, according to the structure documented in the section **Data Structures and Relationships**. The API will provide the response with a Content-Type HTTP-Header value of `application/octet-stream`. Depending on the language this may be represented as a Buffer or an array of bytes. To deserialise the data into an usable form, use the provided client library **decode** function.

Please refer to the **Spot Protocol Buffer Services** for information about Spot Parking’s own client libraries supporting the most popular languages.

The advantage of Google Protocol buffers is that the respective client libraries already fully understand the underlying data structures and their relationships, and the decoding function transforms the API responses into well-formed types, ready for use. They are also written natively, thus take full advantage of the processing capability.

4.3. Examples

Example use of Spot’s Javascript / Node.js library, showing the fetching of parking zone information from the backend, and the deserialization of the data into usable objects, ready for use.

```
const fetch = require('fetch-retry');
import { spotparking } from "../server/protobuf/v1.4/javascript/spotparking_v1_4";
const geohashes = ["r3grgb", “r3grez”];

async function start() {
```

```
  try {
    let results = await retrieveScheduleInformation(geohashes);
    let geohashCollectionOfZones = spotparking.GeoHashCollectionOfZones.decode(results);

    geohashCollectionOfZones.references.forEach(collection => {
       . . . . . .
    });

    process.exit(0);
  }

  catch (error) {
    console.error("An error has occurred: ", error);
    process.exit(1);
  }
}

async function retrieveScheduleInformation(geohashes: string[]): Promise<Buffer> {
  return new Promise<Buffer> ( async (resolve, reject) => {
    try {
      if ((!isArray(geohashes)) || (isEmpty(geohashes)) {
        return reject("No geohashes provided");
      }

      let path = "https://data-collection-api.spotparking.com.au/1.4/query/client/zoneGroups";

      let headers = {
        "Content-Type": "application/json",
        "X-Api-Token": "<Token-Provided>"
      };

      let parameters = {
        geohashes,
        userProfile: {}
      };

      let results = await fetch(path, {
        method: 'POST',
        body: JSON.stringify(parameters),
        headers,
        retries: 10,
```

```
        retryDelay: 200,
        retryOn: [504]
      });

      resolve(results.buffer());
    }

    catch (error) {
      reject(error);
    }
  });
}

start();
```

5. Status Codes

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

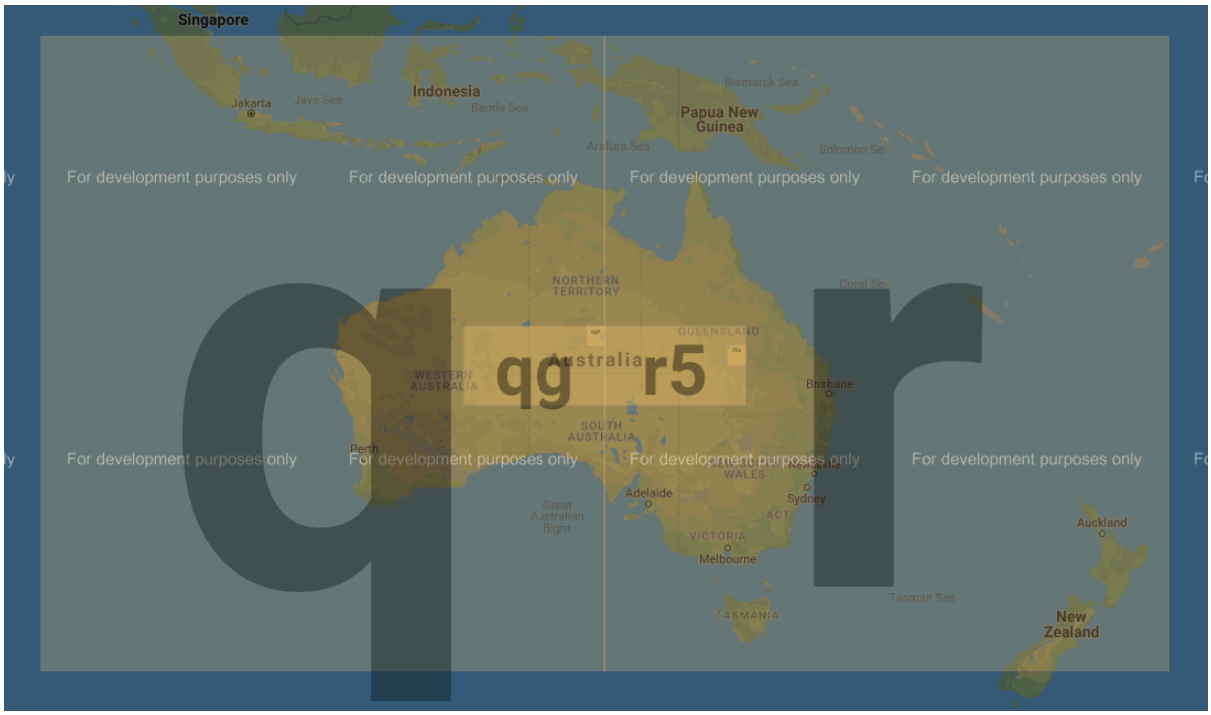
Status Code	Description
200	OK
201	Created
202	Accepted (Request accepted, and queued for execution)
400	Bad request
401	Authentication failure
403	Forbidden
404	Resource not found
405	Method Not Allowed
409	Conflict
412	Precondition Failed
413	Request Entity Too Large
500	Internal Server Error
501	Not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

6. Geohashes

6.1. 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 an 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).



Zone group queries can handle requests for geohashes between 5 – 8 characters in precision. This provides for the ability to retrieve information in a tiled format which suits most use-cases for its use. Typically, for most general purposes such as for map

visualisation, providing geohashes with 6 character precisions are recommended. The relative size of the supported precisions are as follows:

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

The following illustration shows the area size of queries possible with 5, 6 and 7 (small box within *r3gx2d* box) character precision. Remember also that zoneGroup queries support the retrieval of information for multiple geohashes within the same request.

Because geohashes group all information within the same area together, they make excellent tile-based references, as well as convenient hashable keys for implementing caching. The structure of the zoneGroup API response is designed specifically with this in mind.

To generate geohashes from standard latitude-longitude coordinates, there are a multitude of open-sourced libraries available in various languages – some of which are listed below:

Language	Reference
C	https://github.com/simplegeo/libgeohash
C#	https://github.com/alexframe/GeoHash.Net
Go	https://godoc.org/github.com/mmcloughlin/geohash
Java / Android	Internally developed by Spot and available upon request
Javascript / Node.js	https://github.com/chrisveness/latlon-geohash
Objective-C	https://github.com/lyokato/objc-geohash
PHP	https://github.com/skthon/geohash
Python	https://github.com/hkwi/python-geohash/wiki
Ruby	https://github.com/davetroy/geohash
Rust	https://github.com/georust/geohash
Swift	Internally developed by Spot and available upon request

7. Spot Protocol Buffer Services

7.1. Overview

Spot provides standard client libraries for the most common languages (see below) upon request. As these are all native libraries, deserialization is extremely fast and the incoming data is transformed into well-formed, well-typed objects, ready for use.

Language
C++
C#
Go
Java
Javascript / Node.js
Objective-C
PHP
Python
Ruby
Rust
Swift

8. Protobuf Specification

8.1. Spot Protobuf Specification

The following is a replication of the `spotparking_v1_4.proto` file used to generate responses for this API specification. Note: There are references to two additional `.proto` files – these refer to standard Google Protocol Buffer extension libraries that represent loosely defined structures (like JSON objects) and timestamps (date reference). For ease of reference these are also provided in full in subsequent sections.

```
syntax = "proto3";

import "google/protobuf/struct.proto";

import "google/protobuf/timestamp.proto";


// spotparking.proto

package spotparking;


message GeoHashCollectionOfZones {

    repeated GeoHashedZoneCollection references = 1;

}


message GeoHashedZoneCollection {

    string geohash = 1;

    repeated Zone zones = 2;

    repeated google.protobuf.Struct conditions = 3;

    repeated Tariff tariffs = 4;

    google.protobuf.Struct profiles = 5;

    repeated google.protobuf.Struct additionalTypeDetails = 6;

}


enum ZoneType {

  NORMAL = 0;

  OUTLINE = 1;

  BAY = 2;

  BAY_POINT = 3;

  GARAGE = 4;

  MULTI_LEVEL = 5;

  POI = 6;
```

```
  INVALID = 99;
}

message Zone {
    Schedule schedule = 1;
    repeated Path paths = 2;
    string id = 3;
    ZoneType type = 4;
    bool complex_path = 5;
    string custom_name = 6;
    string level = 7;
    repeated ComplexObjectCollection object_collections = 8;
    int32 additionalTypeDetailIndex = 9;
}

message ComplexObjectCollection {
  repeated ComplexObject objects = 1;
}

message ComplexObject {
  repeated Path paths = 1;
}

message Schedule {
    int32 interval = 1;
    int32 total_intervals = 2;
    google.protobuf.Timestamp base_date_timestamp = 3;
    repeated ScheduleIntervals schedule_intervals = 4;
    int32 interval_dst_offset = 5;
}

message Path {
    double latitude = 1;
    double longitude = 2;
}
```

```
message ScheduleIntervals {
    int32 start = 1;
    int32 end = 2;
    int32 max_duration = 3;
    int32 start_dst_offset = 4;
    int32 end_dst_offset = 5;
    int32 tariff_index = 6;
    enum ParkingCondition {
        UNRESTRICTED = 0;
        RESTRICTED = 1;
        METERED = 2;
        NO_PARKING = 3;
        DROP_OFF_PICKUP_ONLY = 4;
        INHERIT_TARIFF_FROM_PARENT = 96;
        INHERIT_FROM_PARENT = 97;
        UNDEFINED = 98;
        INVALID = 99;
    }
    ParkingCondition parking = 7;
    uint32 condition_index = 8;
}

enum TariffType {
  PRO_RATED = 0;
  PRO_RATED_WITH_DAILY_CAP = 1;
  PRO_RATED_WITH_PERIOD_CAP = 2;
  FIXED = 3;
  FREE = 4;
  STEPPED_IN_DURATION = 5;
  SPECIAL_TARIFF = 6;
}

message Tariff {
    int32 charge_interval = 1;
    string currency = 2;
```

```
    float display_charge = 3;
    int32 display_charge_unit_size = 4;
    int32 minimum_charge_unit = 5;
    TariffType tariff_type = 6;
    float capped_charge = 7;
    repeated SteppedTariff stepped_tariffs = 8;
    repeated SpecialTariff special_tariffs = 9;
}

message SteppedTariff {
  int32 less_than_duration = 1;
  int32 tariff_index = 2;
  bool event_based = 3;
  string event = 4;
}

message SpecialTariff {
  string special_name = 1;
  int32 tariff_index = 2;
  int32 entry_begin = 3;
  int32 entry_end = 4;
  int32 exit_begin = 5;
  int32 exit_end = 6;
}
```

8.2. Struct Protobuf Extension Specification

To aid in reader understanding, the `struct` Google Protobuf Extension specification is provided in full. Refer to the original source at https://github.com/protocolbuffers/protobuf/blob/master/src/google/protobuf/struct.proto.

```
// Protocol Buffers - Google's data interchange format
// Copyright 2008 Google Inc.  All rights reserved.
// https://developers.google.com/protocol-buffers/
//
```

```
// Redistribution and use in source and binary forms, with or without
// modification, are permitted provided that the following conditions are
// met:
//
//     * Redistributions of source code must retain the above copyright
// notice, this list of conditions and the following disclaimer.
//     * Redistributions in binary form must reproduce the above
// copyright notice, this list of conditions and the following disclaimer
// in the documentation and/or other materials provided with the
// distribution.
//     * Neither the name of Google Inc. nor the names of its
// contributors may be used to endorse or promote products derived from
// this software without specific prior written permission.
//
// THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
// "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
// LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR
// A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT
// OWNER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
// SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT
// LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE,
// DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY
// THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
// (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE
// OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

syntax = "proto3";

package google.protobuf;

option csharp_namespace = "Google.Protobuf.WellKnownTypes";
option cc_enable_arenas = true;
option go_package = "github.com/golang/protobuf/ptypes/struct;structpb";
option java_package = "com.google.protobuf";
option java_outer_classname = "StructProto";
option java_multiple_files = true;
```

```
option objc_class_prefix = "GPB";


// `Struct` represents a structured data value, consisting of fields
// which map to dynamically typed values. In some languages, `Struct`
// might be supported by a native representation. For example, in
// scripting languages like JS a struct is represented as an
// object. The details of that representation are described together
// with the proto support for the language.
//
// The JSON representation for `Struct` is JSON object.
message Struct {
  // Unordered map of dynamically typed values.
  map<string, Value> fields = 1;
}

// `Value` represents a dynamically typed value which can be either
// null, a number, a string, a boolean, a recursive struct value, or a
// list of values. A producer of value is expected to set one of that
// variants, absence of any variant indicates an error.
//
// The JSON representation for `Value` is JSON value.
message Value {
  // The kind of value.
  oneof kind {
    // Represents a null value.
    NullValue null_value = 1;
    // Represents a double value.
    double number_value = 2;
    // Represents a string value.
    string string_value = 3;
    // Represents a boolean value.
    bool bool_value = 4;
    // Represents a structured value.
    Struct struct_value = 5;
    // Represents a repeated `Value`.
```

```
    ListValue list_value = 6;
  }
}

// `NullValue` is a singleton enumeration to represent the null value for the
// `Value` type union.
//
//  The JSON representation for `NullValue` is JSON `null`.
enum NullValue {
  // Null value.
  NULL_VALUE = 0;
}

// `ListValue` is a wrapper around a repeated field of values.
//
// The JSON representation for `ListValue` is JSON array.
message ListValue {
  // Repeated field of dynamically typed values.
  repeated Value values = 1;
}
```

8.3. Timestamp Protobuf Extension Specification

To aid in reader understanding, the `timestamp` Google Protobuf Extension specification is provided in full. Refer to the original source at https://github.com/protocolbuffers/protobuf/blob/master/src/google/protobuf/timestamp.proto.

```
// Protocol Buffers - Google's data interchange format
// Copyright 2008 Google Inc.  All rights reserved.
// https://developers.google.com/protocol-buffers/
//
// Redistribution and use in source and binary forms, with or without
// modification, are permitted provided that the following conditions are
// met:
//
//     * Redistributions of source code must retain the above copyright
```

```
// notice, this list of conditions and the following disclaimer.
//     * Redistributions in binary form must reproduce the above
// copyright notice, this list of conditions and the following disclaimer
// in the documentation and/or other materials provided with the
// distribution.
//     * Neither the name of Google Inc. nor the names of its
// contributors may be used to endorse or promote products derived from
// this software without specific prior written permission.
//
// THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
// "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
// LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR
// A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT
// OWNER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
// SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT
// LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE,
// DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY
// THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
// (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE
// OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

syntax = "proto3";

package google.protobuf;

option csharp_namespace = "Google.Protobuf.WellKnownTypes";
option cc_enable_arenas = true;
option go_package = "github.com/golang/protobuf/ptypes/timestamp";
option java_package = "com.google.protobuf";
option java_outer_classname = "TimestampProto";
option java_multiple_files = true;
option objc_class_prefix = "GPB";

// A Timestamp represents a point in time independent of any time zone or local
// calendar, encoded as a count of seconds and fractions of seconds at
// nanosecond resolution. The count is relative to an epoch at UTC midnight on
```

```
// January 1, 1970, in the proleptic Gregorian calendar which extends the
// Gregorian calendar backwards to year one.
//
// All minutes are 60 seconds long. Leap seconds are "smeared" so that no leap
// second table is needed for interpretation, using a [24-hour linear
// smear](https://developers.google.com/time/smear).
//
// The range is from 0001-01-01T00:00:00Z to 9999-12-31T23:59:59.999999999Z. By
// restricting to that range, we ensure that we can convert to and from [RFC
// 3339](https://www.ietf.org/rfc/rfc3339.txt) date strings.
//
// # Examples
//
// Example 1: Compute Timestamp from POSIX `time()`.
//
//     Timestamp timestamp;
//     timestamp.set_seconds(time(NULL));
//     timestamp.set_nanos(0);
//
// Example 2: Compute Timestamp from POSIX `gettimeofday()`.
//
//     struct timeval tv;
//     gettimeofday(&tv, NULL);
//
//     Timestamp timestamp;
//     timestamp.set_seconds(tv.tv_sec);
//     timestamp.set_nanos(tv.tv_usec * 1000);
//
// Example 3: Compute Timestamp from Win32 `GetSystemTimeAsFileTime()`.
//
//     FILETIME ft;
//     GetSystemTimeAsFileTime(&ft);
//     UINT64 ticks = (((UINT64)ft.dwHighDateTime) << 32) | ft.dwLowDateTime;
//
//     // A Windows tick is 100 nanoseconds. Windows epoch 1601-01-01T00:00:00Z
//     // is 11644473600 seconds before Unix epoch 1970-01-01T00:00:00Z.
```

```
//     Timestamp timestamp;
//     timestamp.set_seconds((INT64) ((ticks / 10000000) - 11644473600LL));
//     timestamp.set_nanos((INT32) ((ticks % 10000000) * 100));
//
// Example 4: Compute Timestamp from Java `System.currentTimeMillis()`.
//
//     long millis = System.currentTimeMillis();
//
//     Timestamp timestamp = Timestamp.newBuilder().setSeconds(millis / 1000)
//         .setNanos((int) ((millis % 1000) * 1000000)).build();
//
//
// Example 5: Compute Timestamp from current time in Python.
//
//     timestamp = Timestamp()
//     timestamp.GetCurrentTime()
//
// # JSON Mapping
//
// In JSON format, the Timestamp type is encoded as a string in the
// [RFC 3339](https://www.ietf.org/rfc/rfc3339.txt) format. That is, the
// format is "{year}-{month}-{day}T{hour}:{min}:{sec}[.{frac_sec}]Z"
// where {year} is always expressed using four digits while {month}, {day},
// {hour}, {min}, and {sec} are zero-padded to two digits each. The fractional
// seconds, which can go up to 9 digits (i.e. up to 1 nanosecond resolution),
// are optional. The "Z" suffix indicates the timezone ("UTC"); the timezone
// is required. A proto3 JSON serializer should always use UTC (as indicated by
// "Z") when printing the Timestamp type and a proto3 JSON parser should be
// able to accept both UTC and other timezones (as indicated by an offset).
//
// For example, "2017-01-15T01:30:15.01Z" encodes 15.01 seconds past
// 01:30 UTC on January 15, 2017.
//
// In JavaScript, one can convert a Date object to this format using the
// standard [toISOString()](https://developer.mozilla.org/en-
US/docs/Web/JavaScript/Reference/Global_Objects/Date/toISOString]
```

```
// method. In Python, a standard `datetime.datetime` object can be converted

// to this format using
[`strftime`](https://docs.python.org/2/library/time.html#time.strftime)

// with the time format spec '%Y-%m-%dT%H:%M:%S.%fZ'. Likewise, in Java, one

// can use the Joda Time's [`ISODateTimeFormat.dateTime()`](

// http://www.joda.org/joda-
time/apidocs/org/joda/time/format/ISODateTimeFormat.html#dateTime%2D%2D

// ) to obtain a formatter capable of generating timestamps in this format.

//

//

message Timestamp {


  // Represents seconds of UTC time since Unix epoch

  // 1970-01-01T00:00:00Z. Must be from 0001-01-01T00:00:00Z to

  // 9999-12-31T23:59:59Z inclusive.

  int64 seconds = 1;


  // Non-negative fractions of a second at nanosecond resolution. Negative

  // second values with fractions must still have non-negative nanos values

  // that count forward in time. Must be from 0 to 999,999,999

  // inclusive.

  int32 nanos = 2;

}
```

9. Google Protocol Buffers references

The following is a list of excellent Google Protocol (protobuf) references for background information

Reference
Protocol Buffers https://developers.google.com/protocol-buffers/
Google's Data Interchange Format https://opensource.google.com/projects/protobuf