SUMD RFP Data Requirements and Specifications

General Requirements

1. Vendor shall be required to provide data on the entire fleet operating in the Metropolitan Government of Nashville & Davidson County service area.
2. The Metropolitan Government is able to publish real-time SUMD availability data to the public.
3. Vendor consents and agrees the Metropolitan Government of Nashville & Davidson County owns all data collected from Vendors. Metropolitan Government of Nashville & Davidson County will share anonymized data as it deems necessary to include public APIs.
4. Vendor shall include in terms of use that customers also consent that trip data will be shared with the Metropolitan Government and become public record. (All data will be anonymized).
5. Vendor shall support both Agency and Provider API consistent with version 0.3 of the Mobility Data Specification.
6. Vendor will be required to support updated releases on Metropolitan Government of Nashville & Davidson County's schedule.
    a. The ITS Department of Metropolitan Government of Nashville & Davidson County will be the Technology contact for all data related issues.
    b. Vendors will be required to support updated releases within 30 days of notification from the ITS Department of Metropolitan Government of Nashville & Davidson County.
    c. Vendor will be required to support the current version until the upgrade has been tested and accepted by Metropolitan Government of Nashville & Davidson County.
7. Vendor shall be required to provide anonymized real-time trip and telemetry information with GPS coordinates collected at an interval not exceeding 30 seconds with an accuracy of at least 5 decimal places for the entire trip. Intervals of less than 30 seconds are preferred.
8. Vendor will provide all status changes of all vehicles in the service area regardless time duration or trip distance.
9. Vendor will Implement and enforce geo-fenced parking, "no-ride", and "slow-ride" restricted areas.
    a. Metropolitan Government of Nashville & Davidson County will provide GIS information for these areas.
    b. Metropolitan Government of Nashville & Davidson County may add, change and delete restricted areas at any time. Vendor is required to implement requested update within 48 hours.
    c. Vendor will provide daily reports of all violations.
10. The vendor shall be required to provide a data feed capable of delivering up to 2 years of historical data meeting all data requirements listed above.
11. The vendor shall be required to provide working automated sample code with the right security tokens to access the data feed.
12. The vendor shall be required to provide direct telephone and email access to a technical contact(s) for set-up and trouble issues.

13. An edited copy of version 0.3 of the Mobility Data Specification is provided in exhibit ?, indicating which "optional" data elements are instead "required" by Metropolitan Government of Nashville & Davidson County.
14. **Note:** In the tables below for the "Required/Optional" columns, an entry of "Required*" represents a field that was "Optional" in the base MDS guidelines but has been deemed as "Required" by Metropolitan Government of Nashville & Davidson County,
15.

Mobility Data Specification: Provider

This specification contains a data standard for *mobility as a service* providers to define a RESTful API for municipalities to access on-demand.

General Information

The following information applies to all provider API endpoints. Details on providing authorization to endpoints is specified in the auth document.

Currently, the provider API is implemented for dockless scooter and bikeshare. To implement another mode, add it to the schema/generate_schema.py file and this README and submit a pull request.

Versioning

provider APIs must handle requests for specific versions of the specification from clients.

Versioning must be implemented through the use of a custom media-type, application/vnd.mds.provider+json, combined with a required version parameter.

The version parameter specifies the dot-separated combination of major and minor versions from a published version of the specification. For example, the media-type for version 0.2.1 would be specified as application/vnd.mds.provider+json;version=0.2

Note: Normally breaking changes are covered by different major versions in semver notation. However, as this specification is still pre-1.0.0, changes in minor versions may include breaking changes, and therefore are included in the version string.

Clients must specify the version they are targeting through the Accept header. For example:

Accept: application/vnd.mds.provider+json;version=0.3
Since versioning was not added until the 0.3.0 release, if the Accept header is application/json or not set in the request, the provider API must respond as if version 0.2 was requested.

Responses to client requests must indicate the version the response adheres to through the Content-Type header. For example:

Content-Type: application/vnd.mds.provider+json;version=0.3
Since versioning was not added until the 0.3.0 release, if the Content-Type header is application/json or not set in the response, version 0.2 must be assumed.

If an unsupported or invalid version is requested, the API must respond with a status of 406 Not Acceptable. If this occurs, a client can explicitly negotiate available versions.

A client negotiates available versions using the OPTIONS method to an MDS endpoint. For example, to check if trips supports either version 0.2 or 0.3 with a preference for 0.2, the client would issue the following request:

SUMD RFP Data Requirements and Specifications

```
OPTIONS /trips/ HTTP/1.1
Host: provider.example.com
Accept: application/vnd.mds.provider+json;version=0.2,application/vnd.mds.provider+json;version=0.3;q=0.9
```

The response will include the most preferred supported version in the Content-Type header. For example, if only 0.3 is supported:

```
Content-Type: application/vnd.mds.provider+json;version=0.3
```

The client can use the returned value verbatim as a version request in the Accept header.

Response Format

The response to a client request must include a valid HTTP status code defined in the IANA HTTP Status Code Registry. It also must set the Content-Type header, as specified in the Versioning section.

Response bodies must be a UTF-8 encoded JSON object and must minimally include the MDS version and a data payload:

```
{
  "version": "x.y.z",
  "data": {
    "trips": [{
      "provider_id": "...",
      "trip_id": "...",
    }]
  }
}
```

All response fields must use lower_case_with_underscores.

JSON Schema

MDS defines JSON Schema files for trips and status_changes.

provider API responses must validate against their respective schema files. The schema files always take precedence over the language and examples in this and other supporting documentation meant for *human* consumption.

Pagination

provider APIs may decide to paginate the data payload. If so, pagination must comply with the JSON API specification.

The following keys must be used for pagination links:

- first: url to the first page of data
- last: url to the last page of data
- prev: url to the previous page of data
- next: url to the next page of data

At a minimum, paginated payloads must include a next key, which must be set to null to indicate the last page of data.

```
{
  "version": "x.y.z",
  "data": {
    "trips": [{
      "provider_id": "...",
      "trip_id": "...",
```

```
      }]
  },
  "links": {
    "first": "https://...",
    "last": "https://...",
    "prev": "https://...",
    "next": "https://..."
  }
}
```

UUIDs for Devices

MDS defines the *device* as the unit that transmits GPS or GNSS signals for a particular vehicle. A given device must have a UUID (device_id below) that is unique within the Provider's fleet.

Additionally, device_id must remain constant for the device's lifetime of service, regardless of the vehicle components that house the device.

Geographic Data

References to geographic datatypes (Point, MultiPolygon, etc.) imply coordinates encoded in the WGS 84 (EPSG:4326) standard GPS or GNSS projection expressed as Decimal Degrees.

Whenever an individual location coordinate measurement is presented, it must be represented as a GeoJSON Feature object with a corresponding timestamp property and Point geometry:

```
{
  "type": "Feature",
  "properties": {
    "timestamp": 1529968782421
  },
  "geometry": {
    "type": "Point",
    "coordinates": [
      -118.46710503101347,
      33.9909333514159
    ]
  }
}
```

Intersection Operation

For the purposes of this specification, the intersection of two geographic datatypes is defined according to the ST_Intersects PostGIS operation

If a geometry or geography shares any portion of space then they intersect. For geography -- tolerance is 0.00001 meters (so any points that are close are considered to intersect).
Overlaps, Touches, Within all imply spatial intersection. If any of the aforementioned returns true, then the geometries also spatially intersect. Disjoint implies false for spatial intersection.

Municipality Boundary

Municipalities requiring MDS Provider API compliance should provide an unambiguous digital source for the municipality boundary. This boundary must be used when determining which data each provider API endpoint will include.

The boundary should be defined as a polygon or collection of polygons. The file defining the boundary should be provided in Shapefile or GeoJSON format and hosted online at a published address that all providers and provider API consumers can access and download.

SUMD RFP Data Requirements and Specifications

Timestamps

References to timestamp imply integer milliseconds since Unix epoch. You can find the implementation of unix timestamp in milliseconds for your programming language here.

Trips

A trip represents a journey taken by a *mobility as a service* customer with a geo-tagged start and stop point.

The trips endpoint allows a user to query historical trip data.

Unless stated otherwise by the municipality, the trips endpoint must return all trips with a route which intersects with the municipality boundary.

Endpoint: /trips
Method: GET
Schema: trips schema
data Payload: { "trips": [] }, an array of objects with the following structure

Field	Type	Required/Optional	Comments
provider_id	UUID	Required	A UUID for the Provider, unique within MDS
provider_name	String	Required	The public-facing name of the Provider
device_id	UUID	Required	A unique device ID in UUID format
vehicle_id	String	Required	The Vehicle Identification Number visible on the vehicle itself
vehicle_type	Enum	Required	See vehicle types table
propulsion_type	Enum[]	Required	Array of propulsion types; allows multiple values
trip_id	UUID	Required	A unique ID for each trip
trip_duration	Integer	Required	Time, in Seconds
trip_distance	Integer	Required	Trip Distance, in Meters
route	GeoJSON FeatureCollection	Required	See Routes detail below
accuracy	Integer	Required	The approximate level of accuracy, in meters, of Points within route
start_time	timestamp	Required	
end_time	timestamp	Required	
publication_time	timestamp	Required	Date/time that trip became available through the trips endpoint
parking_verification_url	String	Optional	A URL to a photo (or other evidence) of proper vehicle parking
standard_cost	Integer	Required	The cost, in cents, that it would cost to perform that trip in the standard operation of the System
actual_cost	Integer	Required	The actual cost, in cents, paid by the customer of the *mobility as a service* provider

Trips Query Parameters

The trips API should allow querying trips with a combination of query parameters.

- device_id
- vehicle_id
- min_end_time: filters for trips where end_time occurs at or after the given time
- max_end_time: filters for trips where end_time occurs before the given time

SUMD RFP Data Requirements and Specifications

When multiple query parameters are specified, they should all apply to the returned trips. For example, a request with ?min_end_time=1549800000000&max_end_time=1549886400000 should only return trips whose end time falls in the range [1549800000000, 1549886400000).

Vehicle Types

vehicle_type
bicycle
scooter

Propulsion Types

propulsion_type	Description
human	Pedal or foot propulsion
electric_assist	Provides power only alongside human propulsion
electric	Contains throttle mode with a battery-powered motor
combustion	Contains throttle mode with a gas engine-powered motor

A device may have one or more values from the propulsion_type, depending on the number of modes of operation. For example, a scooter that can be powered by foot or by electric motor would have the propulsion_type represented by the array ['human', 'electric']. A bicycle with pedal-assist would have the propulsion_type represented by the array ['human', 'electric_assist'] if it can also be operated as a traditional bicycle.

Routes

To represent a route, MDS provider APIs must create a GeoJSON FeatureCollection, which includes every observed point in the route, even those which occur outside the municipality boundary.

Routes must include at least 2 points: the start point and end point. Routes must include all possible GPS or GNSS samples collected by a Provider. Providers may round the latitude and longitude to the level of precision representing the maximum accuracy of the specific measurement. For example, a-GPS is accurate to 5 decimal places, differential GPS is generally accurate to 6 decimal places. Providers may round those readings to the appropriate number for their systems.


```
"route": {
  "type": "FeatureCollection",
  "features": [{
    "type": "Feature",
    "properties": {
      "timestamp": 1529968782421
    },
    "geometry": {
      "type": "Point",
      "coordinates": [
        -118.46710503101347,
        33.9909333514159
      ]
    }
  },
  {
    "type": "Feature",
    "properties": {
      "timestamp": 1531007628377
    },
    "geometry": {
      "type": "Point",
      "coordinates": [
        -118.464851975441,
```

```
            33.990366257735
          ]
        }
      }]
}
```


Status Changes

The status of the inventory of vehicles available for customer use.

The status changes endpoint allows a user to query the historical availability for a system within a time range.

Unless stated otherwise by the municipality, this endpoint must return only those status changes with a event_location that intersects with the municipality boundary.

Note: As a result of this definition, consumers should query the trips endpoint to infer when vehicles enter or leave the municipality boundary.

Endpoint: /status_changes
Method: GET
Schema: status_changes schema
data Payload: { "status_changes": [] }, an array of objects with the following structure

Field	Type	Required/Optional	Comments
provider_id	UUID	Required	A UUID for the Provider, unique within MDS
provider_name	String	Required	The public-facing name of the Provider
device_id	UUID	Required	A unique device ID in UUID format
vehicle_id	String	Required	The Vehicle Identification Number visible on the vehicle itself
vehicle_type	Enum	Required	see vehicle types table
propulsion_type	Enum[]	Required	Array of propulsion types; allows multiple values
event_type	Enum	Required	See event types table
event_type_reason	Enum	Required	Reason for status change, allowable values determined by event type
event_time	timestamp	Required	Date/time that event occurred at. See Event Times
publication_time	timestamp	Required	Date/time that event became available through the status changes endpoint
event_location	GeoJSON Point Feature	Required	
battery_pct	Float	Required if Applicable	Percent battery charge of device, expressed between 0 and 1
associated_trip	UUID	Required if Applicable	Trip UUID (foreign key to Trips API), required if event_type_reason is user_pick_up or user_drop_off, or for any other status change event that marks the end of a trip.

Event Times

Because of the unreliability of device clocks, the Provider is unlikely to know with total confidence what time an event occurred at. However, they are responsible for constructing as accurate a timeline as possible. Most importantly, the order of the timestamps for a particular device's events must reflect the Provider's best understanding of the order in which those events occurred.

Status Changes Query Parameters

The status_changes API should allow querying status changes with a combination of query parameters.

SUMD RFP Data Requirements and Specifications

- start_time: filters for status changes where event_time occurs at or after the given time
- end_time: filters for status changes where event_time occurs before the given time

When multiple query parameters are specified, they should all apply to the returned status changes. For example, a request with ?start_time=1549800000000&end_time=1549886400000 should only return status changes whose event_time falls in the range [1549800000000, 1549886400000).

Event Types

event_type	Description	event_type_reason	Description
available	A device becomes available for customer use	service_start	Device introduced into service at the beginning of the day (if program does not operate 24/7)
		user_drop_off	User ends reservation
		rebalance_drop_off	Device moved for rebalancing
		maintenance_drop_off	Device introduced into service after being removed for maintenance
		agency_drop_off	The administrative agency (ie, DOT) drops a device into the PROW using an admin code or similar
reserved	A customer reserves a device (even if trip has not started yet)	user_pick_up	Customer reserves device
unavailable	A device is on the street but becomes unavailable for customer use	maintenance	A device is no longer available due to equipment issues
		low_battery	A device is no longer available due to insufficient battery
removed	A device is removed from the street and unavailable for customer use	service_end	Device removed from street because service has ended for the day (if program does not operate 24/7)
		rebalance_pick_up	Device removed from street and will be placed at another location to rebalance service
		maintenance_pick_up	Device removed from street so it can be worked on
		agency_pick_up	The administrative agency (ie, DOT) removes a device using an admin code or similar

Realtime Data

All MDS compatible provider APIs must expose a public GBFS feed as well. Given that GBFS hasn't fully evolved to support dockless mobility yet, we follow the current guidelines in making bike information avaliable to the public.

- gbfs.json is always required and must contain a feeds property that lists all published feeds
- system_information.json is always required
- free_bike_status.json is required for MDS
- station_information.json and station_status.json don't apply for MDS

SUMD RFP Data Requirements and Specifications

Mobility Data Specification: Agency

This specification contains a collection of RESTful APIs used to specify the digital relationship between *mobility as a service* Providers and the Agencies that regulate them.

- Authors: LADOT
- Date: 25 Feb 2019
- Version: BETA

Authorization

When making requests, the Agency API expects provider_id to be part of the claims in a JWT access_token in the Authorization header, in the form Authorization: Bearer <access_token>. The token issuance, expiration and revocation policies are at the discretion of the Agency.

Timestamps

As with the Provider API, timestamp refers to integer milliseconds since Unix epoch.

Vehicles

The /vehicles endpoint returns the specified vehicle (if a device_id is provided) or a list of known vehicles. Providers can only retrieve data for vehicles in their registered fleet.

Endpoint: /vehicles/{device_id} Method: GET

Path Params:

Param	Type	Required/Optional	Description
device_id	UUID	Required*	If provided, retrieve the specified vehicle

200 Success Response:

If device_id is specified, GET will return a single vehicle record, otherwise it will be a list of vehicle records with pagination details per the JSON API spec:

```
{
        "vehicles": [ ... ]
        "links": {
    "first": "https://...",
    "last": "https://...",
    "prev": "https://...",
    "next": "https://..."
```

```
  }
}
```

A vehicle record is as follows:

Field	Type	Field Description
device_id	UUID	Provided by Operator to uniquely identify a vehicle
provider_id	UUID	Issued by City and tracked
vehicle_id	String	Vehicle Identification Number (vehicle_id) visible on vehicle
type	Enum	Vehicle Type
propulsion	Enum[]	Array of Propulsion Type; allows multiple values
year	Integer	Year Manufactured
mfgr	String	Vehicle Manufacturer
model	String	Vehicle Model
status	Enum	Current vehicle status. See Vehicle Status
prev_event	Enum	Last Vehicle Event
updated	Timestamp	Date of last event update

404 Failure Response:

No content returned on vehicle not found.

Vehicle - Register

The /vehicles registration endpoint is used to register a vehicle for use in the Agency jurisdiction.

Endpoint: /vehicles Method: POST

Body Params:

Field	Type	Required/Optional	Field Description
device_id	UUID	Required	Provided by Operator to uniquely identify a vehicle
vehicle_id	String	Required	Vehicle Identification Number (vehicle_id) visible on vehicle
type	Enum	Required	Vehicle Type
propulsion	Enum[]	Required	Array of Propulsion Type; allows multiple values
year	Integer	Required*	Year Manufactured
mfgr	String	Required*	Vehicle Manufacturer
model	String	Required*	Vehicle Model

201 Success Response:

No content returned on success.

400 Failure Response:

error	error_description	error_details[]
bad_param	A validation error occurred.	Array of parameters with errors
missing_param	A required parameter is missing.	Array of missing parameters

409 Failure Response:

error	error_description	error_details[]
already_registered	A vehicle with device_id is already registered	

Vehicle - Update

The /vehicles update endpoint is used to update some mutable aspect of a vehicle. For now, only vehicle_id.

Endpoint: /vehicles/{device_id} Method: PUT

Body Params:

Field	Type	Required/Optional	Field Description
vehicle_id	String	Required	Vehicle Identification Number (vehicle_id) visible on vehicle

201 Success Response:

No content returned on success.

400 Failure Response:

error	error_description	error_details[]
bad_param	A validation error occurred.	Array of parameters with errors
missing_param	A required parameter is missing.	Array of missing parameters

404 Failure Response:

No content returned if no vehicle matching device_id is found.

SUMD RFP Data Requirements and Specifications

Vehicle - Event

The vehicle /event endpoint allows the Provider to control the state of the vehicle including deregister a vehicle from the fleet.

Endpoint: /vehicles/{device_id}/event Method: POST

Path Params:

Field	Type	Required/Optional	Field Description
device_id	UUID	Required	ID used in Register

Body Params:

Field	Type	Required/Optional	Field Description
event_type	Enum	Required	see Vehicle Events
event_type_reason	Enum	Required if Available	see Vehicle Events
timestamp	Timestamp	Required	Date of last event update
telemetry	Telemetry	Required	Single point of telemetry
trip_id	UUID	Required*	UUID provided by Operator to uniquely identify the trip. Required for trip_start, trip_end, trip_enter, and trip_leave event types

201 Success Response:

Field	Type	Field Description
device_id	UUID	UUID provided by Operator to uniquely identify a vehicle
status	Enum	Vehicle status based on posted event_type. See Vehicle Status

400 Failure Response:

error	error_description	error_details[]
bad_param	A validation error occurred	Array of parameters with errors
missing_param	A required parameter is missing	Array of missing parameters
unregistered	Vehicle is not registered	

SUMD RFP Data Requirements and Specifications

Vehicles - Telemetry

The vehicle /telemetry endpoint allows a Provider to send vehicle telemetry data in a batch for any number of vehicles in the fleet.

The Update Telemetry endpoint (/telemetry) shall be called for the specific trip within 24 hrs after the vehicle trip is over.
For any given trip, data reported via the (/telemetry) endpoint shall contain temporal and location data for every 300 ft (91 meters) while vehicle is in motion and 30 seconds while at rest. For Mobility Service Providers who do not calculate distance in real-time, a periodic rate of 14 seconds can be used while vehicle is in motion.

Endpoint: /vehicles/telemetry Method: POST

Body Params:

Field	Type	Required/Optional	Field Description
data	Telemetry[]	Required	Array of telemetry for one or more vehicles.

201 Success Response:

Field	Type	Field Description
result	String	Responds with number of successfully written telemetry data points and total number of provided points.
failures	Telemetry[]	Array of failed telemetry for zero or more vehicles (empty if all successful).

400 Failure Response:

error	error_description	error_details[]
bad_param	A validation error occurred.	Array of parameters with errors
invalid_data	None of the provided data was valid.	
missing_param	A required parameter is missing.	Array of missing parameters

Service Areas

The /service_areas endpoint gets the list of service areas available to the Provider or a single area.

Endpoint: /service_areas/{service_area_id} Method: GET

Path Params:

SUMD RFP Data Requirements and Specifications

Field	Type	Required/Optional	Field Description
service_area_id	UUID	Optional	If provided, retrieve a specific service area (e.g. a retired or old service area). If omitted, will return all active service areas.

Query Params:

Parameter	Type	Required/Optional	Description
bbox	String	Optional	The bounding box upper, left, lower and right coordinates in WGS84 degrees. All geometries overlapping this rectangle will be returned. The format is: lat,long;lat,long

200 Success Response:

Field	Types	Required/Optional	Field Description
service_area_id	UUID	Required	UUID issued by city
start_date	Timestamp	Required	Date at which this service area became effective
end_date	Timestamp	Optional	If exists, Date at which this service area was replaced.
area	MultiPolygon	Required	GeoJson MultiPolygon in WGS84 degrees.
prev_area	UUID	Optional	If exists, the UUID of the prior service area.
replacement_area	UUID	Optional	If exists, the UUID of the service area that replaced this one
type	Enum	Required	See area types

Vehicle Events

List of valid vehicle events and the resulting vehicle status if the event is sucessful. Note that to handle out-of-order events, the validity of the initial-status is not enforced. Events received out-of-order may result in transient incorrect vehicle states.

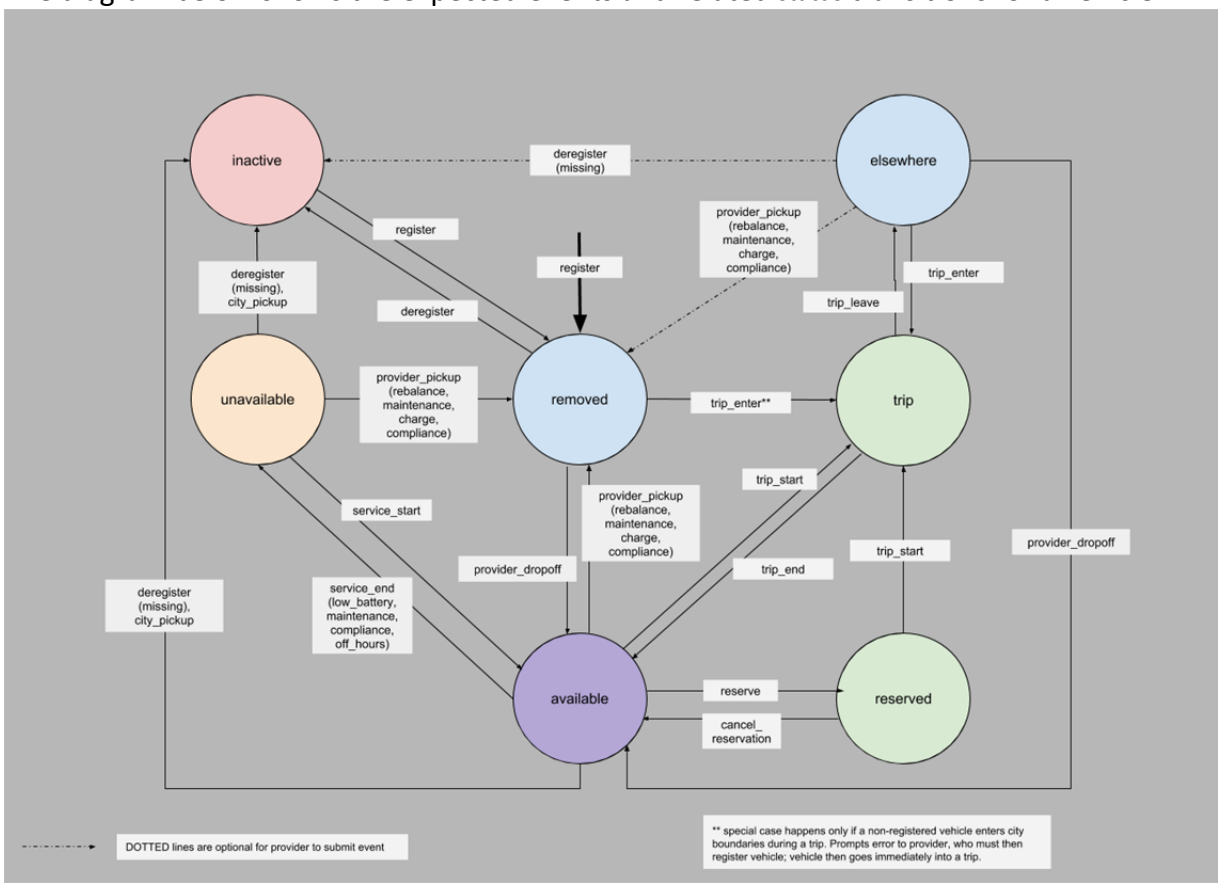
event_type	event_type_reason	description	valid initial status	status on success	status_description
register		Default state for a newly registered vehicle	inactive	removed	A vehicle is in the active fleet but not yet available for customer use
service_start		Vehicle	unavailable	available	Vehicle is on the

event_type	event_type_reason	description	valid initial status	status on success	status_description
		introduced into service at the beginning of the day (if program does not operate 24/7)			street and available for customer use.
service_end	low_battery, maintenance, compliance, off_hours	A vehicle is no longer available due to event_type_reason	available	unavailable	
provider_drop_off		Vehicle moved for rebalancing	removed, elsewhere	available	
provider_pick_up	rebalance, maintenance, charge, compliance	Vehicle removed from street and will be placed at another location to rebalance service	available, unavailable, elsewhere	removed	
city_pick_up		Vehicle removed by city	available, unavailable	removed	
reserve		Customer reserves vehicle	available	reserved	Vehicle is reserved or in use.
cancel_reservation		Customer cancels reservation	reserved	available	
trip_start		Customer starts a trip	available, reserved	trip	
trip_enter		Customer enters the municipal area managed by agency during an active trip.	removed, elsewhere	trip	
trip_leave		Customer leaves the municipal area managed by agency during an active trip.	trip	elsewhere	
trip_end		Customer ends trip and reservation	trip	available	

SUMD RFP Data Requirements and Specifications

event_type	event_type_reason	description	valid initial status	status on success	status_description
deregister	missing, decommissioned	A vehicle is deregistered	available, unavailable, removed, elsewhere	inactive	A vehicle is deactivated from the fleet.

The diagram below shows the expected events and related status transitions for a vehicle:



SUMD RFP Data Requirements and Specifications

Telemetry Data

A standard point of vehicle telemetry. References to latitude and longitude imply coordinates encoded in the WGS 84 (EPSG:4326) standard GPS or GNSS projection expressed as Decimal Degrees.

Field	Type	Required/Optional	Field Description
device_id	UUID	Required	ID used in Register
timestamp	Timestamp	Required	Date/time that event occurred. Based on GPS or GNSS clock
gps	Object	Required	Telemetry position data
gps.lat	Double	Required	Latitude of the location
gps.lng	Double	Required	Longitude of the location
gps.altitude	Double	Required if Available	Altitude above mean sea level in meters
gps.heading	Double	Required if Available	Degrees - clockwise starting at 0 degrees at true North
gps.speed	Float	Required if Available	Speed in meters / sec
gps.hdop	Float	Required if Available	Horizontal GPS or GNSS accuracy value (see hdop)
gps.satellites	Integer	Required if Available	Number of GPS or GNSS satellites
charge	Float	Required if Applicable	Percent battery charge of vehicle, expressed between 0 and 1

Enum Definitions

Area Types

type	Description
unrestricted	Areas where vehicles may be picked up/dropped off. A provider's unrestricted area shall be contained completely inside the agency's unrestricted area for the provider in question, but it need not cover the entire agency unrestricted area. See the provider version of the service areas endpoint
restricted	Areas where vehicle pick-up/drop-off is not allowed
preferred_pick_up	Areas where users are encouraged to pick up vehicles
preferred_drop_off	Areas where users are encouraged to drop off vehicles

SUMD RFP Data Requirements and Specifications

Vehicle Type

type
bicycle
scooter

Propulsion Type

propulsion	Description
human	Pedal or foot propulsion
electric_assist	Provides power only alongside human propulsion
electric	Contains throttle mode with a battery-powered motor
combustion	Contains throttle mode with a gas engine-powered motor

A vehicle may have one or more values from the propulsion, depending on the number of modes of operation. For example, a scooter that can be powered by foot or by electric motor would have the propulsion represented by the array ['human', 'electric']. A bicycle with pedal-assist would have the propulsion represented by the array ['human', 'electric_assist'] if it can also be operated as a traditional bicycle.

Responses

- **200:** OK: operation successful.
- **201:** Created: POST operations, new object created
- **400:** Bad request.
- **401:** Unauthorized: Invalid, expired, or insufficient scope of token.
- **404:** Not Found: Object does not exist, returned on GET or POST operations if the object does not exist.
- **409:** Conflict: POST operations when an object already exists and an update is not possible.
- **500:** Internal server error: In this case, the answer may contain a text/plain body with an error message for troubleshooting.

Error Message Format

Field	Type	Field Description
error	String	Error message string
error_description	String	Human readable error description (can be localized)
error_details	String[]	Array of error details

SUMD RFP Data Requirements and Specifications