Asset Tracker Device Parameter API

Version 1.5
2018-10-15
NimbeLink Corp

Revision History

Revision	Date	Author(s)	Description
1.0	2017-09-08	BR	Initial revision
1.1	2018-04-02	JY	Added in default values
1.2	2018-04-09	BR	Added information on parameters in Section 1.2.1
1.3	2018-04-10	KR	Cleaned up formatting and examples
1.4	2018-10-11	BR	Added information on gps_fence configuration
1.5	2018-10-15	BR	Added information on tracking mode sensitivity configuration

Contents

1 Device Parameter JSON structures

1.1 “network”

This module is responsible for scheduling connections to the cloud via the cellular modem. **The frequency that the device makes a cellular connection has the biggest impact on the battery life of the device.**

1.1.1 Parameters

`period`

Specifies an interval in seconds between subsequent connections to the cloud. In general, **this parameter has the single biggest impact on the battery life of the device.**

Valid range: `900` to `86400` seconds.

Default: `1800` seconds

1.1.2 Sample section

```
"network": {
  "period": 1800
}
```

1.2 “accelerometer”

This module is responsible for monitoring the accelerometer. Depending on the configured mode, this can monitor for movement or simply report the X-Y-Z axis values on a scheduled basis.

1.2.1 Parameters

`enabled`

Determines if the accelerometer module is enabled or disabled.

Valid values: `true`, `false`

Default: `false`

`mode`

Sets the mode of operation for measurement or movement detection as follows:

- “0”: Disabled (Default. Same effect as setting `enabled` to `false`.)
- “1”: After movement has completed, locate the device
  - Once movement is detected, this mode waits for no movement for a period of `inactivity_before_movement_end` seconds before triggering a location event (see `tracking` section for parameter description).
- “2”: During movement, locate the device
  - Once movement is detected, this mode monitors where the device is and triggers location events periodically during movement (see `tracking` section for parameters that determine operation in this mode).
- “3”: Measure accelereometer periodically
  - This mode periodically wakes up and records the current accelerometer axis readings.
- “4”: Reserved
- “5”: Reserved

`measurement_period`

Specifies an interval in seconds between subsequent accelerometer measurements. This parameter only applies when the accelerometer `mode` is set to 3 (measure X-Y-Z periodically).

Valid range: `15` to `86400` seconds

Default: `15` seconds.

`tracking`

The parameters in this section are all nested under the `tracking` block.

The `location_period` and `locations_per_checkin` parameters are only used in accelerometer mode `2`, i.e. when the device is tracking its location based on movement. The remaining parameters are used in both accelerometer modes `1` and `2`. The parameters in this `tracking` section have no effect when the device is in accelerometer mode `3` (measuring and reporting the raw X-Y-Z readings).

When determining whether a movement event has occurred, the device goes through two processing steps: a hardware filter and a “token bucket” filter.

> First, the device determines whether the measured acceleration in mg is greater than the `lower_accel_threshold`.
>
> Second, if the acceleration is above the threshold, then a “token” is removed from the bucket on the rising edge of the transition, as well as every 100 ms after while the accelerometer measurement is above the `lower_accel_threshold`. When the number of tokens reaches 0, the device generates a “movement” record. The bucket fills at a rate determined by the values for `bucket_size` and `bucket_fill_time`. By default, `bucket_size=2` and `bucket_fill_time=1`, so a token would get added to the bucket every 0.5 seconds (`bucket_fill_time / bucket_size`).

After the device determines movement has started, it continues to monitor for movement based on the process described above. Whenever the token bucket is emptied, the timer specified by `inactivity_before_movement_end` is reset. The token bucket must not be emptied for a period of `inactivity_before_movement_end` seconds before the device will generate a “movementEnd” record. See Section 1.2.2 for a sample timing diagram.

`connect_on_movement_start`

When set to `true`, the device will connect and report in after movement has been detected. The device will attempt to take a location reading before connecting.

Valid values: `true`, `false`

Default: `true`

`connect_on_movement_end`

When set to `true`, the device will connect and report in after movement has ended.

Valid values: `true`, `false`

Default: `true`

`location_period`

Determines the amount of time in seconds between location readings while the device is moving.

Valid range: `300` to `86400` seconds

Default: `1800` seconds

`locations_per_checkin`

Determines how many location readings should be taken between cellular connections while the device is moving. When the device connects and checks in, it will report all the location readings it took since the previous connection.

Valid range: `0` to `20`

Default: `8`

`inactivity_before_movement_end`

Determines the amount of time in seconds after a device has stopped moving before a “movement_end” event is generated. This time can be used as a time buffer to keep the device actively tracking even if movement stops for a brief period of time. For example, if the device is on a delivery truck, the truck may stop for 5-10 minutes as deliveries are made. Setting `inactivity_before_movement_end` to >600 seconds (10 minutes) will prevent the device from generating extra “movement_start” and “movement_end” events. This can help minimize cellular connections and preserve battery life if either `connect_on_movement_start` or `connect_on_movement_end` are set to `true`.

Valid range: `60` to `86400` seconds

Default: `900` seconds

`lower_accel_threshold`

Specifies the minimum acceleration in mg on any axis that will cause a token to be removed from the bucket filter. The device filters out the steady state readings, so gravity will not trigger a movement event on a device at rest.

Valid range: `50` to `2000` mg

Default: `50` mg

`bucket_size`

Specifies the number of tokens in the bucket filter used to determine when a movement event has occurred.

Valid range: 1 to 100

Default: 2

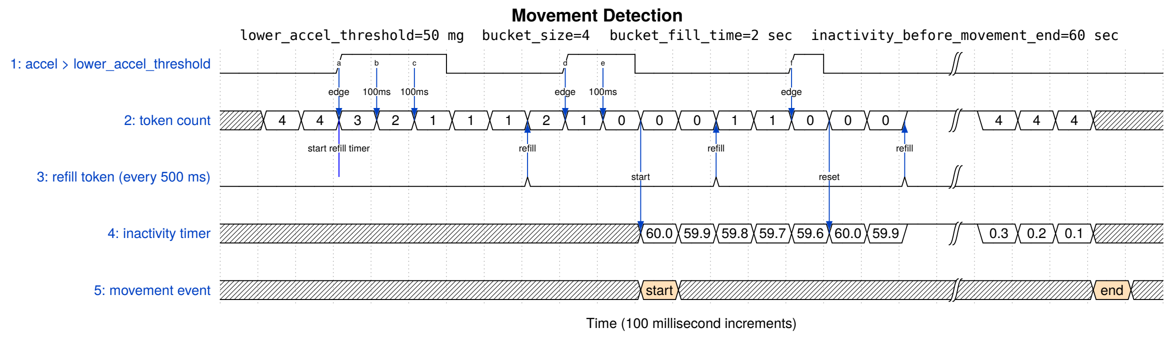
`bucket_fill_time`

Specifies the number of seconds it takes for the bucket to go from empty to full.

Valid range: `1` to `3600` seconds

Default: `1` second

1.2.2 Sample timing diagram of movement events in tracking mode



The timing diagram above illustrates an example set of accelerometer events.

Signal 1 is high when the accelerometer measures any X, Y, or Z axis value above the configured `lower_accel_threshold` mg.

Signal 2 indicates the number of remaining tokens in the bucket filter.

Signal 3 indicates when a token is added back into to the bucket, based on a rate of `bucket_fill_time / bucket_size` seconds.

Signal 4 is the timer used to determine when movement has ended, based on the configured value for `inactivity_before_movement_end`.

Signal 5 shows the actual movement start and end events that get reported by the device to the cloud.

1.2.3 Sample section

```
"accelerometer": {
  "enabled": true,
  "mode": 2,
  "measurement_period": 300,
  "tracking": {
    "connect_on_movement_start": true,
    "connect_on_movement_end": true,
    "location_period": 1800,
    "locations_per_checkin": 8,
    "inactivity_before_movement_end": 900,
    "lower_accel_threshold": 50,
    "bucket_size": 4,
    "bucket_fill_time": 2
  }
}
```

1.3 “location”

The parameters in this block determine how the device takes location readings, including GPS fixes and cell tower scans. **The frequency that a device takes location readings has a significant impact on battery life.**

1.3.1 Parameters

`enabled`

Enables or disables automatic scheduling of location fixes. These are independent of any other triggers for location fixes on the system, such as from movement detection.

Valid values: `true`, `false`

Default: `false`

`sampling_period`

Specifies an interval in seconds between subsequent location readings. This parameter only applies to scheduled location readings. The frequency of location readings outside of the scheduled readings – such as when the device is in accelerometer `mode` 2 and is in motion – is determined by separate parameters (see `tracking` section for more information).

Valid range: `300` to `86400` seconds

Default: `1800` seconds.

`useGPS`

Determines whether the device attempts to get a GPS fix when a location reading is taken. This applies both to scheduled location readings as well as location readings taken while the device is moving in.

Valid values: `true`, `false`

Default: `true`

`useCell`

Determines whether the device attempts to perform a cell tower scan when a location reading is taken.

Valid values: `true`, `false`

Default: `true`

`useWiFi`

Determines whether the device attempts to perform a WiFi scan when a location reading is taken.

Valid values: `true`, `false`

Default: `true`

1.3.2 Sample section

```
"location": {
  "enabled": true,
  "sampling_period": 300,
  "useGPS": true,
  "useCell": true,
  "useWiFi": true
}
```

1.4 “gps_fence”

This module determines when an asynchronous cellular check-in is performed based on the device entering/exiting a GPS-fence. The term “GPS-fence” is used since the device only checks geo-fence status based on GPS records. Location readings from other sources (cellular, WiFi) are **not** used for geo-fencing, since these locations are determined on the cloud side. **The state of the GPS-fence is checked whenever the device takes a GPS reading for any reason (scheduled, during movement in accelerometer mode 2, etc.). It is NOT checked constantly.**

A total of 3 GPS-fences can be configured: 2 fixed, and 1 automatic.

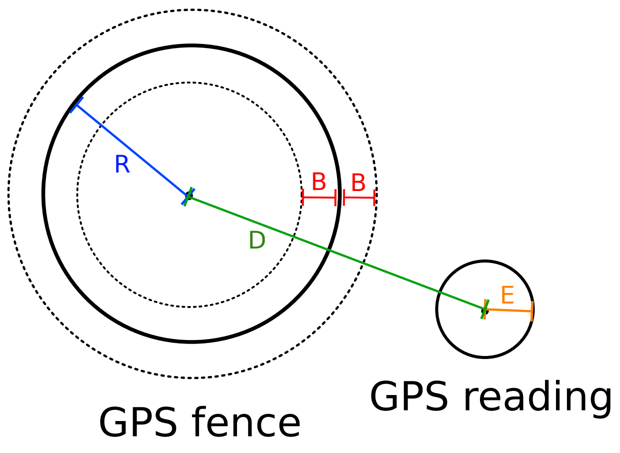
The fixed fences are configured as a center latitude/longitude and a radius. Whenever the device enters or exits the defined fence, it performs an asynchronous cellular check-in.

The automatic fence is configured as a radius only. When the device takes a GPS reading that is outside of this radius, it performs a cellular check-in and stores its current location as the new center of the automatic GPS-fence. This automatic GPS-fence is especially useful when the coordinates of the device’s deployment site are unknown, but it shouldn’t be moving far away from its location once deployed.

The GPS-fence state for a device is determined as follows:

Variables:

- D is the distance to the center of the fence from the latitude and longitude of the GPS reading
- E is the estimated error in meters from the GPS reading
- B is the buffer on either side of the fence perimeter
- R is the radius of the fence

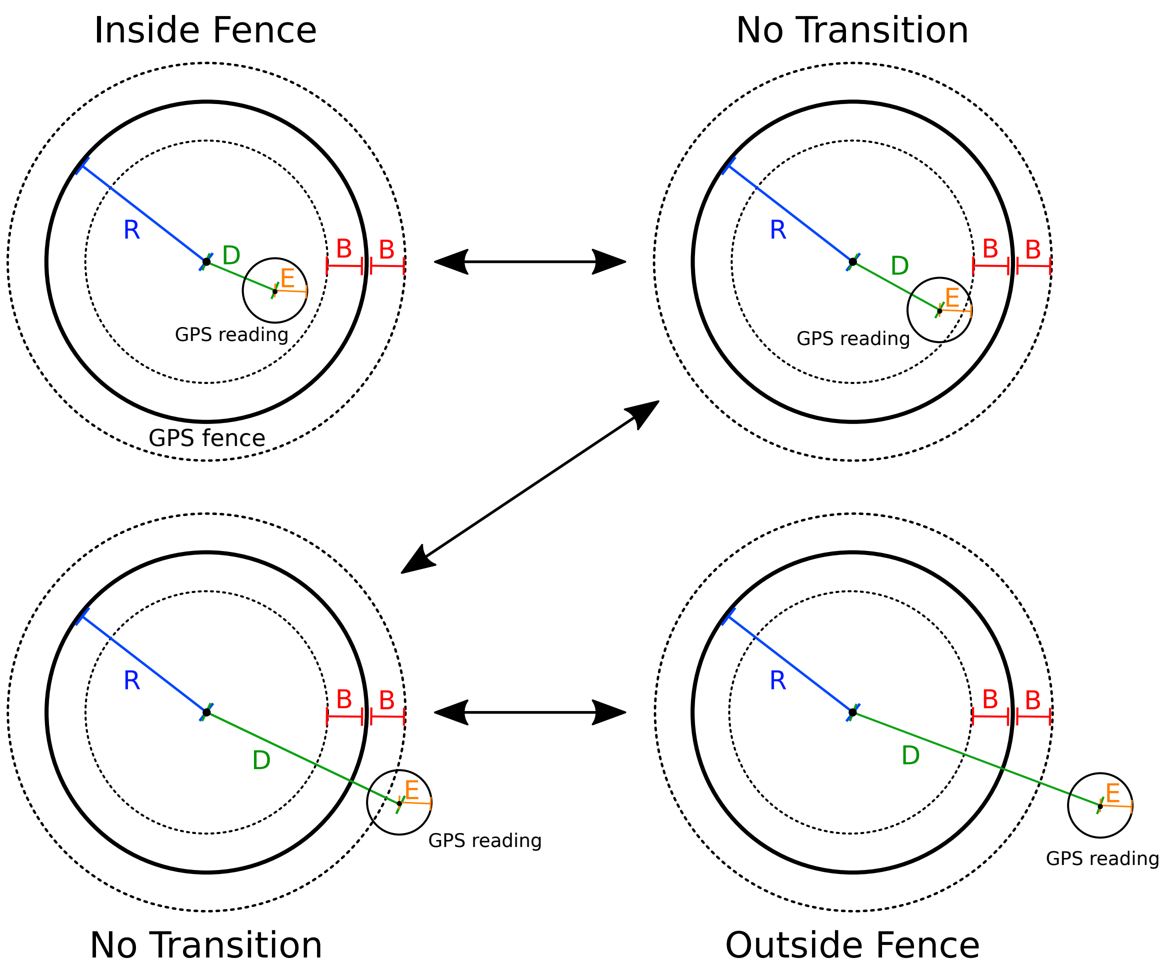


Logic:

- If the last status of the fence is that the device is inside the fence, then consider the reading to be outside the fence if $D \geq R + (E + B)$
- If the last status of the fence is that the device is outside the fence, then consider the reading to be inside the fence if $D \leq R - (E + B)$

In other words, the device is considered outside the GPS-fence when the circle defined by the GPS reading’s accuracy is completely outside the outer buffer of the GPS-fence, and the device is considered inside the GPS-fence when the circle defined by the GPS reading’s accuracy is completely inside the inner buffer of the GPS-fence.

Example GPS-fence state transition:



1.4.1 Parameters

`gps_fence_1`

`enabled`

Enables asynchronous check-ins based on GPS-fence 1.

Valid values: `true`, `false`

Default: `false`

`latitude`

Latitude of the GPS-fence center point.

Valid values: `-90.0` to `90.0` degrees

Default: `0.0` degrees

`longitude`

Longitude of the GPS-fence center point.

Valid values: `-180.0` to `180.0` degrees

Default: `0.0` degrees

`radius`

Radius in meters of the GPS-fence.

Valid values: `0.0` to `42949672.95` meters

Default: `0.0` meters

`buffer`

Defines the buffer width in meters on either side of the GPS-fence perimeter that the device must be clear of in order to transition between GPS-fence states (inside or outside).

Valid values: `0.0` to `42949672.95` meters

Default: `0.0` meters

`gps_fence_2`

`enabled`

Enables asynchronous check-ins based on GPS-fence 2.

Valid values: `true`, `false`

Default: `false`

`latitude`

Latitude of the GPS-fence center point.

Valid values: `-90.0` to `90.0` degrees

Default: `0.0` degrees

`longitude`

Longitude of the GPS-fence center point.

Valid values: `-180.0` to `180.0` degrees

Default: `0.0` degrees

`radius`

Radius in meters of the GPS-fence.

Valid values: `0.0` to `42949672.95` meters

Default: `0.0` meters

`buffer`

Defines the buffer width in meters on either side of the GPS-fence perimeter that the device must be clear of in order to transition between GPS-fence states (inside or outside).

Valid values: `0.0` to `42949672.95` meters

Default: `0.0` meters

`gps_fence_auto`

`enabled`

Enables asynchronous check-ins based on the automatic GPS-fence.

Valid values: `true`, `false`

Default: `false`

`radius`

Radius in meters of the GPS-fence.

Valid values: `0.0` to `42949672.95` meters

Default: `0.0` meters

`buffer`

Defines the buffer width in meters on either side of the GPS-fence perimeter that the device must be clear of in order to transition between GPS-fence states (inside or outside).

Valid values: `0.0` to `42949672.95` meters

Default: `0.0` meters

1.4.2 Sample section

```
"gps_fence": {
  "gps_fence_1": {
    "enabled": true,
    "latitude": 45.1234,
    "longitude": -93.1234,
    "radius": 1000,
    "buffer": 20
  },
  "gps_fence_2": {
    "enabled": true,
    "latitude": 45.789,
    "longitude": -93.789,
    "radius": 1000,
    "buffer": 0
  },
  "gps_fence_auto": {
    "enabled": true,
    "radius": 5000,
    "buffer": 0
  }
}
```

1.5 “temperature”

This module is responsible for sampling the environmental temperature on a periodic basis.

1.5.1 Parameters

`enabled`

Enables or disables periodic sampling of the temperature sensor.

Valid values: `true`, `false`

Default: `false`

`sampling_period`

Specifies an interval in seconds between subsequent temperature measurements.

Valid range: `15` to `86400` seconds

Default: `15` seconds

1.5.2 Sample section

```
"temperature": {
  "enabled": true,
  "sampling_period": 300
}
```

2 Sample API request body

```
{
  "config": {
    "accelerometer": {
      "enabled": false,
      "mode": 3,
      "measurement_period": 300,
      "tracking": {
        "connect_on_movement_start": true,
        "connect_on_movement_end": true,
        "location_period": 900,
        "locations_per_checkin": 4,
        "inactivity_before_movement_end": 900,
        "lower_accel_threshold": 50,
        "bucket_size": 4,
        "bucket_fill_time": 2
      }
    },
    "location": {
      "enabled": true,
      "sampling_period": 3600,
      "useGPS": true,
      "useCell": true,
      "useWiFi": true
    },
    "gps_fence": {
      "gps_fence_1": {
        "enabled": true,
        "latitude": 45.1234,
        "longitude": -93.1234,
        "radius": 1000,
        "buffer": 20
      },
      "gps_fence_2": {
        "enabled": true,
        "latitude": 45.789,
        "longitude": -93.789,
        "radius": 1000,
        "buffer": 0
      },
      "gps_fence_auto": {
        "enabled": true,
        "radius": 5000,
        "buffer": 0
      }
    },
    "network": {
      "period": 3600
    },
    "temperature": {
      "enabled": true,
      "sampling_period": 300
    }
  }
}
```

3 Sample sequence of API requests

3.1 Set environment variables

Set environment variables for the device ID and for the credentials provided by NimbeLink for API access:

```
NLINK_USER="username"; NLINK_AUTH="secret"; NLINK_APIKEY="api_key"
DEVICEID="at-xxxxxxxxxxxxx"
```

Note: the DEVICEID should be lowercase.

3.2 Authenticate with the API

```
curl -H "x-api-key: ${NLINK_APIKEY}" -H "Content-Type: application/json" -X POST -d "{ \"
    username\": \"${NLINK_USER}\", \"password\": \"${NLINK_AUTH}\" }" https://api.iot.
    nimbelink.net/v1/auth
```

This call returns:

```
{"token": "xxxyyyzzz"}
```

Copy the returned token into a variable for use on subsequent curl requests:

```
TOKEN='xxxyyyzzz'
```

3.3 Get device configuration

Send the following HTTP GET request:

```
curl -H "x-api-key: ${NLINK_APIKEY}" -H "Content-Type: application/json" -H "Authorization
    : ${TOKEN}" -X GET https://api.iot.nimbelink.net/v1/devices/${DEVICEID}/config
```

The response is similar to:

```
{"config": {"desired": {"temperature": {"sampling_period": 300, "enabled": true}, "network
    ": {"period": 3600}, "accelerometer": {"enabled": true, "mode": 3, "measurement_period
    ": 20}, "location": {"enabled": true, "sampling_period": 300}}, "reported": {"network
    ": {"period": 900}, "accelerometer": {"enabled": true, "measurement_period": 20, "mode
    ": 3}, "location": {"enabled": true, "sampling_period": 300}, "temperature": {"enabled
    ": true, "sampling_period": 300}}}, "deviceId": "at-xxxxxxxxxxxxx"}
```

3.4 Change device configuration

Send the following HTTP PUT request:

```
curl -H "x-api-key: ${NLINK_APIKEY}" -H "Content-Type: application/json" -H "Authorization
    : ${TOKEN}" -X PUT -d '{"config":  {"accelerometer": {"enabled": true, "
    measurement_period": 20, "mode": 3}, "location": {"enabled": true, "sampling_period":
    300}, "network": {"period": 3600}, "temperature": {"enabled": true, "sampling_period":
     300}}}' https://api.iot.nimbelink.net/v1/devices/${DEVICEID}/config
```

The response is similar to:

```
{"config": {"desired": {"temperature": {"sampling_period": 300, "enabled": true}, "
    location": {"sampling_period": 300, "enabled": true}, "accelerometer": {"
    measurement_period": 20, "enabled": true, "mode": 3}, "network": {"period": 3600}}}, "
    deviceId": "at-xxxxxxxxxxxxx"}
```