FLIR ITS Public API Manual

FLIR ITS Public API Manual

V1.03

Table of Contents

Legal Notice 1

Introduction 1

Errors 1

1. Protocol Version 2

2. Device Information 3

    2.1. General device information 3

    2.2. Getting the current time from the device 3

    2.3. Sending the current time to the device 4

    2.4. Uptime of the device 4

3. Images 5

    3.1. Getting the current image 5

    3.2. Getting the image for an event 5

4. Videos 6

    4.1. Getting the video for an event 6

5. Events 7

    5.1. Structure of an event 7

    5.2. List of supported event types 8

    5.3. Getting the currently open events 9

6. Traffic Data 10

    6.1. Structure of traffic data 10

    6.2. List of supported traffic data types 11

    6.3. Get Data 11

7. Subscriptions 14

    7.1. WebSockets 14

    7.2. Subscribe 14

    7.3. Unsubscribe 17

    7.4. Keepalive 18

    7.5. ForceEvent 18

Appendix A - Event Types 19

    Traffic Events 19

    Technical Events 25

Appendix B - Event States 29

Appendix C - Traffic Data Types 29

    Traffic Data 29

Legal Notice

FLIR Intelligent Transportation Systems
Hospitaalweg 1B
B-8510 Marke
Belgium
Tel +32 (0)56 37.22.00
Fax +32 (0)56 37.21.96
E-mail flir@flir.com

Introduction

Welcome to the FLIR ITS Public API Manual. This manual will help you integrate your FLIR ITS device into your own software.

The API is a REST API combined with WebSockets for subscriptions. The protocol uses basic HTTP GET and POST for communication. For subscriptions from the device the users should upgrade their HTTP connection to a WebSocket and will receive events and/or data as they occur.

In all examples you will have to substitute `<device_ip_address>` with the IP address of your device. Path parameters and query parameters are indicated by prefixing the parameter name with `:` (for example `/api/image/event/:eventNumber` has the parameter `eventNumber`).

Errors

The API uses conventional HTTP response codes to indicate success or failure of an API request. In general, codes in the 2xx range indicate success, codes in the 4xx range indicate an error that resulted from the provided information (for example a required parameter was missing), and codes in the 5xx range indicate an error with the device itself.

In some cases the error responses can also contain an JSON error object. The error object has the following form:

```
{
  "messageType":"Error",
  "returnInfo":"Information about the error.",
  "returnValue":"Error"
}
```

This is a quick summary of possible error codes:

200 OK

Everything worked as expected.

400 Bad Request

The request was invalid: missing parameter, invalid parameter, unknown URL path.

403 Forbidden

The request is not supported by the device.

404 Not Found

The requested item doesn’t exist.

409 Conflict

The device cannot handle the request because of a conflicting parameter. Check the error message for more information.

500 Internal Server Error

Something went wrong on the device.

1. Protocol Version

You can retrieve the version of the Public API.

URL

`/api/version`

Method

`GET`

Response Body

```
{
  "protocolVersion": "V1.03"
}
```

2. Device Information

The API endpoint `/api/device` allows to retrieve information about the device itself.

2.1. General device information

The device information contains company name, device name, product type and firmware version of the device.

URL

`/api/device/info`

Method

`GET`

Response Body

```
{
  "company":"Flir",
  "name":"TrafiCam x-stream S/N:223888",
  "product":"TrafiCam x-stream",
  "version":"V1.00"
}
```

2.2. Getting the current time from the device

Get the current time of the device as ISO8601 string. The UTC offset will be the offset corresponding with the time zone configured on the device.

URL

`/api/device/time`

Method

`GET`

Response Body

```
{
  "time": "2015-06-05T10:43:16.284+02:00"
}
```

2.3. Sending the current time to the device

If the device is not synced with an NTP server, you can manually set the current time to the device as ISO8601 string. The sent time will be converted to the corresponding local time in the time zone configured on the device. The resolution of the time you can set is one second. The decimal fraction of a second will be ignored.

URL

`/api/device/time`

Method

`POST`

Request Body

```
{
  "time": "2015-06-05T10:43:16.284+02:00"
}
```

Response Body

```
{
  "time": "2015-06-05T08:43:16.284+00:00"
}
```

Error codes

Error Code	Description
400 Bad Request	Required `time` parameter not found.
422 Unprocessable Entity	Invalid `time` value.
409 Conflict	Cannot set time when NTP is enabled.

2.4. Uptime of the device

The uptime shows how long the device has been running (in seconds).

URL

`/api/device/uptime`

Method

`GET`

Response Body

```
{
  "uptimeInSeconds": "85094"
}
```

3. Images

The API endpoint `/api/image` allows to retrieve a static image from the device.

3.1. Getting the current image

Gets the current image from the device as a JPEG file.

URL

`/api/image/current`

Method

`GET`

Response Body

A JPEG image (With media type `image/jpeg`)

Error codes

Error Code	Description
403 Forbidden	Retrieving images is not supported by the device.

3.2. Getting the image for an event

You can retrieve an image with for a certain event by its event number as long as it is available on the device. For retrieving events, see section Events.

URL

`/api/image/event/:eventNumber`

Method

`GET`

Parameters

- `eventNumber`: required - the event number that is contained in the event message.

Example Request

`http://<device_ip_address>/api/image/event/1234567`

Response Body

A JPEG image (with media type `image/jpeg`)

Error Code	Description
400 Bad Request	Invalid or missing event number.
403 Forbidden	Retrieving images is not supported by the device.
404 Not Found	The image with given event number cannot be found or is not available anymore.

4. Videos

The API endpoint `/api/video` allows to retrieve a video file from the device.

4.1. Getting the video for an event

You can retrieve a video with for a certain event by its event number as long as it is available on the device. Pre and post history of the video around the event can be specified. For retrieving events, see section Events.

URL

`/api/video/event/:eventNumber?pre=:preHistorySeconds&post=:postHistorySeconds`

Method

`GET`

Parameters

- `eventNumber`: required - the event number that is contained in the event message.
- `pre`: required - the pre history in seconds.
- `post`: required - the post history in seconds.

Example Request

`http://<device_ip_address>/api/video/event/1234567?pre=60&post=30`

Response Body

A video file (with media type `video/avi`)

Error Code	Description
400 Bad Request	Invalid or missing event number, pre history or post history.
403 Forbidden	Retrieving videos is not supported by the device.

Error Code	Description
404 Not Found	The video with given event number cannot be found.
410 Gone	The video with given event number, pre history and post history is not available anymore.
409 Conflict	The video with given event number, pre history and post history is not yet available.

5. Events

Events are notifications about certain traffic or technical situations that have been detected by the device.

5.1. Structure of an event

This is how a typical event will look:

```
{
  "messageType":"Event" ①
  "time": "2015-01-09T16:15:39.117+01:00", ②
  "type": "SpeedAlarm", ③
  "level": 2, ④
  "zoneId": 1, ⑤
  "eventNumber": 16, ⑥
  "state": "Begin" ⑦
}
```

① `messageType`: required - states that this message is an event message.

② `time`: required - the time of the event in ISO8601 format.

③ `type`: required - the type of the event. See Appendix A - Event Types.

④ `level`: optional - indicates the level of the event. Only events that have a level will have this element.

⑤ `zoneId`: optional - Id of the zone on which the event is generated as defined in the configuration.

⑥ `eventNumber`: required - the event number is a unique sequential number to identify the event. It can be used to retrieve an event image. See Getting the image for an event.

⑦ `state`: optional - the state of the event. This can be `Begin` or `End`. Stateless events will not have this element. See Appendix B - Event States.

When the event is stateful, it has a `Begin` state when it starts and an `End` state when it is done. This is an example `End` event message:

```
{
  "messageType":"Event"
  "time":"2015-01-09T16:16:31.123+01:00",
  "type":"SpeedAlarm",
  "level":"2",
  "zoneId":"1",
  "eventNumber":"17",
  "beginEventNumber":"16" ①
  "state":"End"
}
```

① `beginEventNumber`: required (for end events) - this shows that this `End` corresponds to the `Begin` with `eventNumber` 16. This makes it possible to correlate `Begin` and `End` messages.

Event numbers are reset when the device reboots. They cannot be used as universally unique identifiers.

5.2. List of supported event types

Not every device has the same functionality and as such, cannot generate each event type supported by the protocol. This command allows to retrieve the list of event types that are supported by the device.

See Appendix A - Event Types for all possible event types. See Appendix B - Event States for all possible event states.

URL

`/api/events/supported`

Method

`GET`

Response Body

```
[
  {
    "eventType":"Presence"
  },
  {
    "eventType":"BadVideo"
  },
  {
    "eventType":"BadPresenceQuality"
  },
  {
    "eventType":"PresenceLevel"
  },
  {
    "eventType":"SpeedAlarm"
  }
]
```

5.3. Getting the currently open events

Returns a list of events which are currently open.

URL

`/api/events/open`

Method

`GET`

Response Body

```
[
  {
    "eventNumber":"2",
    "messageType":"Event",
    "state":"Begin",
    "time":"2015-06-05T11:37:59.230+02:00",
    "type":"Presence",
    "zoneId":"2"
  },
  {
    "eventNumber":"3",
    "level":"1",
    "messageType":"Event",
    "state":"Begin",
    "time":"2015-06-05T11:38:13.470+02:00",
    "type":"PresenceLevel",
    "zoneId":"3"
  }
]
```

The response will contain 0 or more events in the `Begin` state.

6. Traffic Data

Traffic data is a periodical message about the detected traffic conditions. It can include things like number of vehicles, average speed…

6.1. Structure of traffic data

The structure of a traffic data message depends on the type of data that is sent. You can find an overview of the different types in Appendix C - Traffic Data Types.

```
{
  "dataNumber":"42", ①
  "intervalTime":"10", ②
  "messageType":"Data", ③
  "time":"2015-06-05T13:01:10.038+02:00", ④
  "type":"FlowSpeedData" ⑤
}
```

① `dataNumber`: required - an unique sequential number to identify the data.

② `intervalTime`: required - the interval time.

③ `messageType`: required - states that this message is an data message.

④ `time`: required - the time of the data in ISO8601 format.

⑤ `type`: required - the type of the data. See Appendix C - Traffic Data Types.

6.2. List of supported traffic data types

Not every device has the same functionality and as such, cannot generate each traffic data type supported by the protocol. This command allows to retrieve the list of traffic data types that are supported by the device.

URL

`/api/data/supported`

Method

`GET`

Response Body

```
[
  {
    "dataType": "IntegratedData"
  },
  {
    "dataType": "FlowSpeedData"
  }
]
```

6.3. Get Data

Retrieve the data between the open interval `]beginTime, endTime[` (not including `beginTime` and `endTime`).

The results will be paginated: When there are more than the allowed maximum number of results, the response will contain a `nextDataUrl` attribute. `nextDataUrl` contains the URL where the next part of the data can be retrieved with. You can request all data by executing consecutive GetData calls until a response does not contain `nextDataUrl` anymore.

Each query parameter should be URL encoded.

URL

`/api/data?begintime=:beginTime&endtime=:endtime`
`/api/data/:dataType?begintime=:beginTime&endtime=:endtime`

Method

`GET`

Parameters

- `begintime`: optional - the begin time in URL encoded ISO8601 format.
- `endtime`: optional - the end time in URL encoded ISO8601 format.
- `dataType`: optional - the type of data to filter on. When not provided, all data types will be returned.

Example request

```
http://<device_ip_address>/api/data?begintime=2015-06-04T11%3A56%3A40.009%2B02%3A00
```

Response Body

```
{
  "data":[
    {
      "dataNumber":"5",
      "intervalTime":"60",
      "messageType":"Data",
      "time":"2015-06-04T11:57:00.040+02:00",
      "type":"IntegratedData",
      "zone":[
        {
          "class":[
            {
              "classNr":"1",
              "gapTime":"78",
              "gapTimeSq":"12050",
              "numVeh":"5",
              "speed":"101",
              "speedSq":"10139"
            },
            {
              "classNr":"3",
              "gapTime":"28",
              "gapTimeSq":"830",
              "numVeh":"3",
              "speed":"91",
              "speedSq":"8409"
            }
          ],
          "confidence":"9",
          "density":"6",
          "headWay":"153",
          "headWaySq":"55422",
          "length":"91",
          "occupancy":"6",
          "zoneId":"1"
        },
        {
          "class":[
            {
              "classNr":"1",
              "gapTime":"69",
              "gapTimeSq":"7385",
              "numVeh":"8",
              "speed":"110",
              "speedSq":"12273"
            }
          ],
            "confidence":"8",
            "density":"5",
            "headWay":"196",
            "headWaySq":"63657",
            "length":"61",
            "occupancy":"3",
            "zoneId":"2"
          }
        ]
      }
    ],
    "nextDataUrl":"/api/data?begintime=2015-06-04T12%3A18%3A00.024%2B02%3A00" ①
  }
```

① nextDataUrl: optional - the relative URL where the next part of the data can be retrieved with.

7. Subscriptions

7.1. WebSockets

Data and events can be received by using a WebSocket.

First you create a WebSocket with url `ws://<device_ip_address>/api/subscriptions`. Then you can send commands over the WebSocket to subscribe or unsubscribe for events and data.

The same API can be used for both data and events. The general structure of a subscription message is as follows:

```
{
  "messageType":"Subscription", ①
  "subscription":{
    "type":"Event", ②
    "action":"Subscribe" ③
  }
}
```

① `messageType`: required - states that this message is a subscription message.

② `type`: required - `Event` or `Data`.

③ `action`: required - `Subscribe` or `Unsubscribe`.

7.2. Subscribe

Send a subscription over the WebSocket:

```
{
  "messageType":"Subscription",
  "subscription":{
    "type":"Event",
    "action":"Subscribe"
  }
}
```

Response

```
{
  "messageType":"Subscription",
  "subscription":{
    "returnValue":"OK",
    "type":"Event"
  }
}
```

You will now start to receive events as they happen over the WebSocket. For example:

```
{
  "eventNumber":"2",
  "messageType":"Event",
  "state":"Begin",
  "time":"2015-06-05T12:14:56.112+02:00",
  "type":"Presence",
  "zoneId":"4"
}
```

7.2.1. Inclusions

If you only want to get informed about certain event types, you can use an inclusion in the subscription:

```
{
  "messageType":"Subscription",
  "subscription":{
    "type":"Event",
    "action":"Subscribe",
    "inclusions":[
      {
        "type":"Presence"
      },
      {
        "type":"SpeedAlarm"
      }
    ]
  }
}
```

Response

```
{
  "messageType":"Subscription",
  "subscription":{
    "returnValue":"OK",
    "type":"Event"
  }
}
```

In this way only events of type `Presence` and `SpeedAlarm` will be sent over the socket.

7.2.2. Exclusions

If you want to get informed about most event types, except some, then it is easier to use exclusions:

```
{
  "messageType": "Subscription",
  "subscription": {
    "type": "Event",
    "action": "Subscribe",
    "exclusions": [
      {
        "type": "Presence"
      },
      {
        "type":"SpeedAlarm"
      }
    ]
  }
}
```

Response

```
{
  "messageType":"Subscription",
  "subscription":{
    "returnValue":"OK",
    "type":"Event"
  }
}
```

Now all events except `Presence` and `SpeedAlarm` events will be sent over the socket.

7.3. Unsubscribe

Send an unsubscription message over the websocket.

```
{
  "messageType":"Subscription",
  "subscription":{
    "type":"Event",
    "action":"Unsubscribe"
  }
}
```

Response

```
{
  "messageType":"Subscription",
  "subscription":{
    "returnValue":"OK",
    "type":"Event"
  }
}
```

7.4. Keepalive

You can optionally send keepalive messages to the device to check if it is still alive.

```
{
  "messageType":"KeepAlive"
}
```

Response

```
{
  "messageType":"KeepAlive",
  "keepAlive":{
    "returnValue":"OK"
  }
}
```

7.5. ForceEvent

For debugging you can force the device to generate an event. For more information on events, see section Events.

```
{
  "messageType":"ForceEvent",
  "forceEvent":{
    "type":"BadVideo",
    "state":"Begin",
    "level":2,
    "zoneId":1
  }
}
```

Response

```
{
  "messageType":"ForceEvent",
  "forceEvent":{
    "returnValue":"OK"
  }
}
```

Appendix A - Event Types

An overview of the general structure of an event can be found in section Events.

⚠ Not all event types are available on all devices! See List of supported event types for more information.

Traffic Events

Bad Presence Quality

Quality of the video is too bad to do detection.

```
{
  "eventNumber":"103375",
  "messageType":"Event",
  "state":"Begin",
  "time":"2012-11-08T16:32:03.465-03:00",
  "type":"BadPresenceQuality"
}
```

Bicycle Count

Stateless event triggered every time a bicycle leaves the zone.

```
{
  "eventNumber":"103375",
  "frameCounter":"8192406", ①
  "messageType":"Event",
  "time":"2012-11-08T16:32:03.465-03:00",
  "type":"BicycleCount",
  "zoneId":"2"
}
```

① `frameCounter`: optional - field-counter when codec is enabled in the configuration.

Bicyle Presence

```
{
  "eventNumber":"103375",
  "messageType":"Event",
  "state":"Begin",
  "time":"2012-11-08T16:32:03.465-03:00",
  "type":"BicyclePresence",
  "zoneId":"2"
}
```

Day Night

Beginning of Night.

```
{
  "eventNumber":"6",
  "messageType":"Event",
  "state":"Begin",
  "time":"2014-12-02T16:44:06.830-03:00",
  "type":"DayNight"
}
```

Dilemma Zone

One or more vehicles are in the dilemma zone.

```
{
  "eventNumber":"103375",
  "messageType":"Event",
  "state":"Begin",
  "time":"2012-11-08T16:32:03.465-03:00",
  "type":"DilemmaZone",
  "zoneId":"2"
}
```

Fallen Object

A fallen object was detected.

```
{
  "eventNumber":"103375",
  "messageType":"Event",
  "position":{ ①
    "x":"20",
    "y":"30"
  },
  "time":"2012-11-08T16:32:03.465-03:00",
  "type":"FallenObject",
  "zoneId":"2"
}
```

① `position`: required - position of the vehicle triggering the fallen object.

Over Speed

```
{
  "eventNumber":"3",
  "level":"2",
  "messageType":"Event",
  "state":"Begin",
  "time":"2015-06-06T16:44:02.958+02:00",
  "type":"OverSpeed",
  "zoneId":"1"
}
```

Pedestrian

```
{
  "eventNumber":"103375",
  "messageType":"Event",
  "state":"Begin",
  "time":"2012-11-08T16:32:03.465-03:00",
  "type":"Pedestrian",
  "zoneId":"2"
}
```

Presence

```
{
  "eventNumber":"103375",
  "messageType":"Event",
  "state":"Begin",
  "time":"2012-11-08T16:32:03.465-03:00",
  "type":"Presence",
  "zoneId":"2"
}
```

Presence Count On Red

Stateless event triggered every time a vehicle leaves the presence zone while the linked RYG channel is red.

```
{
  "eventNumber":"6842",
  "messageType":"Event",
  "time":"2017-02-08T16:32:03.465-03:00",
  "type":"PresenceCountOnRed",
  "zoneId":"2"
}
```

Presence Level

```
{
  "eventNumber":"93",
  "level":"1",
  "messageType":"Event",
  "state":"Begin",
  "time":"2015-06-23T09:36:13.260-03:00",
  "type":"PresenceLevel",
  "zoneId":"4"
}
```

PTZ Preset

Indicates the current preset position for a PTZ camera.

```
{
  "eventNumber":"1",
  "level":"0",
  "messageType":"Event",
  "state":"Begin",
  "type":"PtzPreset"
}
```

Queue

The vehicle threshold in the queue is reached.

```
{
  "eventNumber":"15",
  "messageType":"Event",
  "state":"Begin",
  "time":"2015-06-06T18:17:19.200+02:00",
  "type":"Queue",
  "zoneId":"1"
}
```

Radar Presence

Presence of one or more vehicles based on radar input.

```
{
  "eventNumber":"103375",
  "messageType":"Event",
  "radarPresenceZoneId":"1", ①
  "state":"Begin",
  "time":"2012-11-08T16:32:03.465-03:00",
  "type":"RadarPresence",
  "zoneId":"2"
}
```

① `radarPresenceZoneId`: Id of the radar presence zone (loop) in the 'region'.

Smoke

```
{
  "eventNumber":"103375",
  "messageType":"Event",
  "state":"Begin",
  "time":"2012-11-08T16:32:03.465-03:00",
  "type":"Smoke"
}
```

Speed Alarm

```
{
  "beginEventNumber":"16",
  "eventNumber":"17",
  "level":"2",
  "messageType":"Event",
  "state":"End"
  "time":"2015-01-09T16:16:31.123+01:00",
  "type":"SpeedAlarm",
  "zoneId":"1"
}
```

Speed Drop

```
{
  "eventNumber":"4",
  "level":"2",
  "messageType":"Event",
  "state":"Begin",
  "time":"2015-06-06T16:45:14.699+02:00",
  "type":"SpeedDrop",
  "zoneId":"1"
}
```

Stop

```
{
  "eventNumber":"103375",
  "messageType":"Event",
  "position":{ ①
    "x":"20",
    "y":"30"
  },
  "time":"2012-11-08T16:32:03.465-03:00",
  "type":"Stop",
  "zoneId":"2"
}
```

① `position`: required - position of the vehicle triggering the stop event.

Under Speed

```
{
  "eventNumber":"103375",
  "messageType":"Event",
  "speed":"12",
  "time":"2012-11-08T16:32:03.465-03:00",
  "type":"Underspeed",
  "zoneId":"2"
}
```

Wrong Way Driver

```
{
  "eventNumber":"103375",
  "messageType":"Event",
  "position":{ ①
    "x":"20",
    "y":"30"
  },
  "time":"2012-11-08T16:32:03.465-03:00",
  "type":"WrongWayDriver",
  "zoneId":"2"
}
```

① `position`: required - position of the vehicle triggering the wrong way driver event.

Technical Events

Bad Video

Video contrast dropped below threshold.

```
{
  "eventNumber":"103375",
  "messageType":"Event",
  "state":"Begin",
  "time":"2012-11-08T16:32:03.465-03:00",
  "type":"BadVideo"
}
```

Configuration

Sent when a device has received a new configuration.

```
{
  "eventNumber":"96",
  "messageType":"Event",
  "returnValue":"OK",
  "time":"2015-06-27T02:02:51.012-03:00",
  "type":"Configuration"
}
```

FirmwareUpdate

Active during firmware update.

```
{
  "eventNumber":"103",
  "messageType":"Event",
  "state":"Begin",
  "time":"2015-06-27T04:06:57.012-03:00",
  "type":"FirmwareUpdate"
}
```

Input

Sent when a device input changes state.

```
{
  "eventNumber":"5",
  "level":"2",
  "messageType":"Event",
  "state":"Begin",
  "time":"2015-06-06T16:46:10.069+02:00",
  "type":"Input",
  "zoneId":"1"
}
```

No Video

Sent when a device loses video input (only for devices with external video input).

```
{
  "eventNumber":"6",
  "messageType":"Event",
  "state":"Begin",
  "time":"2015-06-06T16:48:32.434+02:00",
  "type":"NoVideo",
}
```

Temperature

The device temperature is above the configured threshold.

```
{
  "eventNumber":"7",
  "messageType":"Event",
  "state":"Begin",
  "time":"2015-06-06T17:00:05.094+02:00",
  "type":"Temperature",
}
```

Power Drop

Sent when the power drops for a short period without reboot.

```
{
  "eventNumber":"10",
  "messageType":"Event",
  "state":"Begin",
  "time":"2015-06-06T18:07:16.810+02:00",
  "type":"PowerDrop",
}
```

Fuse Blown

Sent by interface boards when the fuse of one of the connections is blown.

```
{
  "eventNumber":"15",
  "messageType":"Event",
  "state":"Begin",
  "time":"2015-06-06T18:17:19.200+02:00",
  "type":"FuseBlown",
  "value": "1" ①
}
```

① `value`: required - fuse number.

Remote Device Connected

Sent by BPL communication boards when connected with a remote device. Stays open until disconnected.

```
{
  "eventNumber":"15",
  "ip":"172.17.17.16", ①
  "mac":"00:0C:C6:78:2E:75", ②
  "messageType":"Event",
  "state":"Begin",
  "time":"2015-06-06T18:07:16.810+02:00",
  "type":"RemoteDeviceConnected"
}
```

① `ip`: required - the IP address of the remote device.

② `mac`: required - the MAC address of the remote device.

Appendix B - Event States

`Begin`
: Indicates that this event is a stateful event and it has just started.

`End`
: Indicates that this event is the closing part of the stateful event.

Stateless events don't have a state attribute.

Appendix C - Traffic Data Types

An overview of the general structure of a data item can be found in section Traffic Data.

> ⚠ Not all data types are available on all devices! See List of supported traffic data types for more information.

Traffic Data

Individual Data

Individual data messages are sent each time the Traffic Data functionality detects a vehicle on one of its zones.

```
{
  "classification":"3", ①
  "confidence":"8", ②
  "dataNumber":"54",
  "detectorZone":"1", ③
  "gapTime":"224", ④
  "length":"230", ⑤
  "messageType":"Data",
  "sequenceNr":"2", ⑥
  "speed":"78", ⑦
  "time":"2010-11-29T12:25:15.315-03:00",
  "type":"IndividualData"
}
```

① `classification`: The detected vehicle class.

② `confidence`:The confidence of the detection on 10 points.

③ `detectorZone`: The zone ID that generated this data.

④ `gapTime`: The gap time since the last detected vehicle (in 1/10 of a sec)

⑤ `length`: The detected vehicle length (in dm)

⑥ `sequenceNr`: A unique number identifying this Individual Data. a `sequenceNr` is unique for one product during the uptime of the product: if the product is reset, the numbering restarts at 1 and is incremented for each Individual Data message sent.

⑦ `speed`: the speed of the vehicle. Unit = km/h (no imperial units)

Integrated Data

For each set interval, integration of individual data is done and sent out.

```
{
  "dataNumber":"49",
  "intervalTime":"60",
  "messageType":"Data",
  "time":"2010-11-29T12:25:00.032-03:00",
  "type":"IntegratedData",
  "zone":[
    {
      "class":[
        {
          "classNr":"1", ①
          "gapTime":"78", ②
          "numVeh":"5", ③
          "speed":"101" ④
        },
        {
          "classNr":"3",
          "gapTime":"28",
          "numVeh":"3",
          "speed":"91"
        }
      ],
      "confidence":"9", ⑤
      "density":"6", ⑥
      "headWay":"153", ⑦
      "length":"91", ⑧
      "occupancy":"6", ⑨
      "zoneId":"1" ⑩
    },
    {
      "class":[
        {
          "classNr":"1",
          "gapTime":"69",
          "numVeh":"8",
          "speed":"110"
        }
      ],
      "confidence":"8",
      "density":"5",
      "headWay":"196",
      "length":"61",
      "occupancy":"3",
      "zoneId":"2"
    }
  ]
}
```

① `classNr`: the number identifying the class.

② `gapTime`: the gap time of the vehicles of this class, averaged over all vehicles of this class in this interval (NumVeh). Unit = decisecond (1/10th of a second)

③ `numVeh`: the number of vehicles of this class (that exited the zone) during the integration interval.

④ `speed`: the speed of the vehicles of this class, averaged over all vehicles of this class in this interval (NumVeh). Unit = km/h (no imperial units)

⑤ `confidence`: the average of the confidence of each individual vehicle, averaged over all vehicles in the integration interval.

⑥ `density`: The average amount of vehicles per km. i.e. the number of vehicles per time interval, divided by the mean speed. Unit = number of vehicles / km

⑦ `headWay`: the distance inbetween 2 succeeding vehicles, averaged over all vehicles in the integration interval. Unit = meter

⑧ `length`: the vehicle length, averaged over all vehicles in the integration interval. Unit = decimeter (no imperial units).

⑨ `occupancy`: the percentage of the integration interval a point (at the zone exit) is occupied by a vehicle.

⑩ `zoneId`: the Id of the zone that generated the individual data. Range = 1 to 8 (8 being the maximum number of zones).

Flow Speed Data

For each set interval the current flow speed and zone occupancy is sent out. This is not averaged or otherwise integrated, but the current values, since those values are already integrated.

```
{
  "dataNumber":"742315",
  "intervalTime":"60",
  "messageType":"Data",
  "time":"2010-02-17T09:25:47.000-03:00",
  "type":"FlowSpeedData",
  "zone":[
    {
      "count":"3", ①
      "flowSpeed":"92", ②
      "zoneId":"1", ③
      "zoneOccupancy":"15" ④
    },
    {
      "count":"1",
      "flowSpeed":"101",
      "zoneId":"2",
      "zoneOccupancy":"12"
    },
    {
      "count":"4",
      "flowSpeed":"90",
      "zoneId":"3",
      "zoneOccupancy":"0"
    }
  ]
}
```

① `count`: Number of vehicles at this time interval.

② `flowSpeed`: The current flow speed value for this zone. Unit = km/h (no imperial units). This is the current averaged speed of the flow (rougly the same as the average speed of the last x vehicles).

③ `zoneId`: The zone identifier (is always > 0).

④ `zoneOccupancy`: The current zone occupancy of the zone (also called Zocc). Unit = %. It indicates the % that the zone was occupied by vehicles. In other words a Zocc of 10% for example means that the zone is occupied by a vehicle for 10%, so when monitoring the zone for about 60 seconds it means that we have 6 seconds where a vehicle is in the zone.

Presence Data

```
{
  "dataNumber":"1",
  "intervalTime":"10",
  "messageType":"Data",
  "time":"2012-01-03T13:57:40.092-03:00",
  "type":"PresenceData",
  "zone":[
    {
      "numVeh":"27", ①
      "zoneId":"1", ②
      "zoneOccupancy":"78" ③
    },
    {
      "numVeh":"4",
      "zoneId":"2",
      "zoneOccupancy":"99"
    }
  ]
}
```

① `numVeh`: The number of vehicles.

② `zoneId`: The zone ID that generated this data.

③ `zoneOccupancy`: The current zone occupancy (in %).

Bicycle Data

```
{
  "intervalTime":"10",
  "messageType":"Data",
  "time":"2012-01-03T13:57:40.092-03:00",
  "type":"BicycleData",
  "zone":[
    {
      "numVeh":"27", ①
      "zoneId":"1" ②
    },
    {
      "numVeh":"4",
      "zoneId":"2"
    }
  ]
}
```

① `numVeh`: The number of bicycles.

② `zoneId`: The zone ID that generated this data.