# Vehicle Information Messaging Interface

# Document history

| | | | |
|---|---|---|---|
| PA01 | 2015-05-13 | HLN | Initial version. |
| PA02 | 2015-05-18 | HLN | Added description of parameters.<br>/vimi/pis/sensor/temperature/outdoor/main<br>• Added timestamp.<br>/vimi/apc/sensor/onboardcount<br>• Added timestamp. |
| PA03 | 2015-05-20 | HLN | Added section "Appendix - APC Train Reporting Interface 1.0 - Interface Specification". |
| PA04 | 2015-05-21 | HLN | Added section "Appendix - APC Bus Reporting Interface 1.0 - Interface Specification".<br>Added APC report examples for the topic /vimi/report-gateway/send/<app> |
| PA05 | 2015-05-22 | CJA | Added approval from Skånetrafiken and Skånetrafiken logotyp |
| PA06 | 2015-05-25 | HLN | Added section "Onboard device". |
| PA07 | 2015-05-27 | HLN | Specified local time zone in section "Conventions and guidelines".<br>Added abbreviation for "UTC".<br>Removed trainNumber from /vimi/pis/identity/vehicleinfo<br>Added trainNumber to /vimi/pis/route/journey. |
| PA08 | 2015-06-16 | HLN | For all topics; added responsible application for publishing and subscription.<br><br>/unit/<provider>/< name/label >/<mac>/version<br>• Specified that the MAC address shall be used.<br><br>/vimi/pis/route/journey<br>• Added more info which fields that are available on bus and train respectively.<br><br>/vimi/report-gateway/send/<app><br>• More info regarding the seq number.<br><br>/vimi/report-gateway/send/<app><br>• Added sub section"APC Report validation".<br>• Added more details regarding APC reports.<br><br>/vimi/report-gateway/res/<app><br>• Added field "result" in order to typedef errors (needed due to the APC Report validation).<br>/vimi/apc/command/resetonboardcount<br>• New topic. |
| PA09 | 2015-06-18 | HLN | Administrative: Fixed broken cross references. |
| PA10 | 2015-06-25 | HLN | /vimi/apc/command/resetonboardcount<br>• Retain = false.<br><br>/vimi/pis/assignment/block |

| | | | |
|---|---|---|---|
| | | | • New topic.<br><br>/vimi/pis/assignment/vehicle_journey<br>• New topic.<br><br>Changed syntax in Bus APC report (Upper->Lower case changes) to follow same rules as in Train APC report.<br><br>Added type to Bus APC report.<br><br>Added example and notes of how to calculate xsum in Train APC report.<br><br>"Appendices" replaced by "Reports". |
| A<br>PA11 | 2015-07-07 | CJA | Added general comment about Sign on/Sign off on page 24<br><br>Corrected spelling of "timestamp" on page 26, 30, 31, 32, 34 and 35 |
| PB1<br>~~PA12~~<br>2.0.0 | 2016-02-18 | LMA | Topics organized after source<br><br>Corrected spelling and formatting errors<br><br>Added parameter naming guidelines and clarifications to "Conventions and Guidelines"<br><br>Added "Background" chapter<br><br>Added "Security" chapter<br><br>Added "Versioning" chapter<br><br>Added "Watchdog" chapter<br><br>Added "Partners" section<br><br>/unit/<provider>/<name/label>/<mac>/version<br>• Aligned parameter name presentationName with guidelines<br><br>/unit/<provider>/<name/label>/<mac>/state<br>• Added error object<br>• Aligned parameter names for epochNow, uptimeStart, uptimeNow and publishInterval with guidelines<br><br>/vimi/pis/identity/vehicleinfo moved to /vimi/system/identity/info<br><br>GPS:<br>• Moved to /vimi/system/sensor/gps<br>• Added "direction", "numberSatellites" and "valid" parameters<br>• Changed date and time parameters to "datetime" object |

| | | | |
|---|---|---|---|
| | | | Added /vimi/system/status/event<br><br>Added Modem topic /vimi/modem/sensor/signal<br><br>Added VoIP topics<br>• /vimi/voip/assignment/vehicle_journey<br>• /vimi/voip/call/event<br>• /vimi/voip/call/ongoing<br>• /vimi/voip/message/statusmessage<br><br>Added Driver identification topics /vimi/driverid/user/event and /vimi/driverid/command/signoff<br><br>Added Internet Onboard topic /vimi/iob/connection/active_devices<br><br>Added Seat Detection topics /vimi/seatdetection/sensor/data and /vimi/seatdetection/sensor/occupancy<br><br>Added status reports to /vimi/report-gateway/send/<app><br><br>Added "event=passage" as a condition of an APC report in /vimi/report-gateway/send/<app><br><br>Added "messageId" to APC train reports<br><br>Added "Bus status reports" to Reports chapter |
| PB2<br>2.0.0<br>rev. 2 | 2016-04-06 | LMA | Updated Skånetrafiken template and colors<br><br>Corrected revision history<br><br>Corrected typo in section "Background"<br><br>Removed definition of "Availability" from /vimi/report-gateway/send/apc, since it is not related to the messaging interface<br><br>Added clarification of boarding/alighting passengers between stops to /vimi/report-gateway/send/apc |
| PB3<br>2.1.0 | 2016-04-25 | LMA | Added Ticket Validation topic /vimi/ticketing/validator/ticket_validation<br><br>Added missing bracket in example for topic /vimi/seatdetection/sensor/occupancy<br><br>Added text "This topic is preliminary and may be subject to change." To Seat Detection topics<br><br>Corrected Bus Status Report example |

| | | | |
|---|---|---|---|
| | | | Added "Bus seat detection report" to Reports chapter<br><br>Added occupancy reports to /vimi/report-gateway/send/<app><br><br>Added occupancy reports to /vimi/report-gateway/recieve/<app><br><br>Updated Telia logo in the "Partners" section |
| PB4 2.1.1 | 2016-04-29 | LMA | Corrected `type/name` to `name/label` in topic description for /unit/<provider>/<name/label>/<mac>/version<br><br>New subtopic: /vimi/voip/message/textmessage<br><br>Removed `priority` from /vimi/voip/call/event<br><br>Corrected `Timestamp` to `timestamp` in /vimi/apc/sensor/onboardcount<br><br>Corrected `Route` to `route` in /vimi/pis/route/journey<br><br>/vimi/system/modem/<mac>/signal<br>• Moved /vimi/modem/sensor/signal to /vimi/system/modem/<mac>/signal<br>• Updated radio carrier descriptions<br>• Added parameters mcc, mnc, lac and cell id<br>• Updated and corrected example<br>• Rename of `signal` object to `modeminfo`<br>• Topic now contains a list of `modeminfo` objects<br><br>Renamed /vimi/iob/connection/active_devices to /vimi/iob/gateway/<mac>/active_connections<br><br>Added VOIP and IOB to abbreviations |
| PB5 2.1.2 | 2016-05-19 | LMA | New subtopic: /vimi/ticketing/identity/info<br><br>New subtopic: /vimi/ticketing/command/power<br><br>New subtopic: /vimi/pis/sensor/ignition/main<br><br>Added `zone` to /vimi/pis/route/journey, /vimi/pis/route/journey_point and /vimi/pis/route/progress<br><br>Added `datetimePlanned` to /vimi/pis/route/journey_point<br><br>Added TriNorth Solutions to Partners section<br><br>Corrected example for /vimi/iob/gateway/<mac>/active_connections |

| | | | |
|---|---|---|---|
| PB6<br>2.1.3 | 2016-09-19 | LMA | Clarified parameter descriptions in /unit/<provider>/<name/label>/<mac>/state<br><br>Added `datetimePlanned` to all stops in /vimi/pis/route/journey<br><br>Changed reporting rules for /vimi/report-gateway/send/apc |
| PB7<br>2.1.4 | 2016-12-01 | LMA | Added `operatingDayDate` to /vimi/pis/route/journey<br><br>Added `source` to /vimi/driverid/user/event<br><br>Added Infodev and Transdev to Partners section |
| PB8<br>2.1.5 | 2017-01-20 | LMA | Moved /vimi/report-gateway/sent/status to /vimi/report-gateway/send/statmon, added parameters `countSeen`, `countGone`, `epochFirstSeen`, `epochLastGone` and `active`. Changed reporting intervals.<br><br>Corrected description and example of `operatingDayDate` in /vimi/pis/route/journey<br><br>Added `type` and `status` to /vimi/ticketing/validator/ticket_validation/ |
| PB9<br>B<br>2.1.5<br>rev.2 | 2017-02-17 | LMA | Added Maintenance section |
| PC1<br>2.2.0 | 2017-08-22 | LMA | Added diagnostics topics<br>• /vimi/diagnostics/performance<br>• /vimi/diagnostics/flags<br><br>Added Drivec to Partners section<br><br>Added Nobina to Partners section<br><br>Added `timetableDeviation` to /vimi/pis/route/progress<br><br>Added `timingPoint` to /vimi/pis/route/journey and /vimi/pis/route/progress |
| PC2<br>2.2.0 | 2017-09-04 | LMA | Corrected formatting |
| PC3<br>2.2.0 | 2017-09-26 | LMA | Updated Drivec logotype<br><br>Added Mobitec to Partners section |
| PC4<br>2.2.1 | 2018-09-21 | LMA | Added PIS topic /vimi/pis/block/journeys<br><br>Added Blekingetrafiken to Partners section<br><br>Added Consat to Partners section |

# Abbreviations

| | |
|---|---|
| APC | Automatic Passenger Count. |
| PIS | Passenger Information System. |
| VIMI | Vehicle Information Message Interface. |
| VOIP | Voice over IP |
| UTC | Coordinated Universal Time. |
| IOB | Internet Onboard |

# Vehicle Information Messaging Interface (VIMI)

This document contains a specification of the vehicle information messaging interface for Public Transport. The interface is built on MQTT where we have the following components

- Message broker (server)
- Clients (subscribe/publish)

Data is ordered by ‘topics’ where clients can subscribe and publish data.  The format of the data is JSON encoded.

## Background

The VIMI specification has its origins at Fältcom 2014, whom in collaboration with Skånetrafiken developed the initial VIMI topic structure and implemented the VIMI system and required MQTT broker for the Fältcom MIIPS ® unit.

Its first usage was in collaboration with Skånetrafiken and Telia, under the Telia "Smart Public Transport/M2M" product portfolio. Usage of VIMI as a method of conveying information between onboard units has since been adapted by other vendors of onboard equipment such as Dilax, MultiQ and Swarco.

## Maintenance

The VIMI specification is currently maintained at Skånetrafiken by Love Månsson, love.mansson@skanetrafiken.se. Any proposed additions or changes to the specification is identified by any partner/partners, and the updating of the specification is done by Skånetrafiken in collaboration with the partner/partners.

Going forward, the intention is that a neutral organization will take ownership of and handle the maintenance and updating of the VIMI specification.

## Onboard device

At the vehicle, the onboard device is a Fältcom MIIPS ®, and this is the device that runs the VIMI. Specific details on how to connect to the MQTT broker on the onboard device, subscribing to topics etc. are covered in a separate document for the device in question. VIMI itself supports any selection of device platform, as long as it implements a MQTT broker using the security measures mentioned in the Security section below.

## Conventions and guidelines

In general, a MQTT subscriber must assume that any JSON parameter is optional and its value might be null (if not available).

Topic names should be declared in small case with underscores, e.g. /vimi/pis/assignment/vehicle_journey, and key names should be declared in CamelCase, e.g. trainSetId or doorActivities.

A MQTT publisher shall always use the 'RETAIN' flag unless otherwise specified. This tells the MQTT Broker to remember the last message for a topic and new subscribers will get a copy of the "last published message" when they connect. In the cases where the 'RETAIN' flag is used, but the last message is faulty or no longer valid, the publisher can opt to publish an empty message to ensure that the last message does not contain faulty or invalid information.

Charset for JSON data is UTF-8.

Numbers are specified according to the JSON standard[1].

For all topics a client may add additional fields in the JSON data as long as the field name begins with `"vend-"`

GPS positions are according to the WGS84 coordinate system in decimal format unless otherwise specified.

Local time zone is CET (Central European Time) unless otherwise specified.

## Security

Only trusted units shall be able to connect to the MQTT broker and publish/receive topics. An ACL (Access Control List) is used to control which units are allowed to publish each topic.

In general, all units should be able to subscribe to every topic unless stated otherwise in the topic description, but publishing topics should be restricted to the relevant topics for that unit (for instance APC would be limited to publishing /vimi/apc/ topics, apc reports in the report gateway and /unit/<vendor>/apc/-topics).

## Versioning

The VIMI specification is versioned to better keep track of compatibility and changes. Since a new revision does not necessarily break backwards compatibility, this is reflected in the versioning scheme.

The version of the VIMI interface will be in the form "X.Y.Z revision R" where the "revision R" part is optional depending on whether or not any revisions of the documentation has been made.

**X** Major update: One or more topics no longer backwards compatible or other major changes to the specification.

**Y** Update: New top level topics (e.g. /vimi/**iob** or /vimi/**apc**)

**Z** Minor update: New parameter in topic or new subtopics

**R** Documentation update or clarification

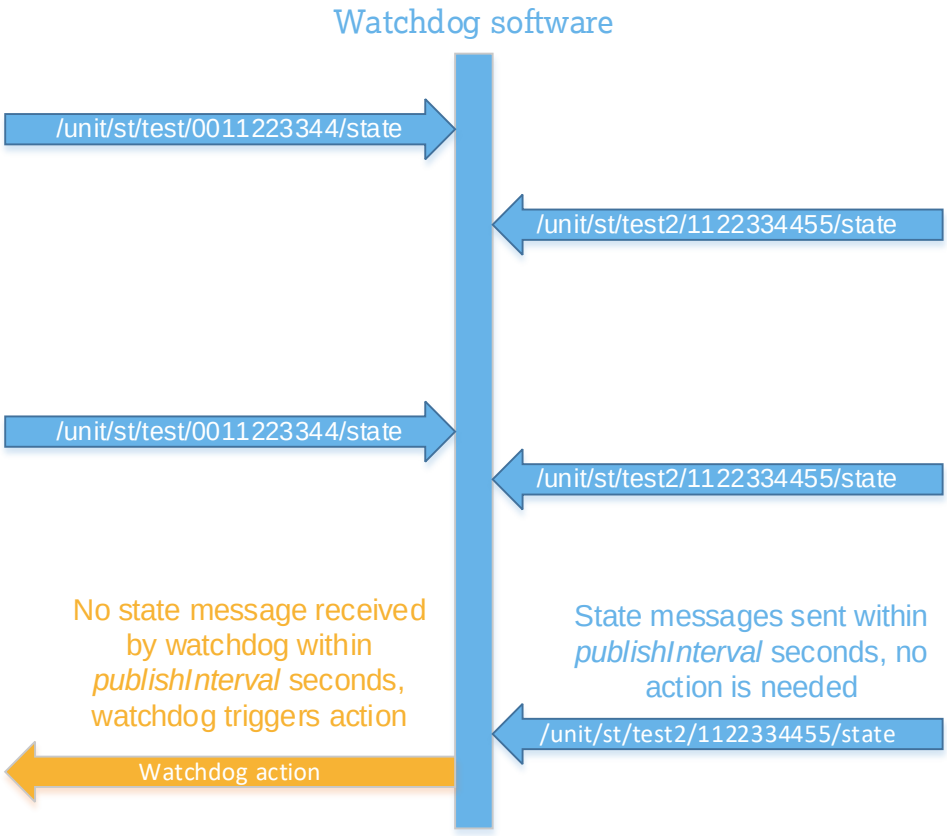
This version of the VIMI specification is **2.2.1**.

[1] http://www.json.org/

## Watchdog

The VIMI system application contains a watchdog, which will monitor the connected units (units which have registered with the /unit/<provider>/<name /label>/<mac>/version message) to ensure that actions are taken if the unit fails to provide its heartbeat message (/unit/<provider>/<name/label>/<mac>/state) within the time interval stated in the heartbeat message.

The actions taken when the watchdog is triggered should be specified by the unit supplier. Instructions on how to specify these actions are found in a separate document for the onboard device in question. The actions could be (but are not limited to) a reboot of the unit or execution of a specific subroutine.



# Topics

In general, the name of the topics is derived from the following logical structure

1. Global namespace (vimi)
2. Source in the system (e.g. pis)
3. Category/ Function (e.g. route)
4. Parameter/value (e.g. progress)

## Status reporting

This section contains topics related to the current status for each connected application and their versions.

### /unit/<provider>/<name/label>/<mac>/version

Published by each application connected to VIMI.

Unit info. Used for supervision.

Publish upon a new connect.

Example topic:

```
/unit/dilax/apc/001122334455/version
```

| Subtopic | Description |
|---|---|
| `<provider>` | Name of provider for this unit. |
| `<name/label>` | Some kind of type or name that tells which kind of device this is. |
| `<mac>` | The MAC address of the unit. This should be specified with only numbers and lower case letters. Examples: `0f23e45d67c8` – correct `0F:23:E4:5D:67:C8` – incorrect |

| Key | Description |
|---|---|
| `presentationName` | Public name used to announce/supervise the unit. |
| `versionName` | Version name. Up to the unit to format this string depending of its own versioning system. |

```
{
    "presentationName": "Name used to present this unit",
    "versionName": "2.3.5"
}
```

## /unit/<provider>/<name/label>/<mac>/state

Published by each application connected to VIMI.

Unit status published on a periodic basis (typically once every 5th minute). When an error occurs or an error code is cleared in a unit, the unit should publish a new status message as soon as possible, not waiting for the next scheduled heartbeat which could be minutes away.

This is the topic where one should use the MQTT "last will" functionality.

This topic is also used for units to send heartbeats.

See /unit/<provider>/<name/label>/<mac>/version for description of the subtopics.

| Key | Description |
|---|---|
| `status` | `"running"`: Unit is alive and running.<br>`"stopped"`: Unit is stopped.<br>`"dead"`: Disconnected (Last will). |
| `error` | Array of current errors/warnings. If no current error or warning, this can be omitted. |
| `code` | Error code (integer) |
| `severity` | Error severity. 0 indicates no error, 1 indicates minor error or warning, 2 indicates major error (such as faulty non-critical component), and 3 indicates a critical error or system halted |
| `message` | Additional information relevant to this error. Limited to 255 characters. |
| `epochNow` | Current unix epoch timestamp of the unit. |
| `uptimeStart` | Uptime when the unit application started (seconds). |
| `uptimeNow` | Current system uptime of the unit (seconds). |
| `publishInterval` | How often the unit will publish to this topic (seconds). |

```
{
    "status": "running",
    "error" : [
        {
            "code": 14,
            "severity": 1,
            "message": "Warning: Message buffer almost full"
        }
    ],
    "epochNow": 1431502390,
    "uptimeStart": 125,
    "uptimeNow": 3601,
    "publishInterval": 300
}
```

## System

This section contains the topics published by the VIMI system application.

### /vimi/system/identity/info

Published by the VIMI system application.

Holds information about the vehicle.

| Key | Description |
|---|---|
| `id` | The vehicle (or stop) identifier; typical 16 numerics |
| `type` | Type of vehicle (or stop)<br>`vehicleId`<br>`trainSetId`<br>`stopId`<br>`tramSetId`<br>`ferryId` |

```
{
    "id": "0011223344556677",
    "type": "vehicleId"
}
```

## /vimi/system/modem/<mac>/signal

Published by the VIMI system application.

Radio quality and information from the modems installed. Typically updated every few seconds but may vary from device to device.

MQTT Retain: false.

| Subtopic | Description |
|---|---|
| `<mac>` | The MAC address of the unit. This should be specified with only numbers and lower case letters. Examples:<br>`0f23e45d67c8` – correct<br>`0F:23:E4:5D:67:C8` – incorrect |

| Key | Description |
|---|---|
| `modeminfo` | Modem information object. |
| `identifier` | Modem identifier |
| `quality` | Radio quality (CSQ), number between 0 and 31 where 31 indicates highest quality |
| `carrier` | Radio carrier:<br>`gprs`<br>`edge`<br>`hsdpa`<br>`hsupa`<br>`hsdpa+`<br>`lte` |
| `mcc` | Mobile country code |
| `mnc` | Mobile network code |
| `lac` | Local area code |
| `cellId` | Cell ID |
| `datetime` | Datetime object. See /vimi/system/sensor/clock |

```
{
    "modeminfo": [
        {
            "identifier": "Modem 1",
            "quality": 27,
            "carrier": "hsdpa+",
            "mcc": 240,
            "mnc": 01,
            "lac": 2012,
            "cellId": 45201
        },
        {
            "identifier": "Modem 2",
            "quality": 18,
            "carrier": "lte",
            "mcc": 240,
            "mnc": 01,
            "lac": 2012,
            "cellId": 45201
        }
    ],
    "datetime": {
        "zone": "local",
        "date": "2014-02-03",
        "time": "18:31:46"
    }
}
```

## /vimi/system/sensor/clock

Published by the VIMI system application.

Used for time synchronization. This information origins from the device that holds the MQTT broker and is typically updated every 10-60th second.

MQTT Retain: false.

| Key | Description |
|---|---|
| `datetime` | Array for local and UTC date and time. |
| `zone` | Timezone: `"local"` or `"utc"`. |
| `date` | Date formatted as YYYY-MM-DD where<br>YYYY = year<br>MM = month [01..12]<br>DD = day [01..31] |
| `time` | Time formatted as hh:mm:ss where<br>hh = hour [00..23]<br>mm = minute [00..59]<br>ss = seconds [00..59] |

```
{
    "datetime": [
        {
            "zone": "local",
            "date": "2014-02-03",
            "time": "18:31:46"
        },
        {
            "zone": "utc",
            "date": "2014-02-03",
            "time": "16:31:46"
        }
    ]
}
```

## /vimi/system/sensor/gps/data

GPS position. Typically updated every second but may vary from device to device.

MQTT Retain: false.

| Key | Description |
|---|---|
| position | Position object. |
| latitude | Latitude. |
| longitude | Longitude. |
| datetime | Datetime object. See /vimi/system/sensor/clock |
| speed | Speed from GPS, km/h. |
| direction | Current direction, in degrees |
| numberSatellites | The number of currently available satellites |
| valid | Whether or not the GPS fix is deemed valid by the GPS device. |

```
{
    "position": {
        "latitude": 55.71433,
        "longitude": 13.21444,
        "datetime": {
            "zone": "local",
            "date": "2014-02-03",
            "time": "18:31:46"
        },
        "speed": 34.5,
        "direction": 125,
        "numberSatellites": 2,
        "valid": true
    }
}
```

## /vimi/system/status/event

Published by the VIMI system application.

Last generated status report. This topic shall be published at the same time the report is generated, independent of what's reported to the PT authority. The purpose with this topic is to have the report available in real-time onboard at the vehicle.

### Example bus status report


```
{
    "seq": 1432209505,
    "message": {
        "type": "Status",
        "vehicleRef": "9031012004507123",
        "timestamp": "2013-08-17T09:30:47+02:00",
        "messageId": "372,
        "statusMessages": [
            {
                "unit": "/unit/Dilax/apc/001122334455",
                "version": "1.20.0",
                "messages": [
                    {
                        "status": "running",
                        "error": [
                            {
                            "code": 14,
                            "severity": 1,
                            "message":"Warning: message buffer almost full"
                            }
                        ],
                        "epochNow": 1431502890,
                        "uptimeStart": 125,
                        "uptimeNow": 4401,
                        "publishInterval": 300
                    },
                    {
                        "status": "running",
                        "epochNow": 1431502390,
                        "uptimeStart": 125,
                        "uptimeNow": 3601,
                        "publishInterval": 300
                    }
                ]
            }
        ]
    }
}
```

# PIS

This section contains the topics published by the PIS application.

## /vimi/pis/sensor/clock

Published by the PIS application.

Used for time synchronization. This information origins from the PIS device and is typically updated every 10-60th second.

MQTT Retain: false.

| Key | Description |
|---|---|
| `datetime` | Array for local and UTC date and time. |
| `zone` | Timezone: `"local"` or `"utc"`. |
| `date` | Date formatted as YYYY-MM-DD where<br>YYYY = year<br>MM = month [01..12]<br>DD = day [01..31] |
| `time` | Time formatted as hh:mm:ss where<br>hh = hour [00..23]<br>mm = minute [00..59]<br>ss = seconds [00..59] |

```
{
    "datetime": [
        {
            "zone": "local",
            "date": "2014-02-03",
            "time": "18:31:46"
        },
        {
            "zone": "utc",
            "date": "2014-02-03",
            "time": "16:31:46"
        }
    ]
}
```

## /vimi/pis/sensor/door/main

Published by the PIS application.

Status of the doors in the vehicle. The scope is global and the purpose is to get information if any door is open or not.

| Key | Description |
|---|---|
| doorOpen | True if any door at the vehicle is open, else false. (Only at bus). |

```
{
    "doorOpen": false
}
```

## /vimi/pis/sensor/ignition/main

Published by the PIS application.

Status of the ignition of the vehicle. The scope is global and the purpose is to get information if the ignition is on or not.

| Key | Description |
|---|---|
| ignitionOn | True if the ignition is on, else false. (Only at bus). |

```
{
    "ignitionOn": false
}
```

## /vimi/pis/sensor/stopbutton/main

Published by the PIS application.

Status of the stop buttons in the vehicle. The scope is global and the purpose is to get information if any stop button is pressed or not.

| Key | Description |
|---|---|
| `stopPressed` | True if any stop button is pressed, else false.<br>Should be cleared to false when stopping at the station.<br>(Only at bus). |

```
{
    "stopPressed": false
}
```

## /vimi/pis/sensor/temperature/outdoor/main

Published by the PIS application if sensor available.

Outdoor temperature.

| Key | Description |
|---|---|
| `outdoorTemp` | Main outdoor temperature sensor from PIS device. Temperature in Celsius, decimal format. |
| `timestamp` | Unix epoch timestamp when the temperature was retrieved. |

```
{
    "outdoorTemp": 24.6,
    "timestamp": 1234
}
```

## /vimi/pis/assignment/block

Published by the PIS application.

Assignment event to inform that the vehicle has signed on or signed off to a block.

| Key | Description |
|---|---|
| `type` | `"signon"`: Sign on to a block.<br>`"signoff"`: Sign off to a block. |
| `datetime` | Local datetime object when the event occurred.<br>Please see /vimi/pis/sensor/clock for description. |
| `vehicleId` | The vehicle identifier (VehicleGid; typical 16 numerics).<br>Not available at train. |
| `trainSetId` | Train set ID. Not available at bus. |
| `blockId` | Block identifier of the block that we signed in or signed off to. (BlockGid) (typical 16 numerics). |

```
{
    "type": "signon",
    "datetime": {
        "zone": "local",
        "date": "2014-02-03",
        "time": "18:31:46"
    },
    "vehicleId": "0011223344556677",
    "blockId": "0011223344556677"
}
```

## /vimi/pis/assignment/vehicle_journey

Published by the PIS application.

Assignment event to inform that the vehicle has signed on or signed off to a journey.

| Key | Description |
|---|---|
| `type` | `"signon"`: Sign on to a journey.<br>`"signoff"`: Sign off to a journey. |
| `datetime` | Local datetime object when the event occurred.<br>Please see /vimi/pis/sensor/clock for description. |
| `vehicleId` | The vehicle identifier (VehicleGid; typical 16 numerics).<br>Not available at train. |
| `trainSetId` | Train set ID. Not available at bus. |
| `vehicleJourneyId` | Journey identifier for the journey we have signed on or signed off to. (VehicleJourneyGid) (typical 16 numerics). |

```
{
    "type": "signon",
    "datetime": {
        "zone": "local",
        "date": "2014-02-03",
        "time": "18:31:46"
    },
    "vehicleId": "0011223344556677",
    "vehicleJourneyId": "0011223344556677"
}
```

## /vimi/pis/block/journeys

Published by the PIS application.

Contains a list of journeys for the current block

| Key | Description |
|---|---|
| `blockId` | Block identifier of the block that we signed in or signed off to. (BlockGid) (typical 16 numerics). |
| `journeys` | Array with journeys belonging to the current block. |

Here follows a description of the journey:

| Key | Description |
|---|---|
| `vehicleJourneyId` | The current journey identifier. (VehicleJourneyGid) (typical 16 numerics). |
| `datetimeStart` | Planned time for journey start. |
| `datetimeEnd` | Planned time for journey end. |

```
{
    "blockId": "0011223344556677"
    "journeys": [
        {
            "vehicleJourneyId": 0011223344556677,
            "datetimeStart": {
                "zone": "local",
                "date": "2014-02-03",
                "time": "18:31:00"
            },
            "datetimeEnd": {
                "zone": "local",
                "date": "2014-02-03",
                "time": "19:29:00"
            }
        },
        {
            "vehicleJourneyId": 0011223344556788,
            "datetimeStart": {
                "zone": "local",
                "date": "2014-02-03",
                "time": "19:40:00"
            },
            "datetimeEnd": {
                "zone": "local",
                "date": "2014-02-03",
                "time": "20:20:00"
            }
        }
    ]
}
```

## /vimi/pis/route/journey

Published by the PIS application.

Contains a list of stops (and other points) for current journey.

| Key | Description |
|---|---|
| `vehicleJourneyId` | The current journey identifier. (VehicleJourneyGid) (typical 16 numerics). |
| `lineName` | Line name. Only at bus. |
| `lineNo` | Line number. Only at bus. |
| `destinationName` | Destination name. |
| `originName` | Origin name. |
| `routeNumber` | Route number. Only at train. |
| `trainNumber` | Train number. Only at train. |
| `operatingDayDate` | Planned date of the block which this journey belongs to. |
| `route` | Array with points on the route ("Route points), e.g. stop or link. Please see separate table for the description of the "route point" |

Here follows a description of the route point:

| Key | Description |
|---|---|
| `type` | Type of route point<br>`"link"`: when moving between stations/stops.<br>`"stop"`: when standing still at the stations/stops. |
| `name` | Name of the stop. |
| `id` | Identifier (JourneyPatternPointGid at bus). Only at bus. |
| `zone` | Zone in which the stop is located. This could be used for ticketing purposes. |
| `timingPoint` | True if stop is a timing point, false otherwise. Not available at train |
| `pointVisitCount` | Normally each stop is called only once during a vehicle journey, and then this attribute may be omitted. However, some vehicle journeys have such journey patterns that the same stop point is called a second time. In such instances this attribute could be included and set to the value 2 to mark that it is the second visit to the same stop. Not available at train. |
| `latitude` | Latitude. Not available at train. |
| `longitude` | Longitude. Not available at train. |
| `datetimePlanned` | Planned time for arrival |
| `length` | If link, this is the length (in meters) of the Link. See also "distance". Not available at train. |
| `distance` | Optional field to inform the distance to the next stop. May be used if the distance between the stops is available but link information is missing in the PIS device. On the other hand, if both sources is available from in the PIS device (and both are decided to be published), the sum of all links between the stop must |

be equal to the distance between the stops. Not available at train.


```
{
    "vehicleJourneyId": "0011223344556677",
    "lineName": "Busline 1",
    "lineNo": 1,
    "destinationName": "Hyllie",
    "originName": "Malmö C",
    "routeNumber": "180817073",
    "trainNumber": "1703",
    "operatingDayDate": "2014-02-03 00:00:00"
    "route": [
        {
            "type": "stop",
            "name": "Start station",
            "id": "1011223344556677",
            "zone": "250",
            "timingPoint": false,
            "pointVisitCount": 1,
            "latitude": 55.71433,
            "longitude": 13.21444,
            "datetimePlanned": {
                "zone": "local",
                "date": "2014-02-03",
                "time": "18:31:00"
            }
        },
        {
            "type": "link",
            "length": 650,
        },
        {
            "type": "stop",
            "name": "Station 1",
            "id": "2011223344556677",
            "zone": "250",
            "timingPoint": true,
            "pointVisitCount": 1,
            "latitude": 55.83433,
            "longitude": 13.24444,
            "datetimePlanned": {
                "zone": "local",
                "date": "2014-02-03",
                "time": "18:33:00"
            }
        },
        {
            "type": "link",
            "length": 1500
        },
        {
            "type": "stop",
            "name": "Station 2",
            "id": "3011223344556677",
            "zone": "240",
            "timingPoint": false,
            "pointVisitCount": 1,
```

```
            "latitude": 55.41433,
            "longitude": 13.23444,
            "datetimePlanned": {
                "zone": "local",
                "date": "2014-02-03",
                "time": "18:35:00"
            }
        },
        {
            "type": "link",
            "length": 2500
        },
        {
            "type": "stop",
            "name": "End station",
            "id": "4011223344556677",
            "zone": "240",
            "timingPoint": false,
            "pointVisitCount": 1,
            "latitude": 55.71433,
            "longitude": 13.21444,
            "datetimePlanned": {
                "zone": "local",
                "date": "2014-02-03",
                "time": "18:40:00"
            }
        }
    ]
}
```

## /vimi/pis/route/journey_point

Published by the PIS application.

Updated when an event on a journey occurs (e.g. arrival, departure…). The purpose with this topic is to inform about important events on a journey to subscribers.

Please note that the parameters within this topic is based on the information from /vimi/pis/route/journey and /vimi/system/identity/info, and in order to determine which parameters that will be available at bus and train one has to study each field that is referred to in the description column (e.g. "vehicleId"; see /vimi/system/identity/info).

| Key | Description |
|---|---|
| `datetime` | Datetime object. |
| `datetimePlanned` | Planned time for arrival/departure |
| `event` | Type of event:<br>`"departure"`: Departing event.<br>`"arrival"`: Arrival event.<br>`"passage"`: Passage event. |
| `vehicleId` | See /vimi/system/identity/info. |
| `trainSetId` | See /vimi/system/identity/info. |
| `trainNumber` | See /vimi/pis/route/journey. |
| `vehicleJourneyId` | See /vimi/pis/route/journey. |
| `currentStop` | Object holding information about the current stop. See description below. |
| `nextStop` | Object holding information about the next stop. See description below. |

### currentStop

| Key | Description |
|---|---|
| `routeIndex` | Index in the route list given in /vimi/pis/route/journey. |
| `id` | See /vimi/pis/route/journey. |
| `name` | See /vimi/pis/route/journey. |
| `zone` | See /vimi/pis/route/journey. |
| `latitude` | See /vimi/pis/route/journey. |
| `longitude` | See /vimi/pis/route/journey. |

### nextStop

| Key | Description |
|---|---|
| `routeIndex` | Index of the route list (given in /vimi/pis/route/journey) for which the route event was generated for. |
| `id` | See /vimi/pis/route/journey. |
| `name` | See /vimi/pis/route/journey. |
| `zone` | See /vimi/pis/route/journey. |
| `latitude` | See /vimi/pis/route/journey. |
| `longitude` | See /vimi/pis/route/journey. |
| `distance` | See /vimi/pis/route/journey. |

```
{
    "datetime": {
        "zone": "local",
        "date": "2014-02-03",
        "time": "18:31:46"
    },
    "datetimePlanned": {
        "zone": "local",
        "date": "2014-02-03",
        "time": "18:31:00"
    },
    "event": "arrival",
    "vehicleId": "0011223344556677",
    "trainSetId": "61063",
    "trainNumber": "1703",
    "vehicleJourneyId": "0011223344556677",
    "currentStop": {
        "routeIndex": 0,
        "id": "2011223344556677",
        "name": "Station 1",
        "zone": "350"
        "latitude": 55.71433,
        "longitude": 13.21444
    },
    "nextStop": {
        "routeIndex": 0,
        "id": "3011223344556677",
        "name": "Station 2",
        "zone": "350"
        "latitude": 55.71433,
        "longitude": 13.21444,
        "distance": 2500
    }
}
```

## /vimi/pis/route/progress

Published by the PIS application.

For subscribers who needs continuous update of the route progress. Please note that this topic is also based on information from /vimi/pis/route/journey and /vimi/system/identity/info, and one has to follow the referenced topic for each parameter in order to know if it will be available at bus or train respectively.

MQTT Retain: false.

| Key | Description |
|---|---|
| currentRouteIndex | Current index in the route list from /vimi/pis/route/journey. |
| datetime | Datetime object. |
| vehicleJourneyId | See/vimi/pis/route/journey. |
| distanceTraveled | The distance in meters to the stop that the vehicle was last standing at, or after arriving; the distance to the stop the vehicle is standing at. This value should thus be close to 0 when standing at a stop. Not available at train. |
| linkLength | Distance to travel until reaching next stop. The ratio distanceTraveled/linkLenght is 0 when we are at a stop and converge to 1 when we are approaching the next stop. Not available at train. |
| timetableDeviation | Calculated deviation from timetable, if available, in seconds |
| nextStop | Object that holds basic information of the next stop. |
| latitude | Latitude. |
| longitude | Longitude. |
| speed | Speed, km/h |

```
{
    "currentRouteIndex": 0,
    "datetime": {
        "zone": "local",
        "date": "2014-02-03",
        "time": "18:31:46"
    },
    "vehicleJourneyId": "0011223344556677",
    "distanceTraveled": 0,
    "linkLength": 1540,
    "timetableDeviation": -75,
    "nextStop": {
        "routeIndex": 0,
        "id": "2011223344556677",
        "name": "Station 2"
        "zone": "350",
        "timingPoint": false
    },
    "latitude": 55.71433,
    "longitude": 13.21444,
    "speed": 34.5
}
```

# APC

This section contains the topics published by the APC application.

## /vimi/apc/sensor/onboardcount

Published by the APC application.

Number of passengers currently onboard.

This topic shall be updated in real-time by the APC application whenever it has a new passenger count reading from its sensors. The purpose with this topic is to inform other applications connected to VIMI with the number of passengers onboard.

| Key | Description |
|---|---|
| `numPassengers` | Number of passengers currently onboard (from the APC system). |
| `timestamp` | Unix epoch timestamp when the value was calculated. |

```
{
    "numPassengers": 1,
    "timestamp": 1234
}
```

## /vimi/apc/command/resetonboardcount

The APC application is responsible to subscribe to this topic.

Used for sending command to the APC application that the onboard count shall be set to zero.

MQTT Retain: false.

| Key | Description |
|---|---|
| `action` | `"reset"`: Set the number of passengers onboard to zero. |

```
{
    "action": "reset"
}
```

## /vimi/apc/event

Published by the APC application.

Last generated apc report. This topic shall be published at the same time the report is generated, independent of what's reported to the PT authority. The purpose with this topic is to have the report available in real-time onboard at the vehicle.

### Example train APC report:

```
{
    "trainSetID": "00002001",
    "type": "APC",
    "trainNumber": 1234,
    "timestamp": "2012-04-19 08:46:36",
    "station": "Helsingborg",
    "onboard": "65",
    "messageId": "121",
    "doorActivities": [
        {
            "door": "01",
            "alighting": "5",
            "boarding": "15"
        },
        {
            "door": "03",
            "alighting": "2",
            "boarding": "7"
        },
        {
            "door": "07",
            "alighting": "1",
            "boarding": "0"
        }
    ],
    "xsum": "0a5c7401e9c4c0f26ae4da5917e062d9fb7b5598"
}
```

## Driver Identification

This section contains the topics published by the driver identification application.

### /vimi/driverid/user/event

Published by the driver identification application.

This topic shall be updated by the driver identification application whenever a user is signed on or off.

| Key | Description |
|---|---|
| type | "signon": Sign on to the system.<br>"signoff": Sign off from the system. |
| driver | Driver object. |
| id | Driver identifier |
| name | Driver display name |
| datetime | Datetime object. See /vimi/system/sensor/clock |
| source | Source of the signon or signoff event |

```
{
    "event": "signon",
    "driver": {
        "id": "0011223344556677",
        "name": "Anders Andersson"
    },
    "datetime": {
        "zone": "local",
        "date": "2014-02-03",
        "time": "18:31:46"
    },
    "source": "RFID Reader"
}
```

## /vimi/driverid/command/signoff

The driver identification application is responsible to subscribe to this topic.

Used for sending forced signoff commands to the driver identification application. This should be used in situations that has invalidated the current signed on user.

MQTT Retain: false.

| Key | Description |
|---|---|
| `action` | `"signoff"`: Sign off the current user. |

```
{
    "action": "signoff"
}
```

# VOIP

This section contains the topics published by the VOIP application.

## /vimi/voip/assignment/vehicle_journey

Published by the VOIP application.

Assignment event to inform that the vehicle has signed on or signed off to a journey.

| Key | Description |
|---|---|
| type | See /vimi/pis/assignment/vehicle_journey |
| datetime | See /vimi/pis/assignment/vehicle_journey |
| vehicleId | See /vimi/pis/assignment/vehicle_journey |
| vehicleJourneyId | See /vimi/pis/assignment/vehicle_journey |

```
{
    "type": "signon",
    "datetime": {
        "zone": "local",
        "date": "2014-02-03",
        "time": "18:31:46"
    },
    "vehicleId": "0011223344556677",
    "vehicleJourneyId": "0011223344556677"
}
```

## /vimi/voip/call/event

Published by the VOIP application.

This topic shall be updated by the VOIP application whenever an incoming, outgoing or missed call occurs.

| Key | Description |
|---|---|
| `type` | Type of event:<br>`incomingCall`<br>`outgoingCall`<br>`missedCall` |
| `datetime` | Datetime object. |

```
{
    "type": "incomingCall",
    "datetime": {
        "zone": "local",
        "date": "2014-02-03",
        "time": "18:31:46"
    }
}
```

## /vimi/voip/call/ongoing

Published by the VOIP application.

Whether or not a telephone call is ongoing.

This topic shall be updated by the VOIP application whenever a call is started or ended.

| Key | Description |
|---|---|
| `callOngoing` | True if a call is in progress, false otherwise |
| `datetime` | Datetime object. |

```
{
    "callOngoing": true,
    "datetime": {
        "zone": "local",
        "date": "2014-02-03",
        "time": "18:31:46"
    }
}
```

## /vimi/voip/message/statusmessage

Published by the VOIP application.

This topic shall be updated by the VOIP application whenever a status message is sent.

| Key | Description |
|---|---|
| `statusMessage` | Status message sent from the VoIP application |
| `datetime` | Datetime object. |

```
{
    "statusMessage": "9076012000000011",
    "datetime": {
        "zone": "local",
        "date": "2014-02-03",
        "time": "18:31:46"
    }
}
```

## /vimi/voip/message/textmessage

Published by the VOIP application.

This topic shall be updated by the VOIP application whenever an incoming or outgoing text occurs.

| Key | Description |
|---|---|
| `type` | Type of event:<br>`incomingText`<br>`outgoingText` |
| `priority` | Priority level:<br>`0 = Normal`<br>`1 = Urgent`<br>`2 = Critical` |
| `datetime` | Datetime object. |

```
{
    "type": "incomingText",
    "priority": 0,
    "datetime": {
        "zone": "local",
        "date": "2014-02-03",
        "time": "18:31:46"
    }
}
```

## Internet Onboard

This section contains the topics published by the Internet Onboard (IOB) gateways.

### /vimi/iob/gateway/<mac>/active_devices

This topic is published by the Internet Onboard gateways, typically every minute

| Subtopic | Description |
|---|---|
| `<mac>` | The MAC address of the unit. This should be specified with only numbers and lower case letters. Examples: `0f23e45d67c8` – correct `0F:23:E4:5D:67:C8` – incorrect |

| Key | Description |
|---|---|
| `numberConnections` | Number of connected devices |
| `datetime` | Datetime object. See /vimi/system/sensor/clock |
| `connectedDevices` | Array of connected devices |
| `macAddress` | MAC address for the connected device |
| `ipAddress` | IP address for the connected device |
| `deviceName` | Device name of connected device |

```
{
    "numberConnections": 2,
    "datetime": {
            "zone": "local",
            "date": "2014-02-03",
            "time": "18:31:46"
    },
    "connectedDevices": [
        {
            "macAddress": "0022446688"
            "ipAddress": "192.168.2.27"
            "deviceName": "android-a12b34c56d78e90f"
        },
        {
            "macAddress": "88AACCEE00"
            "ipAddress": "192.168.2.28"
            "deviceName": "Johns iPhone"
        }
    ]
}
```

## Seat Detection

This section contains the topics published by the Seat Detection application.

### /vimi/seatdetection/sensor/data

**This topic is preliminary and may be subject to change.**

This topic is published by the Seat Detection application, typically every 5 minutes. Seat detectors can be divided in different sections of the vehicle, where a section could refer to for instance decks on a double-decker bus or wagons within a train set.

| Key | Description |
|---|---|
| `datetime` | Datetime object. See /vimi/system/sensor/clock |
| `sensors` | Array of sensors |
| `id` | ID of the sensor |
| `signal` | Signal strength in percent |
| `battery` | Battery level in percent |
| `occupied` | Whether or not the seat is occupied |
| `section` | Which section the sensor is placed in |

```
{
    "datetime": {
            "zone": "local",
            "date": "2014-02-03",
            "time": "18:31:46"
    },
    "sensors": [
        {
            "id": "Sensor 1"
            "signal": 80
            "battery": 75,
            "occupied": true,
            "section": 1
        },
        {
            "id": "Sensor 2"
            "signal": 89
            "battery": 75,
            "occupied": true,
            "section": 1
        },
        {
            "id": "Sensor 3"
            "signal": 80
            "battery": 75,
            "occupied": true,
            "section": 1
        },
        {
            "id": "Sensor 4"
            "signal": 80
            "battery": 75,
            "occupied": false,
            "section": 1
        },
        {
            "id": "Sensor 5"
            "signal": 80
            "battery": 45,
            "occupied": true,
            "section": 2
        },
        {
            "id": "Sensor 6"
            "signal": 82
            "battery": 75,
            "occupied": false,
            "section": 2
        }
    ]
}
```

## /vimi/seatdetection/sensor/occupancy

**This topic is preliminary and may be subject to change.**

This topic is published by the Seat Detection application, typically every 10 seconds. Seat detectors can be divided in different sections of the vehicle, where a section could refer to for instance decks on a double-decker bus or wagons within a train set.

| Key | Description |
|---|---|
| sectionOccupancy | List of occupancy status per section |
| numberSensors | Number of connected devices |
| occupiedSeats | Number of occupied seats |
| section | Which section the occupancy is related to |
| datetime | Datetime object. See /vimi/system/sensor/clock |

```
{
    "datetime": {
            "zone": "local",
            "date": "2014-02-03",
            "time": "18:31:46"
    },
    "sectionOccupancy": [
        {
            "section": 1,
            "numberSensors": 4,
            "occupiedSeats": 3,
        },
        {
            "section": 2,
            "numberSensors": 2,
            "occupiedSeats": 0,
        }
    ]
}
```

## Ticket Validation

This section contains the topics posted by the ticket validation application.

### /vimi/ticketing/identity/info/

This topic is posted by the ticketing validation system.

Holds information about the ticketing system

| Key | Description |
|---|---|
| controllerId | Serial number of the controller device |
| validatorIds | List of validator id objects |
| validator | Number of the validator |
| id | Serial number of the validator |

```
{
  "controllerId": "1234",
  "validatorIds": [
    {
      "validator": "1",
      "id": "12345678"
    },
    {
      "validator": "2",
      "id": "34567890"
    }
  ]
}
```

## /vimi/ticketing/validator/ticket_validation/

This topic is posted by the ticketing validation system upon each ticket validation.

| Key | Description |
|---|---|
| datetime | Datetime object. See /vimi/system/sensor/clock |
| validator | ID of the validator |
| type | Type of validation – Online or Offline |
| status | Status code for the validation |
| ticketValid | Validation result – true or false |
| ticketType | Type of ticket |
| message | Only for invalid tickets. Message containing information of why the validation failed. Should be readable for e.g. bus drivers |

```
{
    "datetime": {
        "zone": "local",
        "date": "2016-04-06",
        "time": "03:00:21"
    },
    "validator": "Door 2, Validator 1",
    "type": "Online",
    "status": 1,
    "ticketValid": false,
    "ticketType": "1-Day",
    "message": "Ticket expired at 2016-04-06, 00:00"
}
```

## /vimi/ticketing/command/power

The ticketing system is responsible to subscribe to this topic.

Used for sending power on or off commands to the ticket validation system.

MQTT Retain: false.

| Key | Description |
|---|---|
| `action` | `"poweron":` Tells the ticketing system to power on.<br>`"poweroff":` Tells the ticketing system to power off. |

```
{
    "action": "poweron"
}
```

# Diagnostics

This section contains the topics posted by the diagnostics application.

## /vimi/diagnostics/performance

**This topic is preliminary and may be subject to change.**

This topic is posted every 10 seconds

| Key | Description |
|---|---|
| vehicleSpeed | Vehicle speed (m/h) |
| enginePower | Current engine power in (W) |
| distanceSinceReboot | Distance traveled since reboot (m) |
| powerConsumptionSinceReboot | Power consumption since last reboot (Wh) |

```
{
    "vehicleSpeed": 67000,
    "enginePower": 87291,
    "distanceSinceReboot": 732231,
    "energyConsumptionSinceReboot": 3320987
}
```

## /vimi/diagnostics/flags

**This topic is preliminary and may be subject to change.**

This topic is posted on any flag change

| Key | Description |
|---|---|
| ebsRedWarning | EBS red warning light, true/false |

```
{
    "ebsRedWarning": true
}
```

## Report Gateway

This section contains the topics related to sending and receiving reports through the Report Gateway application.

### /vimi/report-gateway/send/<app>

Send report to PT authority (central system).

The VIMI gateway subscribes to /vimi/report-gateway/send/% and each application that wants to send a report (report-sender) publish information on "their own" topic (<app>). The report-sender must then subscribe to /vimi/report-gateway/res/<app> in order to get the send result.

| Subtopic | Description |
|---|---|
| `<app>` | Application that sends the message. |

Example:

```
/vimi/report-gateway/send/apc
```

| Key | Description |
|---|---|
| `seq` | A unique sequence number for each report**. Note that a resend of a report shall use the same seq number as used in the previous attempt. A new report shall have a new unique seq number.** Preferable, the unix epoch can be used (e.g. when the report was generated). |
| `message` | The actual message to be sent (report) |

Topics

The following topics are currently defined with the corresponding responsible applications/publishers:

| Topic | Application |
|---|---|
| `/vimi/report-gateway/send/apc` | APC |
| `/vimi/report-gateway/send/statmon` | State monitor |
| `/vimi/report-gateway/send/occupancy` | Seat detection |

### APC

The message for the APC report and the weight report and shall be formatted as specified in

- **For train:** Train APC report on page 66.
- **For bus:** Bus APC report on page 70.

Each report shall have a unique seq number.

APC should send only one APC data report per planned stop on a journey according to the following:

When an arrival event (/vimi/pis/route/journey_point with event=arrival) occurs, the APC starts a timer **t**. The APC stores the count of boarding/alighting that occurs between the arrival and the time **t** as an intermediate result, but continues to count even after the time **t** has elapsed.

After a departure event (/vimi/pis/route/journey_point with event=departure), one of the following cases occur:

- If the vehicleJourneyId is the same for the arrival and departure events, send one APC data report containing the vehicleJourneyId and the total count for this stop.
- If the vehicleJourneyIds are not the same for the arrival and departure events, send two APC data reports for this stop. One containing the arrival vehicleJourneyId and the intermediate result, and one containing the departure vehicleJourneyId and the remaining count (the total count for this stop minus the intermediate result).
- If no arrival event has been observed for this stop (would typically occur at the first stop of a journey that has no preceding journey), send one APC data report containing the vehicleJourneyId and the total count for this stop.

Boarding/alightning counters then turns to zero until next counting-event. If passengers are boarding or alighting between two planned stops, those passengers shall be added to the passenger count for the next planned stop.

If the departure should occur before the time **t** has elapsed, the count up until that point shall be regarded as the intermediate result.

If no departure is observed for a time **X**, send one APC data report containing the vehicleJourneyId for the arrival event and the total count for this stop. The counters shall then be reset.

When a passage event (/vimi/pis/route/journey_point with event=passage) occurs, a report should be sent. This would normally contain zero boarding and alighting passengers, but in the case that the bus has stopped between planned stops and passengers has boarded/alighted, that count should be included in the passage event APC report.

The times **t** and **X** should be easily configurable.

### APC Report validation

The mandatory parameters in the APC report are validated by the report-gateway.

**Figure 1**: Arrival event and Departure event on same jouney id

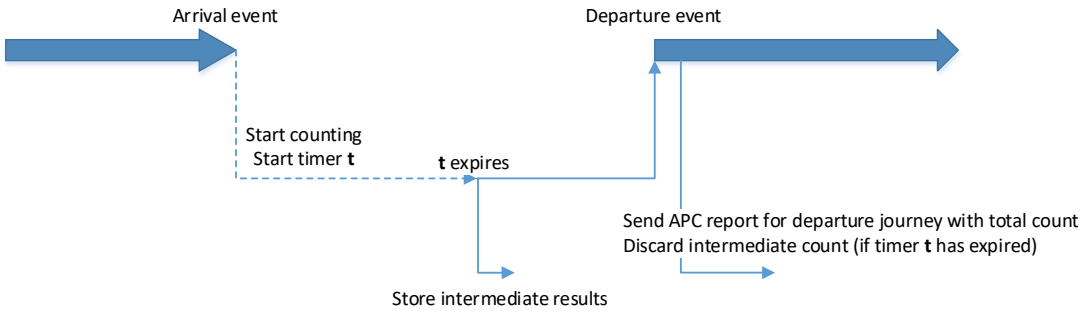



**Figure 2**: Arrival event and Departure event on different jouney id

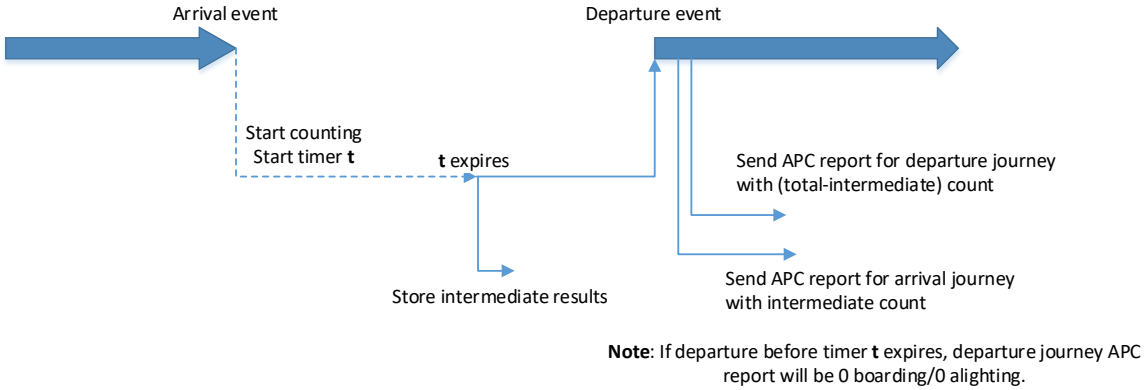



**Figure 3**: Last trip of the day (or long pause in block)

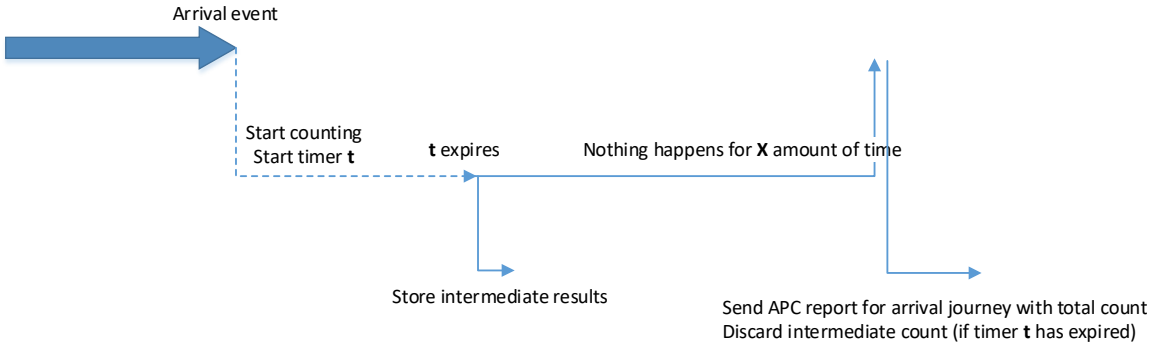
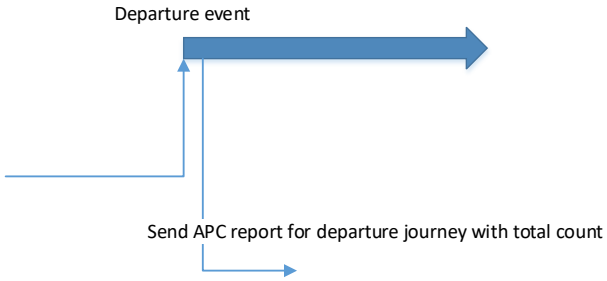



**Figure 4**: First trip of the day (no arrival event)



Example APC Train report


```
{
    "seq": 1432209505,
    "message": {
        "trainSetID": "00002001",
        "type": "APC",
        "trainNumber": 1234,
        "timestamp": "2012-04-19 08:46:36",
        "station": "Helsingborg",
        "onboard": "65",
        "messageId": "121",
        "doorActivities": [
            {
                "door": "01",
                "alighting": "5",
                "boarding": "15"
            },
            {
                "door": "03",
                "alighting": "2",
                "boarding": "7"
            },
            {
                "door": "07",
                "alighting": "1",
                "boarding": "0"
            }
        ],
        "xsum": "0a5c7401e9c4c0f26ae4da5917e062d9fb7b5598"
    }
}
```

Example APC Bus report


```
{
    "seq": 1432209505,
    "message": {
        "type": "APC",
        "vehicleRef": "9031012004507123",
        "journeyRef": "9015012053400111",
        "timestamp": "2013-08-17T09:30:47+02:00",
        "pointRef": "9025012009307001",
        "onboardCount": "37",
        "messageId": "121",
        "doorActivities": [
            {
                "doorRef": "01",
                "alightingCount": "1",
                "boardingCount": "9"
            },
            {
                "doorRef": "02",
                "alightingCount": "8",
                "boardingCount": "2"
            },
            {
                "doorRef": "03",
                "alightingCount": "3"
            }
        ]
    }
}
```

## Status

The system shall send status messages containing status messages for all connected units. The status messages shall be formatted according to "Bus status report" on page 72.

The status message should be sent immediately if an error of severity 2 or 3 has occurred, or if a unit does not send a scheduled state message, otherwise the report should be sent with some recurrence. The message shall contain a list of the connected units, and each unit shall contain a list of status updates that are new since the last status message was sent.

Each report shall have a unique seq number.

Example bus status report


```
{
    "seq":1432209505,
    "message":{
        "type":"Status",
        "vehicleRef":"9031012004507123",
        "timestamp":"2013-08-17T09:30:47+02:00",
        "reportInterval": 3600,
        "messageId":372,
        "units": [
            {
                "unit":"/unit/Dilax/apc/001122334455",
                "unitType": "Dilax",
                "version":"1.20.0",
                "publishInterval":300,
                "epochNow":1431502890,
                "uptimeStart":125,
                "uptimeNow":4401,
                "currentStatus":"running",
                "statusErrors": [
                    {
                        "status" : "dead",
                        "countSeen":1,
                        "countGone":0,
                        "epochFirstSeen":1431502890,
                        "epochLastGone":1431502890
                    }
                ],
                "userErrors":[
                    {
                        "code":15,
                        "severity":2,
                        "message":"No access to internet",
                        "countSeen":12,
                        "countGone":7,
                        "epochFirstSeen":1431502890,
                        "epochLastGone":1431502890,
                        "active":true
                    }
                ]
            },
            {
               "unit":"/unit/faltcom/driverID/001122334455",
                "version":"1.20.0",
                "publishInterval":300,
                "epochNow":1431502890,
                "uptimeStart":125,
```

```
                "uptimeNow":4401,
                "currentStatus":"dead"
            }
        ]
    }
}
```

## Occupancy

The Seat detection app shall publish occupancy reports containing the occupancy status for each seat fitted with a seat detection sensor. The reports shall be formatted according to "Bus occupancy report" on page 75. The reports shall be sent when the vehicle is in traffic (i.e. from a message /vimi/pis/assignment/block with type "signon" is posted until a message /vimi/pis/assignment/block with type "signoff" is posted). The report shall be sent in intervals of one minute.

Each report shall have a unique seq number.

### Example bus occupancy report


```
{
    "seq": 1432209505,
    "message": {
        "type": "Occupancy",
        "vehicleRef": "9031012004507123",
        "timestamp": "2016-04-06T09:30:47+02:00",
        "messageId": "372,
        "occupancies": [
            {
                "id": "Seat 123",
                "occupied": true,
                "section": 1
            },
            {
                "id": "Seat 124",
                "occupied": true,
                "section": 1
            },
            {
                "id": "Seat 125",
                "occupied": false,
                "section": 1
            },
            {
                "id": "Seat 223",
                "occupied": true,
                "section": 2
            },
            {
                "id": "Seat 224",
                "occupied": true,
                "section": 2
            },
            {
                "id": "Seat 225",
                "occupied": false,
                "section": 2
            },
        ]
    }
}
```

## /vimi/report-gateway/res/<app>

Published by the VIMI report gateway application.

Contains the send result of the report (for the report-sender to get feedback if the transmission succeeded or failed). Please see /vimi/report-gateway/send/<app> for more information.

| Subtopic | Description |
|---|---|
| <app> | See /vimi/report-gateway/send/<app> |

Example:

```
/vimi/report-gateway/res/apc
```

| Key | Description |
|---|---|
| seq | Matching sequence number as last received in /vimi/report-gateway/send/<app> |
| result | Result string:<br>"sent": Report sent ok.<br>"busy": Report not accepted (gateway busy). Try send again later.<br>"rejected": Report cannot be sent since it's invalid. Sender shall not re-send.<br>"failed": Report could not be sent due to communication problems, try send again later. |
| errormsg | Optional error message. |

```
{
    "seq": 1432209505,
    "result:": "rejected"
    "errormsg:": "Invalid syntax"
}
```

## Topics

The following topics (and responsible subscribers) are currently defined:

| Topic | Subscriber application |
|---|---|
| `/vimi/report-gateway/res/apc` | APC |
| `/vimi/report-gateway/res/status` | System |
| `/vimi/report-gateway/res/occupancy` | Seat detection |

## Scale

This section contains the topics published by the scale application.

### /vimi/scale/event

Only at train.

Published by the scale application.

Last generated scale report (train-set weight report). Same format as in specified in "APC Train Reporting Interface 1.0". This topic shall be published at the same time the report is generated, independent of what's reported to the PT authority. The purpose with this topic is to have the information available in real-time onboard at the vehicle.

# Reports

Reports shall be sent via the VIMI report-gateway, see section Report Gateway.

The sending application for each report must ensure that the report gateway gives a positive response.

## Train APC report

The following attributes and elements should be included in the APC report:

| Name | Optional? | Description |
|---|---|---|
| `trainSetID` | No | Number identifying the physical train-set. String. Max 16 characters. |
| `type` | No | Fixed value: APC. |
| `trainNumber` | No | Train number. Numeric. Max 5 digits. |
| `timestamp` | No | Time of departure or if not available; timestamp of when data was last received from train-computer. Format 'YYYY-MM-DD HH:MM:SS'. |
| `station` | No | Name of station the train is currently departing from. |
| `onboard` | No | Number of passengers onboard after departing from the station. |
| `messageId` | No | A sequential number incremented by one for each sent APC report. |
| `doorActivities` | No | An array of doorActivity objects. Each doorActivity describes APC information for an individual door of the train. The content of a doorActivity element is described in a separate table below. |
| `xsum` | No | Lower-case hex-string built using SHA-1 on concatenated message content[2] and shared secret character-sequence. |

Content of a *doorActivity* element:

| Name | Optional? | Description |
|---|---|---|
| `door` | No | Alfanumeric reference identifying the door locally within the train-set. String. Max 16 characters. |
| `alighting` | Yes | Number of passengers that left the train through this door at this station. This attribute can be omitted if no passengers alighted through this door. |
| `boarding` | Yes | Number of passengers that entered the train through this door at this station. This attribute can be omitted if no passengers boarded through this door. |

[2] Please note that only parameters defined in this table shall be included in the "content". Also, see example below how to calculate the xsum.

Example - Train APC report


```
{
    "trainSetID": "00002001",
    "type": "APC",
    "trainNumber": 1234,
    "timestamp": "2012-04-19 08:46:36",
    "station": "Helsingborg",
    "onboard": "65",
    "messageId": "121",
    "doorActivities": [
        {
            "door": "01",
            "alighting": "5",
            "boarding": "15"
        },
        {
            "door": "03",
            "alighting": "2",
            "boarding": "7"
        },
        {
            "door": "07",
            "alighting": "1",
            "boarding": "0"
        }
    ],
    "xsum": "0a5c7401e9c4c0f26ae4da5917e062d9fb7b5598"
}
```

Example - xsum

If the shared secret character-sequence is "SECRETKEY" and the message-part before xsum is:

```
"trainSetID": "00002001",
"type": "APC",
"trainNumber": 1234,
"timestamp": "2012-04-19 08:46:36",
"station": "Helsingborg",
"onboard": "65",
"messageId": "121",
"doorActivities": [
    { "door": "01", "alighting": "5", "boarding": "15" },
    { "door": "03", "alighting": "2", "boarding": "7" },
    { "door": "07", "alighting": "1", "boarding": "0" }
],
```

Then build hex-string from:

"00002001APC12342012-04-19 08:46:36Helsingborg65121015150327071OSECRETKEY"

## Train Weight report

The following attributes and elements should be included in the WEIGHT report:

| Name | Optional? | Description |
|---|---|---|
| trainSetID | No | Number identifying the physical train-set. String. Max 16 characters. |
| type | No | Fixed value WEIGHT. |
| trainNumber | No | Train number. Numeric. Max 5 digits. |
| timestamp | No | Time of departure or if not available; time stamp of when data was last received from train-computer. Format 'YYYY-MM-DD HH:MM:SS'. |
| station | No | Name of station the train departed from before measuring the weight. |
| nextStation | Yes | Name of station the train arrives at after measuring the weight. |
| weight | No | Total weight in kilograms of the train-set after departing from the station. Numeric. Max 7 digits. |
| xsum | No | Lower-case hex-string built using SHA-1 on concatenated message content and shared secret character-sequence. |

Example - Train weight report

```
{
    "trainSetID": "00002001",
    "type": "WEIGHT",
    "trainNumber": 1234,
    "timestamp": "2012-04-19 08:46:36",
    "station": "Helsingborg",
    "nextStation": "Landskrona",
    "weight": "159035",
    "xsum": "0b3c7561e9c4c0f26ae4da5917e062e2fb7b5421"
}
```

## Bus APC report

The following attributes and elements should be included in the Bus APC report:

| Name | Optional? | Description |
|---|---|---|
| `type` | No | Fixed value: APC. |
| `vehicleRef` | No | The vehicle identifier (VehicleGid). 16 digits. |
| `journeyRef` | No | The current journey identifier (VehicleJourneyGid). 16 digits. |
| `timestamp` | No | Current time. |
| `pointRef` | No | Identifier (JourneyPatternPointGid) of the stop that the vehicle departed from. 16 digits. |
| `pointVisitCount` | Yes | Normally each stop is called only once during a vehicle journey, and then this attribute can be omitted. However, some vehicle journeys have such journey patterns that the same stop point is called a second time. In such instances this attribute should be included and set to the value 2 for the second visit to the same stop. |
| `onboardCount` | No | Number of passengers onboard after departing from the station. |
| `messageId` | No | A sequential number incremented by one for each sent APC report. |
| `doorActivities` | No | An array of doorActivity objects. Each doorActivity describes APC information for an individual door of the vehicle. The content of a doorActivity element is described in a separate table below. |

Content of a *doorActivity* element:

| Name | Optional? | Description |
|---|---|---|
| `doorRef` | No | Alfanumeric reference identifying the door locally within the train-set. String. Max 16 characters. |
| `alightingCount` | Yes | Number of passengers that left the vehicle through this door at this stop. This attribute can be omitted if no passengers alighted through this door. |
| `boardingCount` | Yes | Number of passengers that entered the vehicle through this door at this stop. This attribute can be omitted if no passengers boarded through this door. |

Example- Bus APC report


```
{
    "type": "APC",
    "vehicleRef": "9031012004507123",
    "journeyRef": "9015012053400111",
    "timestamp": "2013-08-17T09:30:47+02:00",
    "pointRef": "9025012009307001",
    "onboardCount": "37",
    "messageId": "121",
    "doorActivities": [
        {
            "doorRef": "01",
            "alightingCount": "1",
            "boardingCount": "9"
        },
        {
            "doorRef": "02",
            "alightingCount": "8",
            "boardingCount": "2"
        },
        {
            "doorRef": "03",
            "alightingCount": "3"
        }
    ]
}
```

## Bus status report

The following attributes and elements should be included in the system status report:

| Name | Optional? | Description |
|---|---|---|
| type | No | Fixed value: Status. |
| vehicleRef | No | The vehicle identifier (VehicleGid). 16 digits. |
| timestamp | No | Current time. |
| reportInterval | No | Interval between reports (seconds). |
| messageId | No | A sequential number incremented by one for each sent status report. |
| units | No | An array of unit objects. Each unite describes last known status for connected units. The content of a unit element is described in a separate table below. |

Content of a *unit* element:

| Name | Optional? | Description |
|---|---|---|
| unit | No | Name of the unit |
| unitType | No | Input from config file, currently default value is provider, can be different from provider, used as a "tag" to be able to do select in the database. |
| version | No | See /unit/<provider>/<name/label>/<mac>/version |
| status | No | Current status |
| epochNow | No | Current unix epoch timestamp of the unit. |
| uptimeStart | No | Current system uptime of the unit (seconds). |
| uptimeNow | No | Uptime when the unit application started (seconds). |
| currentStatus | No | Current status of the unit. |
| publishInterval | No | How often the unit will publish to this topic (seconds). |
| statusErrors | Yes | An array of statusError objects. The contents of a statusError object is described in a separate table below. |
| userErrors | Yes | An array of userError objects. The contents of a userError object is described in a separate table below. |

Content of a *statusError* element:

| Name | Optional? | Description |
|---|---|---|
| status | No | Error status of the unit. Can be *stopped*, *dead*, *nopulse*, or *ignitionoff*. |
| countSeen | No | Number of times the status error has occurred. |
| countGone | No | Number of times the status error has been cleared. |
| epochFirstSeen | No | Epoch timestamp of first occurrence of this status error. |
| epochLastGone | No | Epoch timestamp of last occurrence of clearing of this status error. |

Content of a *userError* element:

| Name | Optional? | Description |
|---|---|---|
| `code` | No | See /unit/<provider>/<name/label>/<mac>/state |
| `severity` | No | See /unit/<provider>/<name/label>/<mac>/state |
| `message` | No | See /unit/<provider>/<name/label>/<mac>/state |
| `countSeen` | No | Number of times the user error has occurred. |
| `countGone` | No | Number of times the user error has been cleared. |
| `epochFirstSeen` | No | Epoch timestamp of first occurrence of this user error. |
| `epochLastGone` | No | Epoch timestamp of last occurrence of clearing of this user error. |
| `active` | No | *true* if the error is active, *false* otherwise. |

Example bus status report


```
{
    "type":"Status",
    "vehicleRef":"9031012004507123",
    "timestamp":"2013-08-17T09:30:47+02:00",
    "reportInterval": 3600,
    "messageId":372,
    "units": [
        {
            "unit":"/unit/Dilax/apc/001122334455",
            "unitType": "Dilax",
            "version":"1.20.0",
            "publishInterval":300,
            "epochNow":1431502890,
            "uptimeStart":125,
            "uptimeNow":4401,
            "currentStatus":"running",
            "statusErrors": [
                {
                    "status" : "dead",
                    "countSeen":1,
                    "countGone":0,
                    "epochFirstSeen":1431502890,
                    "epochLastGone":1431502890
                }
            ],
            "userErrors":[
                {
                    "code":15,
                    "severity":2,
                    "message":"No access to internet",
                    "countSeen":12,
                    "countGone":7,
                    "epochFirstSeen":1431502890,
                    "epochLastGone":1431502890,
                    "active":true
                }
            ]
        },
        {
            "unit":"/unit/faltcom/driverID/001122334455",
            "version":"1.20.0",
            "publishInterval":300,
            "epochNow":1431502890,
            "uptimeStart":125,
            "uptimeNow":4401,
            "currentStatus":"dead"
        }
    ]
}
```

## Bus occupancy report

**This topic is preliminary and may be subject to change.**

The following attributes and elements should be included in the occupancy report:

| Name | Optional? | Description |
|---|---|---|
| `type` | No | Fixed value: Occupancy. |
| `vehicleRef` | No | The vehicle identifier (VehicleGid). 16 digits. |
| `timestamp` | No | Current time. |
| `messageId` | No | A sequential number incremented by one for each sent status report. |
| `occupancies` | No | An array of occupancy objects. Each occupancy describes last known occupancy status for each seat detection sensor. The content of a statusMessage element is described in a separate table below. |

Contents of an *occupancy* element:

| Name | Optional? | Description |
|---|---|---|
| `id` | No | ID of the sensor |
| `occupied` | No | Whether or not the seat is occupied |
| `section` | No | Which section the sensor is placed in. |

Example bus occupancy report


```
{
    "type": "Occupancy",
    "vehicleRef": "9031012004507123",
    "timestamp": "2016-04-06T09:30:47+02:00",
    "messageId": "372,
    "occupancies": [
        {
            "id": "Seat 123",
            "occupied": true,
            "section": 1
        },
        {
            "id": "Seat 124",
            "occupied": true,
            "section": 1
        },
        {
            "id": "Seat 125",
            "occupied": false,
            "section": 1
        },
        {
            "id": "Seat 126",
            "occupied": false,
            "section": 1
        },
        {
            "id": "Seat 223",
            "occupied": true,
            "section": 2
        },
        {
            "id": "Seat 224",
            "occupied": true,
            "section": 2
        },
        {
            "id": "Seat 225",
            "occupied": false,
            "section": 2
        },
        {
            "id": "Seat 226",
            "occupied": true,
            "section": 2
        }
    ]
}
```

# Partners


Blekingetrafiken


CONSAT
Telematics


DILAX


Drivec


FÄLTCOM
SHAPING THE INTERNET OF THINGS


Hogia


INFODEV
ELECTRONIC DESIGNERS
INTERNATIONAL


mobitec
A WORLD OF SIGNS


multiQ
LEADING DIGITAL SIGNAGE


Nobina


RIDANGO


Skånetrafiken


swarco


Telia


transdev


TRINORTH
SOLUTIONS