TransitCloud
Vehicle Interface Specification

Revision:	1.8
Date:	2016-03-22

TransitCloud
Vehicle Interface Specification

This document is part of the definition and design of the Hogia TransitCloud system. The design of the Hogia TransitCloud system is the property of Hogia Public Transport Systems.

Copyright

Revision History

Revision	Date	Update	Updated by
1.0	2014-01-17	Replaces document *Position Reports – Formats and encoding*.	Stefan Fjällemark
1.1	2014-05-15	Update with additional clarifications.	Stefan Fjällemark
1.2	2015-01-07	Update of usage of *Signals*, see 5.3.	Stefan Fjällemark
1.3	2015-05-29	Edited and added content. Change of name from *TransitCloud Position Message Formats and Encoding*.	Stefan Fjällemark
1.4	2015-10-23	Added clarification of how to provide Task Id in *Hogia Extended Position Message*, see 4.3.9. Changed meaning of *Distance* in *Hogia Extended Position Message*, see 5.4.	Stefan Fjällemark
1.5	2015-11-26	Adjustments of 4.3.10 and 5.4.	Stefan Fjällemark
1.6	2015-12-17	Added test packet, see 7.1.	Stefan Fjällemark
1.7	2016-01-27	Added importance of correct time, see 4.3.1. Added example of definition of *in service*, see 5.3.1.	Stefan Fjällemark
1.8	2016-03-22	AccountRef not mandatory, see chapter 6. Changes to Quality, see 5.2.	Stefan Fjällemark

Table of content

1 Introduction

This guide describes the requirements and the data formats for providing Hogia TransitCloud® with real-time data. This document replaces the document *Position Reports – Formats and encoding*.

Hogia TransitCloud® is a cloud based real-time vehicle monitoring service, capable of tracking vehicles in real-time and monitor progress compared to a planned route.

2 General Requirements

Please, also find additional details in chapter 3 Technical requirements.

2.1.1 Account

Each user of TransitCloud shall have an **account id**, provided by Hogia. In TransitCloud, data for each account are kept isolated from each other.

2.1.2 Operators

Within each account, it is possible to group vehicles into operators. The operator id of a vehicle is used for access control to certain functions in TransitCloud.

2.1.3 Common Identities

Within a TransitCloud account, it is important that all participants agrees to use a common set of identities of **vehicles**, **tasks** and **waypoints**, so that data can be understood by clients of TransitCloud. Hogia recommends that the identities are investigated and concluded before any data is sent in to TransitCloud.

2.1.4 Position Reports

The basic content of a position report is **identity**, **position**, **direction**, **speed** of the vehicle. This data should be sent each second, preferably 7/24/365 and directly from a device in the vehicle to TransitCloud and not passing any intermediate software/servers.

In many cases, the **current task** of the vehicle should also be included in the position report. In public transport this is usually the current service/trip number or the id of the current vehicle schedule.

In addition to the basic content some additional data can be sent.

NOTE:	A vehicle device shall not try to sanitize GPS-data before sending a position report. Data should be sent using the values received from the GPS. TransitCloud has functions that deals with the most common errors in GPS-data.

The format of position reports is described in detail from chapter 5 and onwards.

2.2 Why report so often?

One of the big advantages with TransitCloud is to enable smart real-time monitoring utilizing existing or cheap new equipment, eliminating any need for data or computing locally in the vehicle. All data processing of the GPS data and data from other sources (e.g. BUS FMS) are made centrally in the TransitCloud. Therefore, any captured data in the vehicle shall be sent in a raw simple format and as often as possible.

A GPS unit delivers a new position each second, so therefore it also should be sent each second to TransitCloud. Data should also be sent when the vehicle is out of service or parked, as long as the main power is on.

Using *Hogia Standard Position Message* (UDP) formats, the data volumes for sending messages each second, 24 hours a day is about 150 MB per month Sending task data each 30th second, using *Hogia Extended Position Message*, adds approximately another 50 MB. If data is sent through mobile internet, the data volume is well within most fixed data plans for mobile subscriptions.

2.2.1 What if position reports each second is impossible?

If, for some reason, position reports each second is impossible to implement, individual vehicles can be configured to use an alternative and coarser tracking algorithm, which is adapted for low frequency position reports and only detect passing or missing waypoints, but no detailed information such as arrival and departure times.

Note that TransitCloud is designed for position each second. Using the low frequency position tracking algorithm should be regarded as a last resort for real-time monitoring of vehicles when no alternatives are available.

3 Technical requirements

This chapter describes the requirement for the on-board hardware and installation.

3.1 On-board Systems in general

It is recommended that the design of the on-board equipment make it possible to send positions 24/7/365 while minimising the power consumption when the vehicles main power is off. Therefore, it is recommended that the on-board GPS- and communication units used for sending position messages have separate power not affected by the vehicles main power switch. Other more power consuming parts of the on-board equipment can be powered off when the vehicles main power is off.

Some suppliers offer a *vehicle communication gateway* functionality that can handle all communication and can monitor and control power of other on-board equipment. Implementing the RPS positioning message function in such a gateway would be a suitable solution.

3.2 GPS Devices

The functionality and accuracy of GPS devices has been improved greatly in the recent years.

3.2.1 GPS Device Capabilities

Use a modern GPS device. A modern type of GPS-receiver is generally better to deliver high quality position data. Development in this field continues to improve the quality of positioning.

If you have routes where loss of the GPS-signal is frequent or longer that acceptable for the application purpose, e.g. tunnels, use a GPS device with dead-reckoning capabilities.

Older GPS devices, i.e. GPS devices older that two years, often found in for example ticketing systems, does not perform well enough. For providing GPS-data to TransitCloud, it is better to add an additional modern GPS device than re-using an old existing GPS. Today, GPS devices are quite affordable.

3.2.2 Receiving Satellite Signals

Use an antenna mounted on the roof with a clear view of the sky with no obstructions from about five degrees elevation and up. This usually requires an externally roof mounted antenna.

An antenna used inside a vehicle will not work. An example is equipment with integrated GPS antennas, such as tablets. These devices will not be optimal for receiving the signals from the satellites, with degraded quality as a consequence.

3.2.3 Zero-speed detection

At very low speed, some GPS devices will have the positions dragging in a way that the direction is randomly rotating. Some GPS devices will under these circumstances not correctly detect the vehicle stopping and starting. When the vehicle in reality is standing still, the position is dragging causing a speed to be detected. TransitCloud handles this by having a configurable zero speed detection level. When speed drops below this level, the vehicle is regarded as standing still and its direction is locked.

3.3 Communication

3.3.1 Mobile Network

Position messages is expected to be sent over a 2G/3G/4G mobile network or similar mobile network. Experiences shows that the following problems can occur:

- Switching between 2G/3G/4G can cause interruptions in the communication.
- Roaming can sometimes take unacceptable long time.
- The communication unit simply hangs for some reason.

The user must ensure that the on-board equipment have the necessary functionality to minimise interruptions and quickly re-establish communications. This will require that the on-board communication unit can be monitored and reset if necessary.

For optimal coverage, it is important to use an externally roof mounted antenna.

3.3.2 Protocol

Position messages must be sent directly to TransitCloud sent by UDP and without any relaying via any other servers. Relaying causes unnecessary transmission delays.

If the on-board equipment uses TCP for other purposes, position messages must be sent out of bounds using UDP. TCP must not be used for position messages.

Although UDP has no delivery guarantee, using UDP has advantages over TCP:

- It is session-less which means that it can be sent without waiting for response from the destination.
- There are no timeouts that can cause the transmission to hang.
- If a packet is lost on its way to its destination, a new sent after one second.
- It is suitable for one-way communication.
- No acknowledges are sent by TransitCloud back to the vehicle.

3.3.3 Destination of UDP–packets

The UDP packet containing the position messages must be sent to an endpoint (IP-address and port number). URL-named endpoints depend on DNS lookup must not be used.

The destination endpoint will occasionally change. It is recommended that the on-board equipment's configuration automatically can be updated from a central configuration. The configuration shall be easily maintainable by the user.

3.3.4 When to send position messages

Positions should be sent each second 24/7/365 or at least when main power in a vehicle is on. It is strongly recommended that position messages are sent each second also when a vehicle is parked and also the main power is off. To indicate the state of the main power, each position message must contain the *Power On signal* (see 5.3.4).

If TransitCloud is not receiving position messages, this will be detected as a possible error in the vehicle equipment. If TransitCloud does not receive position messages with the expected time interval it will be recorded as a timeout.

3.4 Integration with other on-board equipment

It is strongly recommended that the vehicle equipment can include the following data in position messages:

- Vehicle id (see 4.3.7).
- Main power (see 5.3.4).
- In service (see 5.3.1).

It is recommended that the vehicle equipment can include the following data in position messages:

- Passenger requested stop (see 0).
- Door lock is released for passenger exchange (see 5.3.3).

The data can usually be retrieved from BUS-FMS bus or read as I/O.

The current value of the data items above should be read each second and included in each position message.

4 Guidelines for messages

4.1 Which Message Types to Use?

If you implement from scratch, it is recommended using one or both of the Hogia Messages below.

- **Hogia Standard Position Message,** described in chapter 5, is the choice if you are sending position, direction and speed using device id to identity sending vehicle, use. In this message, the current task of the vehicle cannot be sent.
- **Hogia Extended Position Message**, described in chapter 6, is the choice if you are sending data about which current journey the vehicle is running, in combination with position, direction and speed using either device id or vehicle id to identity the vehicle.

If you already have support for one of the protocols below, consider to request Hogia to add support for the one you have implemented. Please note, that TransitCloud currently does not support these protocol. However, Hogia will consider to implement it if there is a business case to do so.

- RTIG Digital Air Interface Protocol
- SIRI VM – Vehicle Monitoring

The other message types in this document are either obsolete, custom or for special purposes. These message types should not be used in any new implementation.

4.2 Identities

Identities are the properties that identifies **vehicles**, **service journeys**, **vehicle schedules** and **waypoints** (stops and other traceable points) within an account.

It is important that a user of TransitCloud have defined a common set of identities so that data can be understood by clients of TransitCloud. Hogia strongly recommends that the usage of identities are investigated and concluded before any data is used in TransitCloud.

4.3 Properties of Messages

This section describes in detail the purpose and usage of some common properties used in messages.

4.3.1 Source Timestamp

The timestamp of a message should reflect the time when the data in a message was captured by a device in the vehicle, not when it was sent. Note that the timestamp should be in UTC, not in local time and no adjustment for daylight saving time.

It is important that position messages have correct timestamp. Therefore, the time provided in the data from the GPS unit should be used in each message. Do not use timestamp from other sources, for example a clock in an on-board computer, even if the computers clock is synchronized by the GPS.

In TransitCloud, the source timestamp is used to set the timestamp of detected events, for example arrivals and departures, and to detect old messages. Messages with an older or equal timestamp compared to the last successfully received message will be discarded.

4.3.2 Sequence Number

Some report formats permit usage of sequence numbers. They are used to detect message drop-outs (e.g. messages lost during transmission).

The sequence numbering should be incremented by 1 for each message sent. When the maximum value is reached, the sequence number should restart from 1 (not zero!). When the device is rebooted or cold started, the sequence number should start from zero.

4.3.3 Position

Positions should be delivered in latitude/longitude; WGS 84, ETRS 89, SWEREF 99 or similar WGS84 like datum. These coordinate systems differ only by a few decimetres. Because the accuracy of a GPS devices is approximately ±10 meters, these coordinate systems can be regarded as equal as far as the requirements for vehicle tracking.

4.3.3.1 Invalid positions

In a report to TransitCloud, the vehicle position must be flagged as valid or invalid. Invalid positions are not used for vehicle tracking. They are counted and logged for diagnostic purposes, for example finding vehicles with a hardware problem, e.g. antenna, cabling or circuits.

TransitCloud regards a position with latitude zero and longitude zero as invalid. This is a position on the Equator west of Africa in the Atlantic Ocean, which never will be a valid place for vehicle tracking.

4.3.3.2 Position quality

It is important that the position data is of good quality. Hogia has experienced that these two factors are important:

- A modern type of GPS-receiver is generally better to deliver high quality position data. Development in this field continues to improve the quality of positioning.
- An antenna mounted with a clear view of the sky with no obstructions from about five degrees elevation and up. This usually requires a roof mounted antenna. An antenna used inside a vehicle will not have as good position quality.

4.3.4 Direction

A GPS device also provides the direction (aka course, heading etc.) of the vehicle. The vehicle tracking in TransitCloud will benefit from this data. If not provided, TransitCloud will internally calculate the direction based on the last positions.

4.3.4.1 Direction at low speeds

At very low speed, some GPS devices will have the positions dragging in a way that the direction is randomly rotating. TransitCloud handles this by letting the users configure a zero speed detection level. When speed drops below this level, the vehicle is regarded as standing still and the last stable direction will be outputted, ignoring the actual received GPS direction.

4.3.5 Speed

A GPS device also provides the speed of the vehicle. The vehicle tracking in TransitCloud will benefit from this data. If not provided, TransitCloud will internally calculate the speed based on the last positions.

4.3.5.1 Speed never zero?

Some GPS devices will under special circumstances not correctly detect the vehicle stopping and starting. When the vehicle in reality is standing still, the position is dragging causing a speed to be detected. TransitCloud handles this by letting the users configure a zero speed detection level. When speed drops below this level, the vehicle is regarded as standing still.

4.3.6 Unit Id

The sending *unit id* is any unique identifier of the device that sends position messages (not to be mistaken for a *vehicle id*). If *unit id* is not available, you must use a message type, where a *vehicle id* can be provided.

Hogia stores permitted *unit id* in a central vehicle inventory. When messages are received, the *unit id* is checked against the vehicle inventory. If the *unit id* is found, the vehicle configuration is loaded; otherwise the message is discarded.

4.3.7 Vehicle Id

If nothing else is agreed, the *vehicle id* must identify a physical vehicle (and not the device). The vehicle id must be the same for the same physical vehicle over time.

If a device is moved from a vehicle to another vehicle, the *vehicle id* must be reconfigured to reflect the id of the new vehicle. If a device is replaced in a vehicle, the new device must send the same *vehicle id* as the replaced device.

The *vehicle id* can be omitted if a *unit id* is provided. Then, the mapping of *unit id* to a vehicle can be configured in Hogia's backend. Still, the *unit id* must be mapped to the real *vehicle id* it is mounted in.

In TransitCloud, a *vehicle id* can be any string and must be unique within an account. The value could for example be the vehicle inventory number or the vehicle registration number.

4.3.8 Driver Id

A *driver id* is a unique identifier identifying each driver within a TransitCloud account.

The *driver id* is optional in TransitCloud. If *driver id* is provided, it makes each event traceable per driver in TransitCloud.

A *driver id* can be any unique identifier that will be available from some device in the vehicle, for example the tachometer or the ticket machine.

4.3.9 Task Id

There are two ways to provide the service journey data (schedule and route) that TransitCloud need for tracking:

1. Submitting a *Service Journey Data Object* to the *Journey Assignment Service*.
2. Setting the *task id* in the **Hogia Extended Position Message**.

4.3.9.1 Submitting a Service Journey Data Object

This option means that an external system creates a Service Journey data object that corresponds to the data contract defined by TransitCloud *Journey Assignment Service*, and submits this object, telling which vehicle that will operate this service journey.

4.3.9.2 Setting the *task id* in the Hogia Extended Position Message

Setting the *task id* in a **Hogia Expended Position Message** tell TransitCloud which route and timetable to track. The task id must be present in each message as long as the vehicle performs the task.

The *task id* can be any string and can be the identity of the current service journey, the identities of current and following service journeys or the identity of a vehicle schedule.

Each time a change of the task id is detected, TransitCloud will load data for that task from the backend system. TransitCloud will automatically detect when to start and end tracking of each service journey.

Below some examples of a task id. The examples also show valid combinations and which character to use as separator. Other identification schemes can be agreed upon after consulting Hogia.

URL examples:

123.456.lines = journey 123 on line 456.

123.456.lines,124.456.lines = current journey 123 on line 456 to be followed by journey 124 on line 456. These values shall be separated by a comma.

123.456.lines;9876.22.blocks = current journey 123 on line 456 and current block 9876 for operator 22. These values shall be separated by a semicolon.

PubTrans GID examples:

9015200045600123 = PubTrans GID for current journey 123 on line 456 at transport authority 200.

9015200045600123,9015200045600124 = PubTrans GID for current journey 123 on line 456 at transport authority 200 to be followed by journey 124 on line 456 at transport authority 200. These values shall be separated by a comma.

9015200045600123;9041200002209876 = PubTrans GID for the current journey 123 on line 456 and the current for block 9876 for operator 22. These values shall be separated by a semicolon.

4.3.9.3 Changing the Task Id before end of journey

Sometimes the driver behaviour is to change to next journey before current journey is ended. This is possible. TransitCloud will still track the current journey and automatically detect when the next journey is started.

4.3.9.4 How often should the task id be reported?

As often as possible, as part of a **Hogia Extended Position Message**. It is important that the task-id is sent continuously and not only when it changes.

4.3.9.5 What if task id is not available in the vehicle?

Sometimes the task id is not available in the vehicle, but instead in your back office system. If this is the case, please consult Hogia. We will help you to find a way to retrieve the task-id separately from your back office system and send it to TransitCloud.

4.3.10 Account Id

The *account id* identifies the customer and which configuration TransitCloud shall use, and which other external systems to communicate with. For customers using Hogia PubTrans®, the *account id* is the same number as *transport authority number*.

The *account id* can be included in some of the message types; otherwise it will be derived from which account id that is attached to a vehicle in Hogia's central configuration.

4.4 Configuration

The following configuration data is needed for each vehicle in TransitCloud. These data are stored in the TransitCloud *Vehicle Inventory*.

4.4.1 Account

Each customer must have a TransitCloud account. The following properties is needed:

- **Account Id**: A unique identification of the account.
- **Customer Name**: The name of the customer.

4.4.2 Vehicle

Each vehicle using TransitCloud must be added to the central vehicle inventory. The following properties must be specified for each vehicle:

- **Account Id**: The id of the customer account to which the vehicle belongs.
- **Vehicle Id**: An identification of the vehicle that must be unique within the account.
- **Alias**: If a device id or alternative vehicle id shall be used to find a vehicle, this is configured as an alias.
- **Transport Mode**: BUS, TRAIN, FERRY, METRO etc.

In addition, the Vehicle Inventory contains a number of optional parameters for vehicle configuration. These are described in detail in the *Vehicle Inventory Manual*. Knowledge of these detailed configuration is not necessary for implementing the message format described in this document.

5 Hogia Standard Position Message

The packet is encoded as an UPD-packet as described below. This is the preferred message format that should be used for new implementations. All values should be encoded so they can be read using the standard methods of the **Binary Reader** in Microsoft .NET Framework.

No	Off	Field name	Data type	Content	Comment
1	0	Message type	Byte	Type of message = 1	This value determines how the rest of the message should be interpreted.
2	1	Priority	Byte	Priority of message	Used to dispatch messages to different queues. Priority is 1 to 255, 1 is highest priority. Default is 127.
3	2	Unit identity	Byte (8)	E.g. MAC-address or similar.	Fixed identity of sending unit, see 4.3.
4	10	Sequence number	UInt16	0-65535	At start-up, the counting restarts from 0. When max value 65535 is reached, the counting continues from 1 (not 0).
5	12	Timestamp	UInt32	Milliseconds since midnight (UTC)	Time of fix.
6	16	WGS Latitude	Single	-90,00000° - +90,00000°	IEEE 32-bits floating point number with single precision.
7	20	WGS Longitude	Single	-180,00000° - +180,00000°	
8	24	Speed	UInt16	0 – 65536	Speed in steps of 0,01 m/s
9	26	Direction	UInt16	0 – 35999	Direction in steps of 0.01°
10	28	Position Quality	Byte	0 - 255	See 5.1.
11	29	Signals	Byte	4 signals of 2 bits each	See 5.3.
12	30	Distance	UInt32	0 – 4 294 967 295 m	See 5.4.

Packet length: 34 bytes. Standard port for receive: 2011.

5.1 Sequence Number

Hogia Standard Position Message and **Hogia Extended Position Message** can be used mixed. Therefore, the sending application should use a shared sequence number counter. This means that the sequence counter should be incremented with one, regardless of which of the two message types that are sent.

5.2 Position quality

This field consist of two values, four bits each.

5.2.1 Type of fix

Bit 1-4 of the Quality field specifies the type of GPS fix. If the fix is **invalid** or **undefined** or if the *latitude* and *longitude* both are zero, the position message will be discarded by Hogia TransitCloud.

The values 0 to 8 is NMEA standard for GPS devices. Values 10-14 is alternative position techniques used by mobile handsets. These are regarded as not precise enough to be used for vehicle tracking in TransitCloud.

If a hybrid positioning is used, then report it as the most precise technology used, in most cases a GPS.

Value	Meaning	Type in TransitCloud
0	Invalid fix.	Invalid
1	Fix.	Normal
2	Differential fix.	Normal
3	PPS fix.	Normal
4	Real Time Kinematic (RTK) fix.	Normal
5	Float Real Time Kinematic (Float RTK) fix.	Normal
6	Estimated fix.	Simulated
7	Manual fix.	Simulated
8	Simulated fix.	Simulated
10	Wi-Fi	Normal
11	Handset fingerprinting.	Normal
12	Handset cell identification.	Normal
13	Cellular network forward link.	Normal
14	Cellular network triangulation.	Normal
Other		Undefined

5.2.2 Fix quality

The fix quality is a value that indicates the accuracy of the position in meters. The value corresponds to a maximum deviation in meters. When the fix quality is undefined, the position will be used for all purposes in TransitCloud.

Value	Max deviation	Value	Max deviation	Value	Max deviation	Value	Max deviation
0	Undefined	4	10 meters	8	200 meters	12	5000 metes
1	1 meter	5	20 meters	9	500 meters	13	Over 5000 meters
2	2 meters	6	50 meters	10	1000 meters	14	Reserved
3	5 meters	7	100 meters	11	2000 meters	15	Reserved

5.2.3 Example of setting the Quality field

A normal fix (value 1) and a max deviation value of 4 (10 meters) will give a Quality with value of 65. The calculation is made as follows: Quality = *type-of-fix* + (*fix-quality* × 16) or 33 = 1 + (4 × 16). If fix quality is not provided, Quality is simply equal to *type-of-fix*.

5.3 Signals

NOTE: The signal *Door Open* is removed, use the signal *Door Released*. A new signal *Power On* takes the place of the removed *Door Open* signal.

A byte divided in four groups of two bits. Bit 1 denotes if the signal is available or not, bit 2 denotes the signal itself. If no signals are available, set byte to zero.

- 00 = the signal is not available and therefore its value is UNDEFINED.
- 01 = the signal is not available due to some technical problem.
- 10 = the signal is available and is OFF.
- 11 = the signal is available and is ON.

The four signals are described below:

No	Bits	Signal	Comment
1	7 and 8	In Service	ON indicates that the vehicle is expected to be in service. OFF indicates that the vehicle is expected to be out of service. Example of detection and usage, see 5.3.1.
2	5 and 6	Stop Requested	ON indicates that there is a request for stopping, e.g. a passenger that has pressed the STOP button. This signal should be reset when doors are release or opened. OFF indicates that the request has been reset or cancelled.
3	3 and 4	Door Released	ON indicates that at the lock of at least one door has been released and can be manoeuvred by passengers. OFF indicates that all doors are locked for passengers.
4	1 and 2	Power On	ON indicates that the main power of the vehicle is on; OFF indicates that main power is off. See additional description in section 0 of the use this signal.

5.3.1 In Service

NOTE: The feature is only available in TransitCloud 3.

The signal can be used to signal that a vehicle is expected to operate according a schedule. When this signal is explicitly ON, but no service journey is assigned to the vehicle, special measures can be performed. One example is the automatic journey assignment feature in TransitCloud 3.

When the signal is UNDEFINED it is assumed that a vehicle is in service.

To better understand when a vehicle is in service or not, consider the following rule of thumb: When the vehicle's head signs displays a line number and/or destination, this means that the vehicle is in service. On the other hand, displaying "Not in service", the vehicle is not in service.

5.3.2 Stop Requested

NOTE: The feature described below must be explicitly requested by customers and only available in TransitCloud 3.

The signal has effect only on on-board information services to indicate to drivers and passengers. If this signal is explicitly set, it will be used by the TransitCloud On-board service.

5.3.3 Door Released

If the signal is explicitly set to ON or OFF, it can be used to further qualify arrival and departure times at stops.

NOTE: The feature described below must be explicitly requested by customers and only available in TransitCloud 3.

When a vehicle is stopping at a stop (the standard arrival time is recorded) and the signal a while after goes from OFF to ON, the arrival time will be modified to reflect the time when passengers actually can begin alight.

When a vehicle currently at a stop has signal ON (doors released) and then changes to OFF (doors locked) a potential departure time can be registered. The last OFF-signal will be used as a departure time.

5.3.4 Power On

NOTE: The feature is only available in TransitCloud 3.

This signal is indented to be used when the position devices is sending data 24/7/365, also when the vehicle is parked and main power is turned off. When a vehicle is powered off, it indicates that it cannot be in service any longer. When power is detected as off, TransitCloud will:

- Abort any current tracked task.
- Ignore the signal *In Service*.
- Ignore the Task field (see Hogia Extended Position Message).

5.4 Distance

NOTE: The feature described below must be explicitly requested by customers and only available in TransitCloud 3.

If a running distance is available in a vehicle, it is recommended that the value is used in the position message.

It is permitted to restart the distance from zero at start-up of on-board equipment, but not else.

6 Hogia Extended Position Message

The packet is encoded as an UDP-packet as described below. All values should be encoded so they can be read using the standard methods of the **Binary Reader** in Microsoft .NET Framework.

No	Off	Field name	Data type	Content	Comment
1	0	Packet type	Byte	Type of message = 2	This value determines how the rest of the message should be interpreted.
Field 2-12 are identical to Hogia Standard Position Message					
L13	34	Length of field 13	Byte	0-255	Number of bytes.
13		Vehicle id	String	Identity of vehicle, see 4.3.7.	Optional. Must be a vehicle fixed unique value, e.g. BUS FMS VI field.
L14		Length of field 14	Byte	0-255	Number of bytes.
14		Driver id	String	Identity of current driver, see 4.3.8.	Optional. Could be BUS FMS DI, or unique id from other available source in vehicle.
L15		Length of field 15	Byte	0-255	Number of bytes.
15		Task Id	String	Identity of current task(s), see 4.3.9.	Mandatory. Current tasks assigned to the vehicle, e.g. unique vehicle journey identifier.
L16		Length of field 16	Byte	0-255	Number of bytes.
16		Account Id	String	Identity of customer, see 4.3.10	Optional.

Packet length: minimum 38 bytes, maximum 1058 bytes. Standard port for receive: 2011.

6.1 Sequence Number

Hogia Standard Position Message and **Hogia Extended Position Message** can be used mixed. Therefore, the sending application should use a shared sequence number counter. This means that the sequence counter should be incremented with one, regardless of which of the two message types that are sent.

6.2 String Data Type and Length

A string is encoded as ASCII. An empty string value is encoded as a zero length string. Maximum string length is 255. The length of the string must be given in the preceding length field for the string.

Example: The word VEHICLE should be encoded as hex= 56454849434C45 and the length should be 7.

7 Hogia Binary Data Message

Not yet implemented. This message type can only be used after agreement with Hogia.

The purpose with this message is to make it possible to record raw binary data in the vehicle, for interpretation centrally in TransitCloud.

The packet is encoded as an UDP-packet as described below. All values should be encoded so they can be read using the standard methods of the **Binary Reader** in Microsoft .NET Framework.

No	Off	Field name	Data type	Content	Comment
1	0	Message type	Byte	See table below.	This value determines how the rest of the message should be interpreted.
2	1	Priority	Byte	Priority of message	Used to dispatch messages to different queues. Priority is 1 to 255, 1 is highest priority. Default is 127.
3	2	Unit identity	Byte(8)	Fixed identity of sending unit, see 4.3.	E.g. MAC-address or similar.
4	10	Sequence number	UInt16	0-65535	At start-up, the counting restarts from 0. When max value 65535 is reached, the counting continues from 1 (not 0).
5	12	Timestamp	UInt32	Milliseconds since midnight (UTC)	Time of fix.
6	16	Length of field 7	UInt16	Length of binary data	Number of bytes.
7	18	Binary data	Byte()	Any binary data	Max 65489 bytes.

Packet length: minimum 38 bytes, maximum 65507 bytes. Standard port for receive: 2015.

NOTE: The maximum length is constrained by the maximum UDP packet length.

7.1 Message types

To be able to distinguish between different types of binary data, each type must have a unique message type. Please, consult Hogia for adding new types.

101	Mobitec ICU 400 communication frame. This message type is reserved for user by Hogia.
255	Performance test message. Binary data contains the Vehicle Ref as String. This message type is reserved for user by Hogia.

8 Hogia Older Standard Position Message

NOTE: This format is only for backwards compatibility. It should not be used in any new implementations.

No	Field name	Content	Length	Encoding
1	Protocol version	Version × 10 + revision of the message layout.	1 byte	Binary
2	Message length	Total number of bytes in the message.	1 byte	Binary
3	Packet type	Defines the interpretation of the message content from byte 10 and onwards.	1 byte	Binary
4	Id of sender	The units MAC-address, or other identifier	6 bytes	Binary
5	Latitude	WGS 84	8 bytes	See 8.3
	Longitude			
6	Position quality	No fix/invalid fix | fix | differential fix.		
7	Timestamp	Number of seconds since midnight UTC (0-86400 including leap second) for when position fix was detected.	1-3 bytes	See 8.4 and 4.3.1.
8	Speed	Speed in meters per second, 0-255	1 byte	
9	Direction	Direction in steps of 2°, 0-179	1 byte	
10	DOP	DOP × 10. Value zero indicates that the data is not available.	1 byte	
11	Sequence number	At start-up, the counting restarts from 0. When max value 255 is reached, the counting continues from 1 (not 0).	1 byte	

Packet length: 22-24 bytes. Standard port for receiving: 2010.

8.1 Protocol version and packet type

The current protocol version is 10.

Valid packet types are 100, 101, 150, 151, or 152. There are no difference between these packet types.

8.2 Id of sender

The id of sender should uniquely identify the physical unit that sends the reports. In TransitCloud, this id is mapped to a vehicle. It is possible to map several units to the same vehicle.

If the device is a unit with an Ethernet MAC-address, it is recommended to use this value as sender id. However, any unique number that fits in the field may be used.

8.3 Position and quality according OVL G

OVLS (Open protocol for interfacing Vehicle Location Subsystems) is a Swedish standard (SS 3652).

The encoding of latitude, longitude and quality is described in Swedish Standard SS-3652, revision 1, 1996-11-27, page 27, parameter #155 2-D SPDC. The value is encoded without the beginning "G".

8.4 Multi-byte integers

The value is contained in one or several bytes. When the most significant bit is set, it denotes that the value is continued in the next byte. The following 7 bits is a scalar value encoded with the most significant byte first (big endian).

An integer consists of N bytes, hereof the N-1 first bytes with a value >= 128 and the Nth byte a value < 128. The bytes are ordered with the most significant value first.

Examples:

Value 127 is stored in one byte as: 01111111.

Value 128 is stored in two bytes as: 10000001 00000000.

Value 789 is stored in two byte as: 10000110 00010101 (134-128)*128+21.

Value 987654 is stored in three byte as: 10111100 10100100 00000110 = (((188-128)*128+(164-128))*128)+6.

This is described in detail in http://www.w3.org/TR/wbxml/#_Toc443384895.

9 Hogia Extended GPS RMC

NOTE: This format is only for backwards compatibility. It should not be used in any new implementations.

GPR RMS is a message type that has been standardised by NMEA. In the reporting to Hogia TransitCloud, this message type has been extended. Extensions are marked with yellow background and labelled H1-H6. If not stated otherwise, the separator between fields is comma. Note that *comma*, *asterisk* and *semicolon* cannot be a part of any value. The packet is encoded as ASCII characters in an UDP packet.

No	Field name	Content
1	Packet type	”$GPRMC” where RMC stands for *Recommended Minimum sentence C*
2	Time of fix	Format “HHMMSS” for hour, minutes and seconds, time in UTC.
3	Status	“A” = active or “V” = invalid.
4	Latitude	Degrees and minutes as decimal, example ”4807.038” is 48°7.038’
5	Hemisphere	Hemisphere of latitude: “N” = north, “S” = south.
6	Longitude	Degrees and minutes as decimal, example”01131.000” is 11°31.000’
7	Halvklot	Hemisphere of longitude: “E” = east, “W” = west.
8	Speed	Knots in decimal. Example “022.4” is 22.4 knots.
9	Heading	Degrees in decimal. Example “084.4” is 84.4°.
10	Date of fix	Format “DDMMYY”. Example “230394” is 23 mars 1994.
11	Magnetic variation	Example “003.1”är 3.1°.
12	Hemisphere	The magnetic variations hemisphere: “E” = east, “W” = west.
13	Kind of fix	The value can be A=autonomous, D=differential, E=Estimated, N=not valid, S=Simulator or empty. NOTE: This field was introduced from NMEA 2.3. Hogia TransitCloud can receive both the older and the newer version of the packet.
	Other separator	Asterisk (*) instead of comma.
	Checksum	Hexadecimal value. Example “6A”
Custom fields added by Hogia:		
H1	Sender Id	Unique id of sending unit (or empty if not available), see 4.3.6.
H2	Vehicle Id	Id of vehicle where the sending unit is installed (or empty if not available), see 4.3.7.
H3	Driver Id	Ids of responsible staff on-board the vehicle, i.e. drivers and/or other persons. Each id should be separated by a semicolon, see 4.3.8.
H4	Task Id	Ids of current tasks, separated by a semicolon, see 4.3.9.
H5	Account Id	Id of data provider or data owner, see 0.

Packet length: >57 bytes. Standard port for receive: 2012.

9.1 Example of packets

H1=0009D8021D34
H2=56
H3=523
H4=9015014001100025
H5=VT

Pre-NMEA 2.3 version:

```
$GPRMC,123519,A,4807.038,N,01131.000,E,022.4,084.4,230394,003.1,W*6A,0009D8021D3
4,56,523,9015014001100025,VT
```

NMEA 2.3 version:

```
$GPRMC,123519,A,4807.038,N,01131.000,E,022.4,084.4,230394,003.1,W,A*6A,0009D8021
D34,56,523,9015014001100025,VT
```

9.2 Train Id used by Oxyfi

This format of train id is only used by Oxyfi in the **Hogia Extended GPS RMC**.

A train is normally assigned a public and a technical (internal) number. The public train journey number are encoded as *trainnumber*.public.trains.se, and the technical train journey number are encoded as *train-number*.internal.trains.se. If both is available, they can be provided, separated by a semicolon.

It is recommended that train numbers are assigned at least five minutes ahead of the scheduled start time. When delays occur, the current train numbers must be reported until the train reaches its final station. This means that both current and following train journey numbers will be reported at the same time.

Example:

This example shows how both the public and the technical train identity can be provided in the same task id. The values are separated by semicolon.

```
93.public.trains.se;9301.internal.trains.se
```

10 RTIG Digital Air Interface Protocol

NOTE Not yet implemented. Support for receiving *RTIG Digital Air Interface Protocol* (DAIP) can be provided after agreement with Hogia. This chapter describes the requirements that must be fulfilled by the implementer of RTIG DIAP in order to deliver data to TransitCloud.

10.1 Required messages

This section is based on the document *RTIG Digital Air Interface Protocol – RTIGT030-1.2*, referred to as DAIP.

10.1.1 Session

Prior to sending journey information and positions, a session must be established according DAIP specification, between the vehicle and a server. A session should be established as soon as the vehicle's main power is turned on. The vehicle session should be active until the vehicles main power is turned off.

On customer request, Hogia will consider implementing a RTIG DAIP conformant server with support for the required messages.

10.1.2 Journey Information

Hogia TransitCloud must know the current working of the vehicle, or if the vehicle is not in service. DAIP defines three message types for this purpose:

- Journey Details #30
- Journey Details (basic) #31
- End of Journey #39

Hogia TransitCloud uses the following fields:

Field	Alt 1	Alt 2
Service Code	Required	Required
Journey Number	Required	
Journey Scheduled Start Time		Required
Direction		Required
First stop ID	Optional	Required
Destination stop ID	Optional	Required

Alt 1 And Alt 2 is two different ways of identifying a service journey. A value marked as optional will be used by TransitCloud if it is provided in the message. Implementation of Alt 1 and Alt 2 will depend on customer requirements.

If customer is using Alt 1, the following option can be provided upon request: If *First stop ID* and/or *Destination stop ID* is provided, and any of these are not matching the first and last stop of schedule found using *Service Code* and *Journey Number*, then the schedule will be truncated to reflect the stretch defined by *First stop ID* and/or *Destination stop ID*.

When *End of Journey* is received, it is assumed that the vehicle has reached the *Destination stop ID*. If this is not the desired behaviour, then alternative behaviour must be suggested by the customer.

10.1.3 Position Updates

Hogia TransitCloud requires frequent position updates. Positions should be sent each second from the session is established until it terminates.

DAIP defines two message types for this purpose:

- Position Update Message #40
- Position Update Message (basic) #41

Hogia TransitCloud uses the following fields:

- Position – latitude
- Position - longitude
- Bearing
- Position Quality
- Distance travelled from last stop (optional)

10.1.4 Events

Hogia TransitCloud will perform better if certain events are reported. DAIP defines a message type for this purpose:

- Event (OBU to centre) #50

Hogia TransitCloud will use the following message types:

Message Type	Message Code	TransitCloud usage
Emergency	0, 1 and 2	Handled only if agreed upon. These messages needs to terminate at a manned centre for further actions.
Emergency	Diverting	Will affect route tracking by setting status to off-route until detected back on route again.
Emergency	Abandoning Journey	Aborts tracking of current route. It is expected that this message is sent when the action actually occurs.
Emergency	Curtailing Journey	Aborts tracking of current route. It is expected that this message is sent when the action actually occurs.
Service Messages	Request PMR Radio Session	Handled only if agreed upon. These messages needs to terminate at a manned centre for further actions.
Vehicle Status	0, 1, 2, and 3	Handled only if agreed upon. These messages needs to terminate at a manned centre for further actions.
Free Format Text Messages	0 and 1	Handled only if agreed upon. These messages needs to terminate at a manned centre for further actions.

11 SIRI VM – Vehicle Monitoring

NOTE Not yet implemented. Support for SIRI VM can be provided after agreement with Hogia.

The SIRI Vehicle Monitoring service (VM) provides information about of the current location and expected activities of a particular vehicle.

The SIRI VM feed to TransitCloud is currently not implemented. Due to the many ways to utilize the SIRI protocol, support for feeding TransitCloud using a SIRI VM feed can be provided first after an agreement of which data to provide and verification of that position data for all vehicles can be delivered each second by the data provider.

12 BIMS Status packet

NOTE: This format is only for backwards compatibility. It should not be used in any new implementations.

The BIMS status packet is currently implemented by Hogia to fetch data from ATRONs on-board ticketing equipment.

12.1 Current Status

This message is based on data available from ATRONs on-board equipment, *Current position binary data message* described in document *BIMS – Definition of the interface between ATRON's FRx and the OBU Trivector*, version 1.1, 2011-05-23, section 4.1.1.6.

Off	Field name	Data type	Description	Comment
0	Packet type	Byte	Type of message= 201	This value determines how the rest of the message should be interpreted.
1	Unit Id	Byte(6)	Fixed identity of sending unit.	MAC-address for sending unit.
7	Sequence number	UInt16	0-65535	At start-up, the counting restarts from 0. When max value 65535 is reached, the counting continues from 1.
9	Timestamp	UInt32	Milliseconds since midnight (UTC)	Time of data fetch from ticket machine.
13	Current driver	UInt32	Logged in driver id	Field *driver_id* in packet from ticket machine.
17	Current vehicle schedule	UInt32	Logged in vehicle schedule	Field *service_id* in packet from ticket machine.
21	Current line	UInt32	Logged in line number	Field *line_id* in packet from ticket machine.
25	Current journey	UInt32	Logged in journey number	Field *trip_id* in packet from ticket machine.
29	Current stop	UInt32	Next stop to arrive	Field *stop_id* in packet from ticket machine. When a new line/journey is selected, the value is 0 until the driver has confirmed that the vehicle is at first stop on route.
33	Door state	Byte	Open/closed	Field *door* in packet from ticket machine. Value 1 if the vehicle is at a stop and the door is open; otherwise 0.
34	Service state	Byte	Not started, continue or logged off.	Field *report_status* in packet from ticket machine. See 12.3.1.

Packet length: 31 bytes. Standard port for receive: 2013.

12.2 Error message

Off	Field name	Data type	Description	Comment
0	Packet type	Byte	Type of message= 202	This value determines how the rest of the message should be interpreted.
1	Unit Id	Byte(6)	Fixed identity of sending unit.	MAC-address for sending unit.
7	Sequence number	UInt16	0-65535	At start-up, the counting restarts from 0. When max value 65535 is reached, the counting continues from 1.
10	Status codes	Byte	Connection status.	See 12.3.2.

Packet length: 11 bytes. Standard port for receive: 2013.

12.3 Special Encodings

12.3.1 Service Journey Status

Indicates status of the current service journey:

1. The vehicle is in progress on the current service journey.
2. The driver has accepted the service journey, but not yet indicated that the vehicle is at first stop on current service journey.
3. The driver has indicated that the vehicle is at first stop on current service journey.
4. The ticket machine has returned to the route after detected off-route.

12.3.2 Status codes

Bit value 0 means NO, value 1 means YES.

- Bit 0: Battery power is on?
- Bit 1: Main power is on?
- Bit 2: Contact with RS485-adapter for ticket machine?
- Bit 3: Contact with ticket machine?

13 References

Document	Description
http://www.nmea.org/	NMEA
http://en.wikipedia.org/wiki/Dilution_of_precision_(GPS)	Explanation of DOP-values in GPS data.
https://en.wikipedia.org/wiki/Mobile_phone_tracking	Explanation of techniques used for mobile phone tracking.