TransitCloud Position Messages
Formats and Encoding

Revision: 1.1
Date: 2014-05-15

TransitCloud Position Messages
Formats and Encoding

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

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.

1.1 General Requirements

Please, also find additional details in chapter 2 Guidelines.

1.1.1 Account

Each user of TransitCloud shall have an account id, provided by Hogia.

1.1.2 Common References

It is important that all stakeholders of TransitCloud have agreed to use a common set of identities of vehicles, tasks and waypoints etc. 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.

1.1.3 Data Reports

There are two specific types of data that must be continuously provided to TransitCloud:

1. The **position**, **direction**, and **speed** of the vehicle. This is the data normally available from a GPS device. This data should be sent each second, preferably directly from a device in the vehicle to TransitCloud and not passing any intermediate software/servers.
2. The **current task** of the vehicle. In public transport this is usually the service and trip number. This data should be sent with maximum 30 seconds interval, preferable as often as the position. The source of this information could be the on-board computer in the vehicle, or a central source.

Further details regarding reporting to TransitCloud is found later in this document.

1.2 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.

For low frequency position reports, individual vehicles can be configured to use an alternative and coarser tracking algorithm which only detect passing/missing waypoints.

2 Guidelines

2.1 Which Message Types to Use?

Normally, you implementation one or two message types. I you implement from scratch, consider using one or both of the Hogia Messages below.

- **Hogia Standard Position Message,** described in chapter 3, 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 4, 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**, described in chapter 6.
- **SIRI VM – Vehicle Monitoring**, described in chapter 7.

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.

2.2 Common Properties of Messages

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

2.2.1 Timestamp

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

If sequence numbers are not used, the timestamp is used to detect old messages. Messages with an older or equal timestamp compared to the last successfully received message will be discarded. Note that this behaviour is overridden when sequence numbers are used.

2.2.2 Sequence Number

Some reports permits usage of sequence numbers. They are used to detect messages out of sequence (i.e. old messages) and 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.

If sequence numbers are used, they are used to detect old messages. Messages with a lower or equal sequence number than the last successfully received message, will be discarded. In this case, timestamps must be equal or greater than the timestamp of the last successfully received message.

2.3 Movement Data

This section describes the data needed to track the movement of a vehicle.

2.3.1 Position

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

It is important that waypoint positions and vehicle positions are using the same (or similar) coordinate reference system. If waypoints or vehicle positions are not available in latitude/longitude, TransitCloud currently supports conversion of grid coordinates from UTM zone 32-34 to ETRS 89 and from the older Swedish RT 90 to SWEREF 99.

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 does by default regard 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. Instead, it often indicates an undefined value, as computers often assigns numbers with the default value of zero.

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.

2.3.2 Direction/Heading

A GPS device also provides the direction 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.

At very low speed, some GPS devices will have the positions dragging in a way that the direction is randomly rotating. This is handled in TransitCloud, which means that the last stable direction will be outputted, ignoring the actual GPS direction.

2.3.3 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.

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.

2.4 Identity

This section describes the different types of identities that are used in TransitCloud. A few is mandatory, other are optional.

It is important that all stakeholders of TransitCloud have agreed to use 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 sent in to TransitCloud.

2.4.1 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.

2.4.2 Vehicle Id

If nothing else is agreed, the *vehicle id* must identify a physical vehicle (and not the device). 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.

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

If nothing else is agreed, the *vehicle id* must be a value that is fixed over time for a physical vehicle. In TransitCloud, a *vehicle id* can be any string. The value could for example be the vehicle inventory number or the vehicle registration number. Other identification schemes can be agreed upon after consulting Hogia.

2.4.3 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.

2.4.4 Task Id

A *task id* can be any string. It must uniquely identify the task in am unambiguous way. It could be the identity of a single service journey or the identity of a vehicle schedule containing a sequence of service journeys. Each time a change of the current task id is detected, TransitCloud will load data for that task. If a sequence of service journey are loaded, TransitCloud will automatically detect when to start each service journey in the sequence.

At the time TransitCloud detects a change of the current *task id* there must be only one match in the schedule data for the provided id. For tasks with limited duration times, i.e. journeys on bus routes, this is normally

not a problem. Tasks with run times that exceeds 20 hours often needs special consideration. Please, consult Hogia if this is the case.

Here some examples of a task id:

- Bus service journeys can be encoded as *journeynumber.linenumber.***lines**.
- Train journeys can be encoded as *trainnumber*.**trains**.
- For service journeys registered in Hogia PubTrans®, the service journey GID can be used.
- Other identification schemes can be agreed upon after consulting Hogia.

Examples:

123.456.lines = Uri for journey 123 on line 456.

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

2.4.5 Account Id

The *account id* identifies the customer and which configuration TransitCloud shall use, and which other external systems to communicate with.

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.

Each customer receives an account id from Hogia.

2.5 Configuration

2.5.1 Account

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

- **Account Id**: A unique identification of the account, for example the customers domain name.
- **Customer Name**: The name of the customer.

2.5.2 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.
- **Unit Id**: One or several id's of unit that sends data to TransitCloud.
- **Transport Mode**: BUS, TRAIN, FERRY, METRO etc.
- **Expected max interval of position messages**, to detect loss of position messages.
- **Zero speed detection level**, which is used to determine when a vehicle has stopped.

3 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)	Fixed identity of sending unit, see 2.2.	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	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	Speed in steps of 0,01 m/s	0,00 – 655.36 m/s
9	26	Heading	UInt16	Heading in steps of 0.01°	0,00 – 359.99°
10	28	Quality	Byte	Quality for GPS.	See 3.1.
11	29	Signals	Byte	4 signals of 2 bits each	See 3.3.
12	30	Distance	UInt32	Vehicle distance in steps of 5 meters	Based on FMS VDHR with resolution in steps of 5 meters. Set to zero if not available.

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

3.1 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.

3.2 GPS-quality

3.2.1 Type of fix

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

Value	Meaning	Type in TransitCloud
0	Invalid fix.	Invalid
1	Fix.	Normal
2	Differential fix.	Differential
3	PPS fix.	Differential
4	Real Time Kinematic (RTK) fix.	Differential
5	Float Real Time Kinematic (Float RTK) fix.	Differential
6	Estimated fix.	Simulated
7	Manual fix.	Normal
8	Simulated fix.	Simulated
Other		Undefined

3.2.2 Fix quality

Bit 5-8 of the Quality field specifies the HDOP value as integer value 1-15. If the HDOP value is higher than 15, then set this value to 15. HDOP is the *horizontal dilution of precision* as described by this article: http://en.wikipedia.org/wiki/Dilution_of_precision_(GPS).

If the value is not provided, it should be set to zero, which is interpreted as 'undefined' by Hogia TransitCloud.

3.2.3 Example of setting the Quality field

A normal fix (value 1) and a HDOP value of 2 will give a Quality with value of 33. The calculation is made as follows: Quality = *type-of-fix* + (*fix-quality* × 16) or 33 = 1 + (2 × 16). If fix quality is not provided, Quality is simply equal to *type-of-fix*.

3.3 Signals

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:

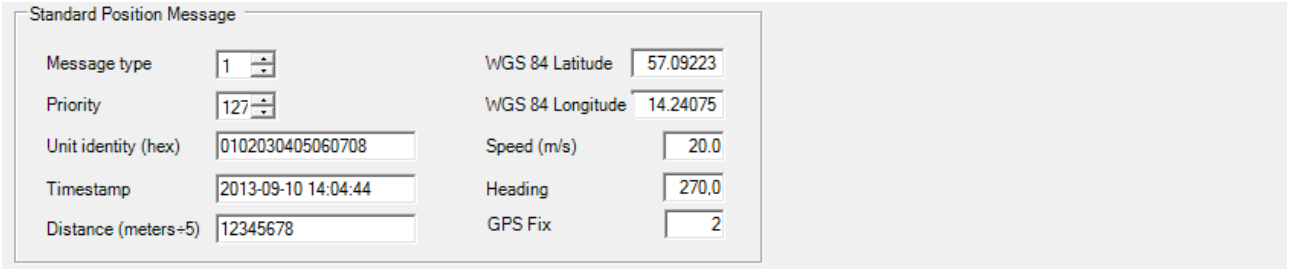
No	Bits	Signal	Comment
1	7 and 8	In Service	ON indicates that the vehicle is expected to be in service. OFF indicates that the vehicle is expected to be out of service. Example of detection and usage, see 3.3.1.
2	5 and 6	Stop Requested	ON indicates that there is a request for stopping, e.g. a passenger 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	Door Open	ON indicates that at least one door is open. OFF indicates that all doors are closed.

3.3.1 In Service

The signal *In Service* can be used to signal that a vehicle is operating according a schedule. The purpose with this signal is to tell the central system that it should expect the vehicle to be assigned to some task. If the central system detects no assigned task, it can perform some predefined action.

In public transport, the signal can for example be detected through the head sign control. If it is set to some line number or destination, the signal In Service can be set to ON; otherwise if it is set to "Not in service" the signal can be set to OFF.

3.4 Message Example



A message with the values above and sequence number 1 will have the following binary representation:

HEX=017F0102030405060708010080298302725E64421DDA6341D007786921024E61BC00

4 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 2.4.2.	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 2.4.3.	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 2.4.4.	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 2.4.5	Mandatory.

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

4.1 Sequence Number

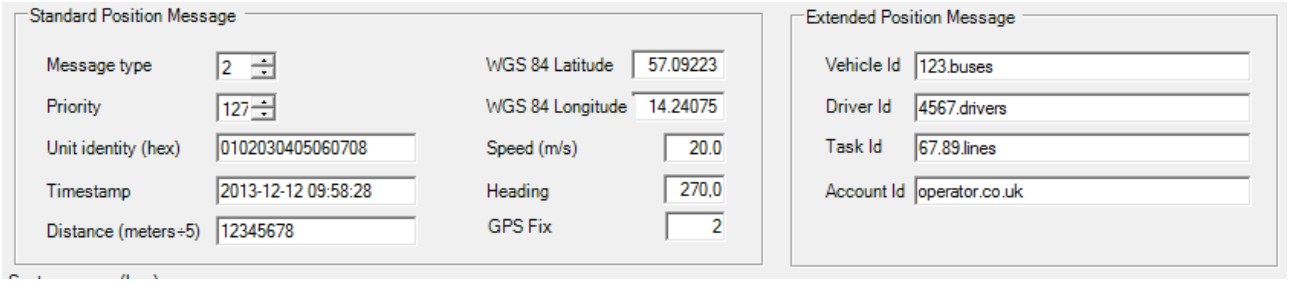
Hogia Standard Position Message and **Hogia Extended Position Message** can be used mixed. Therefore, the sending application should use a shared sequence number counter. This means that the sequence counter should be incremented with one, regardless of which of the two message types that are sent.

4.2 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.

4.3 Message Example



A message with the values above and sequence number 1 will have the following binary representation:

HEX=
027F0102030405060708020070F72202725E64421DDA6341D007786921024E61BC00093132332E62757365730C34
3536372E647269766572730B36372E38392E6C696E65730E6F70657261746F722E636F2E756B

5 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 2.2.	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. The maximum length is constrained by the maximum UDP packet length. Standard port for receive: 2015.

5.1 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.

6 RTIG Digital Air Interface Protocol

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.

6.1 Required messages

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

6.1.1 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.

6.1.2 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.

6.1.3 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)

6.1.4 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.

7 SIRI VM – Vehicle Monitoring

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.

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 2.2.1.
8	Speed	Speed in meters per second, 0-255	1 byte	
9	Heading	Heading 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 2.4.1.
H2	Vehicle Id	Id of vehicle where the sending unit is installed (or empty if not available), see 2.4.2.
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 2.4.3.
H4	Task Id	Ids of current tasks, separated by a semicolon, see 2.4.4.
H5	Account Id	Id of data provider or data owner, see 2.4.5.

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
```

10 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.

10.1 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 10.3.1.

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

10.2 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 10.3.2.

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

10.3 Special Encodings

10.3.1 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.

10.3.2 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?

11 Special Items

11.1 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
```

12 References

Document	Description
http://www.nmea.org/	NMEA
http://en.wikipedia.org/wiki/Dilution_of_precision_(GPS)	Explanation of DOP-values in GPS data.