



# Roadside Unit

MQTT-based Management Interface

For version 02.04.11

YUNEX  
TRAFFIC



# Contents

	<b>Contents</b>	<b>1</b>
	<b>List of Tables</b>	<b>2</b>
	<b>List of Figures</b>	<b>3</b>
	<b>Abbreviations and Definitions</b>	<b>4</b>
<b>1</b>	<b>Interface Specification</b>	<b>6</b>
1.1	Introduction . . . . .	6
1.2	MQTT Protocol . . . . .	6
1.2.1	Topic Structure . . . . .	6
1.2.2	Message Parameters . . . . .	6
1.2.2.1	Quality of Service . . . . .	7
1.2.2.2	Retained Flag . . . . .	7
1.2.3	Common Concepts . . . . .	7
1.3	System Overview . . . . .	7
<b>2</b>	<b>Scenario Overview</b>	<b>9</b>
2.1	CMS Displaying the RSU State and Details . . . . .	9
2.2	CMS Displaying the Position of RSU . . . . .	9
2.3	CMS Sending V2X Messages via RSU . . . . .	9
2.4	CMS Displaying the Numbers of Messages Sent/Received by RSU . . . . .	9
2.5	CMS Displaying the DENMs Received by RSUs . . . . .	9
2.6	CMS Rebooting RSU . . . . .	10
<b>3</b>	<b>Feature Subsets</b>	<b>11</b>
3.1	Mandatory Subset . . . . .	11
3.2	Optional Capabilities . . . . .	11
<b>4</b>	<b>General Info Topics</b>	<b>12</b>
4.1	Online Info (Last Will) . . . . .	12
4.2	Device Status . . . . .	12
4.3	Device Info . . . . .	13
4.4	Real Time Device Position . . . . .	15
<b>5</b>	<b>Vehicle2X Messaging Topics</b>	<b>17</b>
5.1	Message Sender . . . . .	17
5.1.1	List of Installed Messages . . . . .	17
5.1.2	Message Handling Commands . . . . .	18
5.1.3	Command Response . . . . .	19
5.2	Message Statistics . . . . .	20
5.3	Reporting of Received DENMs . . . . .	21
<b>6</b>	<b>Device Management Topics</b>	<b>22</b>
6.1	Device Reboot . . . . .	22
6.2	Reboot Info . . . . .	22
	<b>Appendix D - GeoNetworking Header Json Format</b>	<b>23</b>

# List of Tables

# List of Figures

1	System overview diagram . . . . .	7
---	-----------------------------------	---

# Abbreviations and Definitions

Abbreviation	Comment
API	Application programming interface
BTP	Basic Transport Protocol
CAM	Cooperative Awareness Message
CCH	Control channel
CMS	Central management system
CPM	Collective Perception Message
DENM	Decentralized Environmental Notification Message
DENM_FLT	DENMs filtered by RSU
DSRC	Dedicated Short-Range Communications
GN	GeoNetworking
GPS	Global Positioning System
GUI	Graphical user interface
ID	Identifier
IoT	Internet of Things
IP	Internet Protocol
IVIM	Infrastructure to Vehicle Information Message
JSON	JavaScript Object Notation
L2ID	Layer 2 ID
MAC	Media Access Control
MAPEM	MAP (topology) Extended Message
MQTT	Message Queuing Telemetry Transport
PC5	Proximity-based Communication (Interface) 5
OBU	Onboard Unit
QoS	Quality of Service
RSU	Roadside Unit
RTCM	Radio Technical Commission for Maritime Services
RTCMEM	RTCM Extended Message
SAEM	Services Announcement Essential Message
SCEP	Simple Certificate Enrollment Protocol
SCHn	Service channel N (for N from 1 to 6), e.g. SCH6
SDK	Software development kit
SPATEM	Signal Phase And Timing Extended Message
SREM	Signal Request Extended Message (sometimes referred to as SRM)
SSEM	Signal Status Extended Message

Abbreviation	Comment
SSH	Secure Shell
TCP	Transmission Control Protocol
UPER	Unaligned Packed Encoding Rules
URL	Uniform Resource Locator
V2X	Vehicle-to-everything
VPN	Virtual private network
WebUI	Web User Interface
XER	XML Encoding Rules

# 1. Interface Specification

## 1.1. Introduction

This document specifies the MQTT-based management interface used for communication operations between CMS and RSU and defines its data objects.

The main purpose of this new interface is to provide a flexible, extensible and high-performing alternative to existing interfaces such as OCIT-C.

## 1.2. MQTT Protocol

MQTT is a lightweight publish/subscribe messaging protocol designed for machine-to-machine communication. It is an open standard that is used in various IoT applications and is designed to be used on top of TCP/IP. MQTT uses topics to publish and subscribe to messages, where each message is sent to a specific topic and can be consumed by clients that are subscribed to that topic.

Advantages of MQTT protocol could be summarized to the following categories:

- **Scalability** - MQTT is designed to handle large amounts of data and can handle a high number of connections, making it ideal for high-load applications
- **Low bandwidth usage** - MQTT uses a compact binary format for messages, reducing the amount of bandwidth required for communication
- **Reliability** - MQTT provides features such as quality of service (QoS) levels and message persistence, ensuring reliable communication between the CMS and RSUs
- **Security** - MQTT supports secure communication using SSL/TLS encryption, ensuring that sensitive data is protected during transmission

### 1.2.1. Topic Structure

All topics specified in this document use following structure:

`{TENANT}/devices/rsu/{RSUID}/{REST OF THE PATH}`

- **TENANT** - preparation for multi-tenant systems - currently will have always value default
- **RSUID** - the name of the RSU
- **REST OF THE PATH** - rest of the topic path

Example:

`default/devices/rsu/its-ff-2c-e6/monitoring/device/status`

### 1.2.2. Message Parameters

Every MQTT message has a QoS level and the retained flag value specified. These parameters define the level of delivery guarantee for the message, which is taken care of by broker.

The specific values we use for messages for each topic are specified at the beginning of each topic chapter.

### 1.2.2.1. Quality of Service

MQTT provides three levels of QoS:

- **0** - message is delivered at most once
- **1** - message is delivered at least once
- **2** - message is delivered exactly once

### 1.2.2.2. Retained Flag

A retained message is a normal MQTT message with the retained flag set to true. The broker stores the last retained message and the corresponding QoS for that topic. Each client that subscribes to a topic pattern that matches the topic of the retained message receives the retained message immediately after they subscribe. The broker stores only one retained message per topic.

### 1.2.3. Common Concepts

Common concepts that apply if not explicitly mentioned otherwise:

- timestamp - timestamp of the data, in seconds since January 1, 1970
- rsuld - when there is `rsuId` in the json message it is the same as `RSUID` in the MQTT topic

## 1.3. System Overview

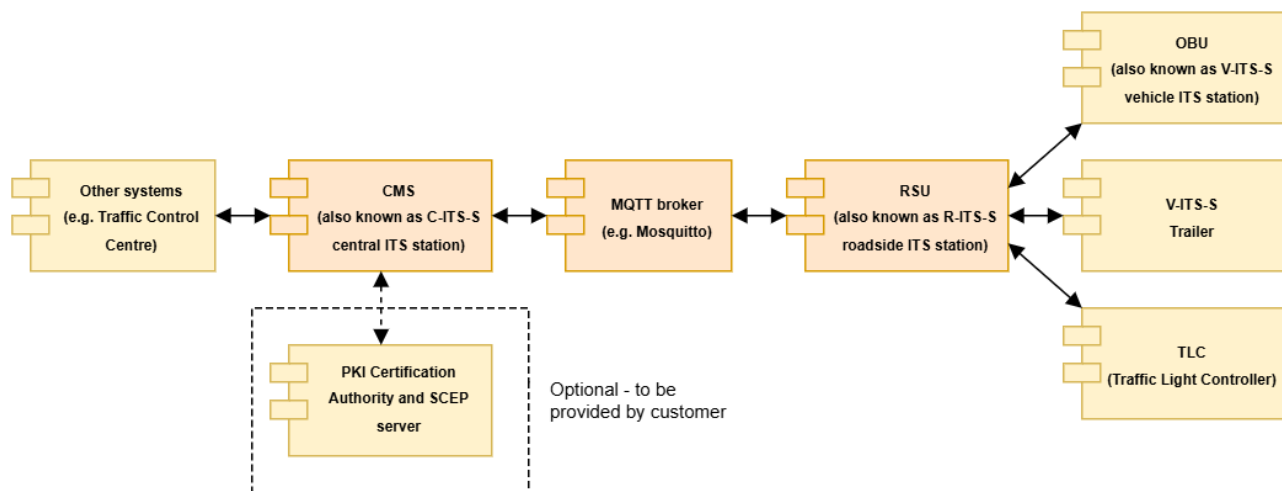


Figure 1: System overview diagram

Each RSU is assigned a unique identifier and publishes data to specific topics based on the message type. The communication between the RSU and the CMS is done via a connection to the MQTT broker. It is based on the publish-subscribe pattern. This means that both CMS and RSU can be producers and also subscribers to the MQTT broker, publishing and subscribing at the same time, depending on their use cases and message types.

The CMS is designed to process high volumes of messages. This makes it ideal for applications that require realtime processing of large amounts of data. The CMS supports secure communication with the RSUs and can be configured to use encryption for data transmission. Also it is possible to activate the SCEP certificate management for the MQTT connection. However, the SCEP certificate management is optional and is not provided. If the customer provides this functionality, security can be enhanced by relying on managed certificates.



For security reasons it is recommended to activate the TLS layer of MQTT. The certificate used for the TLS is pulled from a SCEP server. The CMS and RSU have mechanisms for the certificate expiry and renewals.

## 2. Scenario Overview

In this chapter we provide a top level overview of available scenarios from the user's point of view. The topics used, format of data exchanged and other technical details are described in linked chapters.

We describe the capabilities and intended usage based on the underlying data object design. Some of the described functionality might not be available in your current version of central system.

### 2.1. CMS Displaying the RSU State and Details

In the list of roadside units connected to the CMS the CMS can display for each RSU:

- the current online state of the RSU (chapter [4.1](#))
- the list of application services and their current state (which is the equivalent of the RSU WebUI → Status page, with some additional details, chapter [4.2](#))
- the list of software and hardware components installed (which is the equivalent of the RSU WebUI → Maintenance → System Overview page, chapter [4.3](#))

These data are automatically updated as the RSUs publish their info periodically, so the CMS can always provide a real time overview of all roadside units and report any warning/error states that require attention.

### 2.2. CMS Displaying the Position of RSU

Each RSU reports periodically its GPS position, so the CMS can visualize all RSUs on the map, and also display the movement of RSUs installed on Trailers in real time. Additionally the CMS can select the relevant traffic messages (roadwork info, in-vehicle signage, ...) for each RSU based on their position. (chapter [4.4](#))

### 2.3. CMS Sending V2X Messages via RSU

RSUs provide an interface to their Message Sender application, which is the main application used for the sending of V2X messages intended for OBUs, able to cover a wide range of use cases (equivalent to the RSU WebUI → Applications → Message Sender).

This means the CMS can install the messages the RSUs will transmit, edit the messages, define their transmission parameters and GN headers, play/pause them, delete them, and read the current list and state of messages installed on RSU. This is the interface the CMS uses for sending of DENM and IVIM messages via RSUs. (chapter [5.1](#))

### 2.4. CMS Displaying the Numbers of Messages Sent/Received by RSU

RSUs periodically report the numbers of transmitted and received V2X messages categorized per message type, which can be visualized in the CMS, used for various statistics or analyzed for the identification of possible issues. (chapter [5.2](#))

### 2.5. CMS Displaying the DENMs Received by RSUs

RSUs collect, filter and report the DENMs received via air from onboard units (functionality known as DENM Forwarding, configured via RSU WebUI → Applications → Vehicle Aggregations → DENM Forwarding).

CMS can use these DENMs from OBUs in similar manner as DENMs originating in CMS, so they can be for example installed on other RSUs to be further transmitted, or displayed on the map to identify hazardous traffic events reported by OBUs. (chapter [5.3](#))

## **2.6. CMS Rebooting RSU**

CMS can reboot RSU by this command. (chapter [6.1](#))

RSUs also share the info about the last performed reboot, so CMS can track their uptime or recognize the reboots that were not initiated by CMS. (chapter [6.2](#))

## 3. Feature Subsets

### 3.1. Mandatory Subset

It is expected that every RSU connected supports at least these:

- General topics:
  - Last Will (chapter [4.1](#))
  - Device Status (chapter [4.2](#))
    - `radioStatus` element is not mandatory and can be left as an empty array
    - `serviceStatus` element is not mandatory and can be left as an empty array
  - Device Info (chapter [4.3](#))
  - Real Time Device Position (chapter [4.4](#))
- Messaging topics:
  - Message Sender (chapter [5.1](#))
    - only the UPER encoding is mandatory
- Device management topics:
  - Device Reboot (chapter [6.1](#))

### 3.2. Optional Capabilities

RSU has to announce its optional capabilities via the Device Info message (chapter [4.3](#)). If no optional capabilities are declared, it is assumed that RSU supports only the mandatory set.

Optional topics have their capability names specified via the `Optional capability name` note in their respective chapters.

## 4. General Info Topics

### 4.1. Online Info (Last Will)

Informs whether the RSU is online or not.

- `{TENANT}/devices/rsu/{RSUID}/monitoring/device/online`
  - Direction: RSU → Broker
  - Sent as retained
  - Sent with QoS 1
  - Mandatory for RSU

```
{  
  "online": true  
}
```

On connection to the broker, RSU sets the online state to `true`. The Last Will sets it to `false`. Remember that broker might still provide additional retained messages even if RSU is offline. This topic is the source of truth for the online state of the RSU.

### 4.2. Device Status

Informs about the current state of the RSU.

- `{TENANT}/devices/rsu/{RSUID}/monitoring/device/status`
  - Direction: RSU → Broker
  - Sent as retained
  - Sent with QoS 1
  - Mandatory for RSU

```
{  
  "deviceStatus": {  
    "bluetoothFailure": false,  
    "gpsFailure": false,  
    "hsmFailure": false,  
    "rsuFailure": false,  
    "wlanFailure": false,  
    "temperature": 25.3  
  },  
  "radioStatus": [  
    {  
      "channel": 180,  
      "defaultDataRate": 6000,  
      "defaultTxPower": 23  
    },  
    {  
      "channel": 176,  
      "defaultDataRate": 6000,  
      "defaultTxPower": 23  
    }  
  ],  
}
```

```

    "serviceStatus": [
      {
        "title": "RAM Usage",
        "service": "MEMORY",
        "state": "OK",
        "details": "84.18% free",
        "section": "device"
      },
      ...
    ],
    "lastboot": 1674716672,
    "uptime": 983885,
    "timestamp": 1675700557
  }

```

- **deviceStatus** - basic device status information about the state of hardware modules and housing temperature
- **radioStatus** - list of V2X channels that are currently tuned
  - **channel** - channel number
  - **defaultDataRate** - default data rate (in kbps)
  - **defaultTxPower** - default transmit power (in dBm)
- **serviceStatus** - list of services and their status details (matching the status page on the WebUI)
  - **title** - human-readable title of the service
  - **service** - internal name of the service
  - **state** - out of enum [OK / WARNING / CRITICAL / UNKNOWN]
  - **details** - human-readable description of the state (never rely on the format of this string, it is not considered stable API and it can change without notice)
  - **section** - can be used to group the services into logical groups
- **lastboot** - UNIX timestamp of the last boot
- **uptime** - uptime of the device in seconds
- **timestamp** - timestamp of the message

This device status is being published periodically every 60 seconds.

### 4.3. Device Info

Informs about the hardware/software version of the RSU.

- `{TENANT}/devices/rsu/{RSUID}/monitoring/device/info`
  - Direction: RSU → Broker
  - Sent as retained
  - Sent with QoS 1
  - Mandatory for RSU



```
{
  "applications": [
    {
      "build": "20240616_15:45",
      "isMounted": true,
      "name": "eu",
      "requiredCoreVersion": "02.03.08",
      "requiredMode": "etsi",
      "state": "DACHv02.03.08:20240616_15:45:SDKv02.03.08",
      "version": "DACHv02.03.08"
    },
    ...
  ],
  "customData": {
    "<custom key string>": "<custom value string>",
    ...
  },
  "network": {
    "interfaces": {
      "eth0": ["192.168.12.1/24", "172.168.1.1/24"],
      "eth1": ["192.168.12.1/24"]
    },
    "routes": [
      {
        "dest": "default",
        "gateway": "192.168.22.1",
        "interface": "eth0",
        "metric": 0
      },
      ...
    ]
  },
  "stationId": 123456,
  "basesystem": "rsu2x ESCoS Poky distribution v4.0.8",
  "bootloader": "iMX8 BOOT",
  "firmware": "02.03.08-240616-2271478",
  "hardware": "RSU2X",
  "hardwareVersion": "1",
  "protocolVersion": "02.04.11",
  "optionalCapabilities": [
    "messageStatistics", "denmForwarding", ...
  ],
  "latitude": 49.15948,
  "longitude": 16.58756,
  "manufacturer": "YunexTraffic",
  "mode": "etsi",
  "sdk": "02.03.08",
  "serialNumber": "04E5481222E8",
  "timestamp": 1718794999
}
```

- applications - list of applications installed on the RSU
  - build - build date
  - isMounted - true if the application is mounted, false if not

- `name` - application name
- `requiredCoreVersion` - required SDK version in the core firmware (see the field `sdk`)
- `requiredMode` - required RSU mode (see the field `mode`)
- `state` - detailed state of the application
- `version` - version of the application
- `customData` - key value store for additional custom data defined by user (OPTIONAL)
- `network` (OPTIONAL)
  - `interfaces` - list of interfaces with IP addresses
  - `routes` - list of routes
- `stationId` - RSU stationID
- `basesystem` - base-system version
- `bootloader` - bootloader version
- `firmware` - version of the core firmware
- `hardware` - hardware type of the unit, out of enum [`RSU1` / `RSU2X`]
- `hardwareVersion` - hardware revision
- `protocolVersion` - version of this MQTT protocol document the RSU conforms to
- `optionalCapabilities` - list of the optional features the RSU supports (chapter 3.2)
- `latitude` - latitude of the RSU in degrees
- `longitude` - longitude of the RSU in degrees
- `manufacturer` - manufacturer of the device
- `mode` - radio mode, out of enum [`etsi` / `wave`]
- `serialNumber` - unique serial number
- `sdk` - version of the SDK in core firmware
- `timestamp` - timestamp of the generation of this message

This device info is being published periodically every 30 minutes.

#### 4.4. Real Time Device Position

Informs about the position of the RSU.

- `{TENANT}/devices/rsu/{RSUID}/monitoring/gps/fix`
  - Direction: RSU → Broker
  - Sent as retained
  - Sent with QoS 1

■ Mandatory for RSU

```
{  
  "alt": 244,  
  "heading": 167.9251,  
  "lat": 49.1847362,  
  "lon": 16.6260559,  
  "speed": 0.015,  
  "timestamp": 1718796952  
}
```

- alt - current altitude in meters
- heading - current heading angle in degrees, clockwise from the north (north = 0, east = 90, ...)
- lat - current latitude in degrees
- lon - current longitude in degrees
- speed - current speed in m/s
- timestamp - timestamp of the generation of this message

This GPS info is being published periodically every second, always containing the actual RSU position.

## 5. Vehicle2X Messaging Topics

### 5.1. Message Sender

Following chapters define the interface for the Message Sender, which is an RSU application used for the periodical transmission of V2X messages.

#### 5.1.1. List of Installed Messages

- {TENANT}/devices/rsu/{RSUID}/msgsender/messages

- Direction: RSU → Broker
- Sent as retained
- Sent with QoS 1
- Mandatory for RSU

```
{
  "messages": [
    {
      "msgId": "test message",
      "active": true,
      "interval": 1000,
      "duration": -1,
      "payloadType": "XER",
      "payload": "PERFTk0...U5NPg==",
      "btp": {
        "type": "DENM"
      },
      "gn": {
        "chan": "CCH",
        "scf": false,
        "offload": false,
        "tc": 1,
        "transport": "SHB",
        "security": {
          "sign": false
        }
      },
      "autofill": [
        "time",
        "station"
      ],
      ...
    ],
    "timestamp": 1718798888
  }
}
```

This topic contains the list of messages installed in the Message Sender. For the description of objects see the next chapter.

This list is automatically published after any Message Sender change, so it always provides the actual state.

### 5.1.2. Message Handling Commands

■ {TENANT}/devices/rsu/{RSUID}/msgsender/command

- Direction: CMS → Broker → RSU
- Sent as not retained
- Sent with QoS 1
- Mandatory for RSU

```
{
  "msgId": "test message",
  "command": "install",
  "active": true,
  "interval": 1000,
  "duration": 720,
  "payloadType": "UPER",
  "payload": "d36c000023cf00000004a88960800000045ae00000005a4b5a4d9f86b49ce38d86fd64cc19253308",
  "btp": {
    "type": "DENM"
  },
  "gn": {
    "chan": "CCH",
    "scf": false,
    "offload": false,
    "tc": 1,
    "transport": "GBC",
    "dest": {
      "area": {
        "shape": "circle",
        "center": {
          "lon": 166260271,
          "lat": 491846912
        },
        "distA": 400
      }
    },
    "security": {
      "sign": false
    }
  },
  "autofill": [
    "time",
    "station"
  ]
}
```

- msgId - name of the message, serves as unique key to identify messages and to pair commands to replies
- command - specifies the operation, can contain one of following values:
  - install - installs a new message based on the provided data (if the message with specified msgId already exists it will be replaced)
  - delete - deletes the message with specified msgId (other fields can be omitted)
  - play - starts sending the message with specified msgId (other fields can be omitted)

- `pause` - stops sending the message with specified `msgId` (other fields can be omitted)
- `active` - if `true` the message will be active, if `false` it will be paused (OPTIONAL, default = `false`)
- `interval` - interval of the message sending in milliseconds (100 - 5000)
- `duration` - defines for how long the RSU will send the message in seconds (-1 = forever)
- `payloadType` - specifies the payload type, can contain one of following values:
  - `XER` - for XER-encoded payloads
  - `UPER` - for UPER-encoded payloads
  - `RAW_UPER` - for custom non-Etsi messages (RSU will not check the message validity, nor sign it), RSU still generates GN,BTP headers
  - `RAW_DATA` - for data already containing all headers (BTP/GN/security) together with the ITS message
- `payload` - the actual payload data the RSU will transmit, they must be:
  - HEX-encoded if the `payloadType` is `UPER`, `RAW_UPER` or `RAW_DATA`
  - Base64-encoded if the `payloadType` is `XER`
- `btp` - definition of the Basic Transport Protocol header (OPTIONAL)
  - `type` - out of enum [`CAM` / `DENM` / `SPATEM` / `IVIM` / `SREM` / `SSEM` / `CPM` / `RTCMEM` / `SAEM`]
- `gn` - definition of the GeoNetworking header, see [Appendix D](#) for details (OPTIONAL)
- `autofill` - specifies which payload fields will RSU adjust (OPTIONAL, default = [`"time"`, `"station"`])
  - `time` - RSU will automatically adjust time related fields
  - `position` - RSU will automatically adjust position related fields
  - `station` - RSU will automatically adjust station id/type related fields

If `btp` or `gn` are omitted, RSU will try to use the best possible header content based on the `payload` content.

It is possible to send messages also on non-standard BTP ports by using  
`"btp":{"type":"OTHER","dest":7000}`

Once the `duration` elapses, the message is automatically deleted.

`RAW_DATA` cannot use `btp` (because the BTP header is already part of their payload) and `autofill` (because the payload isn't parsed or adjusted in any way). And from the `gn` they can utilize only parameters `chan` and `tc`.

### 5.1.3. Command Response

- `{TENANT}/devices/rsu/{RSUID}/msgsender/command/result`
  - Direction: RSU → Broker
  - Sent as not retained



- Sent with QoS 1
- Mandatory for RSU

```
{
  "msgId": "test message",
  "command": "install",
  "success": true,
  "details": "Message was installed."
}
```

- msgId - name of the message this response refers to
- command - command type the RSU is responding to
- success - true if the command was successfully executed by the RSU, false otherwise
- detail - optional text details (never rely on the format of this string, it is not considered stable API and it can change without notice)

## 5.2. Message Statistics

Informs about the numbers of sent/received V2X messages.

- {TENANT}/devices/rsu/{RSUID}/monitoring/messages/last60s
  - Direction: RSU → Broker
  - Sent as not retained
  - Sent with QoS 1
  - Optional capability name: messageStatistics

```
{
  "CAM": {
    "active": 0,
    "rx": 60,
    "tx": 0
  },
  "DENM": {
    "active": 1,
    "rx": 0,
    "tx": 31
  },
  "IVIM": {
    "active": 0,
    "rx": 0,
    "tx": 0
  },
  "timestamp": 1675701692
}
```

- active - number of active messages of this type installed in the Message Sender
- rx - number of messages of this type that were received by the RSU during the last 60 seconds
- tx - number of messages of this type that were sent by the RSU during the last 60 seconds

These statistics are being published periodically every 60 seconds.

### 5.3. Reporting of Received DENMs

Reports the DENMs received by the RSU to CMS.

- {TENANT}/devices/rsu/{RSUID}/warning/denm
  - Direction: RSU → Broker
  - Sent as not retained
  - Sent with QoS 1
  - Optional capability name: denmForwarding

```
{  
  "rsuId": "RSU81",  
  "originatingStationId": 19806,  
  "sequenceNumber": 6,  
  "stationId": 16724,  
  "timestamp": 1718885018,  
  "uper": "02010000007...c2e6"  
}
```

- rsuId - RSUID
- originatingStationId - value of <originatingStationId> from the received DENM
- sequenceNumber - value of <sequenceNumber> from the received DENM
- stationId - value of <stationId> from the received DENM
- timestamp - timestamp of when the published MQTT message was created
- uper - whole DENM message in the HEX-encoded UPER format

In order to access these data, the DENM Forwarding application must be enabled and configured on the RSU. (See the WebUI → Applications → Vehicle Aggregations → DENM Forwarding).

This topic then serves as an interface for this application, so all the application settings (holdoff interval, security profile validation, ...) will apply also here.

RSU will skip DENMs with <stationType> set to 15 which stands for Roadside Unit.

## 6. Device Management Topics

### 6.1. Device Reboot

Instructs RSU to perform a reboot.

- `{TENANT}/devices/rsu/{RSUID}/mgmt/reboot`
  - Direction: CMS → Broker → RSU
  - Sent as not retained
  - Sent with QoS 2
  - Mandatory for RSU

Upon receiving *any* message on this topic RSU immediately performs a reboot.

### 6.2. Reboot Info

Informs about the last performed reboot.

- `{TENANT}/devices/rsu/{RSUID}/monitoring/reboots`
  - Direction: RSU → Broker
  - Sent as retained
  - Sent with QoS 1
  - Optional capability name: `rebootInfo`

```
{  
  "rsuId": "RSU81",  
  "timestamp": 1675543532,  
  "reason": "UNKNOWN"  
}
```

- `rsuId` - RSUID
- `timestamp` - timestamp of the kernel startup after the reboot
- `reason` - currently always `UNKNOWN`, reserved for future use

RSU publishes this message once after it boots up.

This info is sent after every reboot, not only those triggered by MQTT.

## Appendix D - GeoNetworking Header Json Format

*Description will be available in future versions. For now please refer to the RSU Xfer Gateway specification which uses the same JSON format for GeoNetworking header settings.*

## Yunex GmbH

Otto-Hahn-Ring 6  
81739 Munich  
Germany

Tel: +49 (0) 89 7805 0

Email: [contact@yunextraffic.com](mailto:contact@yunextraffic.com)

All hardware and software names used are brand names  
and/or trademarks of their respective holders.

© 2026 - Yunex Traffic.

Right of modifications reserved.

[Imprint](#)

[Data Privacy Notice](#)

Subject to changes and errors. The information given in this document only contains general descriptions and/or performance features which may not always specifically reflect those described, or which may undergo modification in the course of further development of the products. The requested performance features are binding only when they are expressly agreed upon in the concluded contract.

