LTE MBMS Gateway

Version: 2025-05-21

Table of Contents

1 Introduction

LTEMBMSGW is a LTE MBMS Gateway. It can easily be used with the Amarisoft LTE eNodeB to build an LTE MBMS test system.

2 Features

- User configurable list of service and multicast components.
- M2AP protocol support.
- Generate one stream per service over the M1 interface (GTP + SYNC protocols).
- Built-in test RTP packet generator.
- Remote API using WebSocket.

3 Requirements

3.1 Hardware requirements

- LTEMBMSGW can run on the same PC as the Amarisoft eNodeB if a simple and compact solution is needed. Otherwise, any reasonnably recent PC with at least one Gigabit Ethernet port is acceptable.

3.2 Software requirements

- A 64 bit Linux distribution. Fedora 39 is the officially supported distribution.
  The following distributions are known as compatible:
  - Fedora 22 to 39
  - Cent OS 7
  - Ubuntu 14 to 22

  Your system requires at least GLIBC 2.17.

4 Installation

We assume that the Fedora distribution is running and that the network access thru the Gigabit Ethernet port is correctly configured.

LTEMBMSGW can be run directly from the directory when it was unpacked. No need for explicit installation.

4.1 Linux setup

4.1.1 Packages

LTEMBMSGW uses the SCTP protocol for which the necessary packages are not usually installed. In order to install them, do as root user:

- Fedora

  ```
  dnf install lksctp-tools kernel-modules-extra
  ```

- Ubuntu

  ```
  sudo apt-get install lksctp-tools linux-image-extra-3.13.0-24-generic
  ```

  Note that linux-image-extra package name may differ depending on your kernel version.

To verify that SCTP kernel module is running, do as root user:

```
checksctp
```

If it reports that the protocol is not supported,

- check if you have a /etc/modprobe.d/sctp-blacklist.conf file
- edit it to comment the 'blacklist sctp' line

Then reboot the PC in case the Linux kernel was upgraded too.

4.1.2 OpenSSL

LTEMBMSGW has been compiled against openssl version 1.1.1w.
If your system does not have compatible version installed you may have this error message at startup:

```
error while loading shared libraries: libssl.so.1.1: cannot open shared object file: No such file or directory
```

To overcome this problem, you may:

- Copy libssl.so.1.1 and libcrypto.so.1.1 from `libs` subdirectory of your release tarball.
  If you have installed software with automatic install script, this should have been done automatically.
- Compile and install proper openssl version yourself

In case of persisting issue, raise a ticket from our support site at `https://support.amarisoft.com/` with the information provided by below commands executed in LTEMBMSGW directory:

```
uname -a
ls -l
ldd ./ltembmsgw
openssl version
```

4.2 License key installation

LTEMBMSGW needs a `LTEMME` license key to run. Please refer to the `ltemme` documentation.

4.3 Initial testing

- Start the eNodeB with the example MBMS configuration:

  ```
  ./lteenb config/enb-mbms.cfg
  ```

- Start the program as root with the default configuration. This configuration contains several MBMS services. For each service, RTP dummy streams are generated:

  ```
  ./ltembmsgw config/mbmsgw.cfg
  ```

- Verify that the MBMS GTP data is correctly received by the eNodeB with the `mbms` command in the eNodeB monitor. You should see a non zero bitrate for each service and zero packet error.
- Verify that you can receive the corresponding services on your LTE device. The exact setup depend on your device.

When this basic test work, you can customize the eNodeB and MBMS Gateway configuration to use your own generated multicast services.

5 Configuration reference

5.1 Configuration file syntax

The main configuration file uses a syntax very similar to the Javascript Object Notation (JSON) with an extension to support complex numbers and a few mathematical operations. The supported types are:

- Numbers (64 bit floating point). Notation: `13.4`
- Complex numbers. Notation: `1.2+3*I`
- Strings. Notation: `"string"`
- Booleans. Notation: `true` or `false`.
- Objects. Notation: `{ field1: value1, field2: value2, .... }`
- Arrays. Notation: `[ value1, value2, .... ]`

The basic operations `+`, `-`, `*` and `/` are supported with numbers and complex numbers.

The numbers `0` and `1` are accepted as synonyms for the boolean values `false` and `true`.

5.2 Properties

`log_filename`
: String. Set the log filename. If no leading `/`, it is relative to the configuration file path. See [Log file format], page 19.

`log_options`
: String. Set the logging options as a comma separated list of assignments.

  - *layer*.level=*verbosity*. For each layer, the log verbosity can be set to `none`, `error`, `info` or `debug`. In debug level, the content of the transmitted data is logged.
  - *layer*.max_size=*n*. When dumping data content, at most `n` bytes are shown in hexa. For ASN.1, NAS or Diameter content, show the full content of the message if `n > 0`.
  - *layer*.payload=[0|1]. Dump ASN.1, NAS, SGsAP or Diameter payload in hexadecimal.
  - *layer*.key=[0|1]. Dump security keys (NAS and RRC layers).
  - *layer*.crypto=[0|1]. Dump plain and ciphered data (NAS and PCDP layers).
  - time=[sec|short|full]. Display the time as seconds, time only or full date and time (default = time only).
  - time.us=[0|1]. Dump time with microseconds precision.
  - file=cut. Close current file log and open a new one.
  - file.rotate=now. Move and rename to the same directory or to the directory pointed by `file.path` and open a new log file (Headers are kept).
  - file.rotate=*size*. Every time log file size reaches *size* bytes, move and rename to the same directory or to the directory pointed by `file.path`, and open a new log file (Headers are kept).
    Size is an integer and can be followed by K, M or G.
  - file.rotate=#*count*. Everytime number of logs in log file reaches *count*, move and rename to the same directory or to the directory pointed by `file.path`, and open a new log file (Headers are kept).
    Size is an integer and can be followed by K, M or G.

- file.path=*path*. When log rotation is enabled (`file.rotate` set), rename and move current log to this path instead of initial log path.
- append=[0|1]. (default=0). If 0, truncate the log file when opening it. Otherwise, append to it.

Available layers are: `gtpu`, `m2ap`

`log_sync` Optional boolean (default = false). If true, logs will be synchronously dumped to file.
Warning, this may lead to performances decrease.

`com_addr` Optional string. Address of the WebSocket server remote API. See [Remote API], page 12.
If set, the WebSocket server for remote API will be enabled and bound to this address.
Default port is 9004.
Setting IP address to [::] will make remote API reachable through all network interfaces.

`com_name` Optional string. Sets server name. MBMSGW by default

`com_ssl_certificate`
Optional string. If set, forces SSL for WebSockets. Defines CA certificate filename.

`com_ssl_key`
Optional string. Mandatory if *com_ssl_certificate* is set. Defines CA private key filename.

`com_ssl_peer_verify`
Optional boolean (default is false). If *true*, server will check client certificate.

`com_ssl_ca`
Optional string. Set CA certificate. In case of peer verification with self signed certificate, you should use the client certificate.

`com_log_lock`
Optional boolean (default is false). If *true*, logs configuration can't be changed via `config_set` remote API.

`com_log_us`
Optional boolean (default is false). If *true*, logs sent by `log_get` remote API response will have a `timestamp_us` parameters instead of `timestamp`

`com_auth` Optional object. If set, remote API access will require authentication.
Authentication mechanism is describe in [Remote API Startup], page 14, section.

`passfile` Optional string. Defines filename where password is stored (plaintext). If not set, `password` must be set

`password` Optional string. Defines password.
If not set, `passfile` must be set.

`unsecure` Optional boolean (default false). If set, allow password to be sent plaintext.
NB: you should set it to true if you access it from a Web Browser (Ex: Amarisoft GUI) without SSL (https) as your Web Browser may prevent secure access to work.

`com_log_count`
: Optional number (Default = 8192). Defines number of logs to keep in memory before dropping them.
Must be between 4096 and 2097152).

`sim_events`
: Array of object. Each element defines a remote API request ([Remote API], page 12) except that `message` field is replaced by `event`.

`sim_events_loop_count`
: If set, will define `loop_count` for each event of `sim_events`, See [loop_count], page 13.

`sim_events_loop_delay`
: If set, will define `loop_delay` for each event of `sim_events`, See [loop_delay], page 13.

`license_server`
: Configuration of the Amarisoft license server to use.
Object with following properties:

  `server_addr`
  : String. IP address of the license server.

  `name`
  : Optional string. Text to be displayed inside server monitor or remote API.

  `tag`
  : Optional string. If set, server will only allow license with same tag.

  Example:

```
license_server: {
    server_addr: "192.168.0.20",
    name: "My license"
}
```

`gtp_bind_addr`
: String. Set source IP address (and an optional port) of the GTP-U packets. The default value is `"0.0.0.0:2152"`.

  Syntax:

  - `"1.2.3.4"` (use default port)
  - `"1.2.3.4:5678"` (use explicit port)
  - `"2001:db8:0:85a3::ac1f:8001"` (IPv6 address and default port)
  - `"[2001:db8:0:85a3::ac1f:8001]:5678"` (IPv6 address and explicit port)

`m2ap_bind_addr`
: Optional string. IP address and optional port on which the M2AP SCTP connection is bound. The default port is 36443.

`mce_id`
: Integer. Range: 0 to 65535. Global MCE Identifier used in M2 signaling.

`enb_time_offset`
: Optional integer (default = 0). Offset in ms applied to the MBMSGW International Atomic Time (TAI) so as to generate a time that should match the eNB RF time. The current value can be retrieved by typing the `time` monitor command in eNB or MBMSGW prompt. This is used to synchronize the two components so as to have meaningful timestamps in the SYNC packets (indicating the start of the MCH Scheduling Periods).

Note: the MBMSGW derives the TAI from the UTC OS clock and the `right/UTC` OS time zone.

`time_offset`

Integer. Default time offset in ms added to all the SYNC timestamps. Can be overridden by the `time_offset` property of each service. It is recommended to set it to at least 2 MCH Scheduling Period to avoid having the eNB dropping SYNC packets due to a timestamp equal to the current MCH Scheduling Period.

Note: the MBMS Gateway uses the system real time clock as clock source. If synchronous transmission is needed, it should be synchronized to the eNodeB RF time.

`services`

Array of objects. Contain the definition of each service.

Property of each service:

`tmgi` Object. Service identifier (only used for error reporting). Contain the following fields:

`plmn` String (5 or 6 digits). PLMN identity of the service.

`service_id`
Integer. 24 bit service identity.

`service_area_id`
Integer. Range: 0 to 65535. MBMS service area identifier for this service.

`session_id`
Optional integer. Range: 0 to 255. MBMS session identifier for this service.

`gtp_addr` String. IP address (and optional port) to which the GTP packets are sent. It is normally a multicast address. Several services can share the same IP address if they have a different TEID.

`gtp_teid` 32 bit integer. GTP TEID on which the GTP packets are sent.

`autostart`
Optional boolean (default = true). Indicates if service is automatically started when the eNB connects to the MBMS Gateway or if it should be manually launched with the service_start command.

`scheduling_period`
Range: from 4 to 1024. Must be a power of two. Duration of the scheduling period in 10 ms units. Must match the corresponding MCH scheduling period configured in the eNodeB.

`time_offset`
Optional integer. Time offset in ms added to the SYNC timestamps. If not provided, the default time offset is used.

`forward_mode`
Optional boolean (default = false). If set, gateway won't add sync headers and only forward packet to the eNB.

`tos` Optional integer (default = 0). IPv4 header TOS field (6 bits DSCP + 2 bits ECN).

`traffic_class`
: Optional integer (default = 0). IPv6 header traffic class field (6 bits DSCP + 2 bits ECN).

`ttl`
: Optional integer (default = 64). IP header TTL field.

`components`
: Array of object. A service contains several components. Each component is the data coming from a given IP address (usually multicast).

  Component properties:

  `ip_addr`
  : String. Destination IPv4/v6 address and port for the component.

  `if_addr`
  : Optional string (default = `"0.0.0.0"`). IP address of the network interface for the multicast join.

  `sim`
  : Optional boolean (default = false). If true, RTP packets coming from `ip_addr` are generated using a RTP payload of `rtp_payload_len` bytes and a bitrate of `bitrate`.

  `rtp_payload_len`
  : Optional integer. Only meaningful if sim = true. RTP payload length in bytes (default = 1460 for IPv4 of 1440 for IPv6).

  `bitrate`
  : Optional integer. Only meaningful if sim = true. Bitrate in bit/s of the generated RTP stream. The bitrate includes the size of the IP, UDP and RTP headers.

`area_info_list`
: Array of object. Each object defines the parameters of one MBSFN area:

  `area_id`
  : Range: 0 to 255. Area identifier.

  `non_mbsfn_region_length`
  : Enumeration: 1, 2. Number of CCH symbols. For 1.4 MHz downlink, only `2` is allowed.

  `mcch_config`
  : Object. MCCH configuration:

    `mcch_repetition_period`
    : Range: 32 to 256, power of two. MCCH repetition period (in 10 ms frames).

    `mcch_offset`
    : Range: 0 to 10. MCCH offset.

    `mcch_modification_period`
    : Enumeration: 512, 1024. (in 10 ms frames).

    `mcch_sf_alloc`
    : Bit string. Length = 6 (1 frame). In FDD, the bits correspond to subframes 1, 2, 3, 6, 7, 8. In TDD, the bits correspond to subframes 3, 4, 7, 8, 9.

    `signalling_mcs`
    : Enumeration: 2, 7, 13, 19. MCS for MCCH and MCHSI transmission. MCCH and MCHSI are critical to decode the MBMS data (MTCH), so their MCS should be lower than the one of the data.

`mbsfn_area_configuration`
: Object. MBSFN area configuration. Most of the content of this object is transmitted in the MCCH.

  `common_sf_alloc`
  : Array of object. Defines the subframes dedicated to this MBSFN area. Each object has the following fields:

    `radio_frame_allocation_period`
    : Range: 1 to 32, power of two. Allocation period (in 10 ms frames).

    `radio_frame_allocation_offset`
    : Range: 0 to 7. offset in the allocation period (in 10 ms frames).

    `subframe_allocation`
    : Bit string. Length = 6 (1 frame) or 24 (4 frames). In FDD, the bits correspond to subframes 1, 2, 3, 6, 7, 8. In TDD, the bits correspond to subframes 3, 4, 7, 8, 9.

  `common_sf_alloc_period`
  : Range: 4 to 256, power of two. Common subframe allocation period (in 10 ms frames). The PMCH are allocated consecutively during this period.

  `pmch_info_list`
  : Array of objects. List of PMCH. Each PMCH has the following properties:

    `pmch_config`
    : Object. PMCH physical parameters.

      `sf_alloc_count`
      : Integer >= 1. Number of subframes allocated to this PMCH per common period.

      `data_mcs` Range: 0 to 28. MCS used for the MBMS data (MTCH).

      `data_mcs2`
      : Optional integer. Range: 0 to 27. If provided, `data_mcs` is ignored and an alternate MCS table is used to allow 256QAM MBMS. Note: 256QAM MBMS is an optional release 12 feature, so not all UEs can receive a PMCH using `data_mcs2`.

      `mch_scheduling_period`
      : Range: 4 to 1024, power of two. Scheduling period (in 10 ms frames) for the MCH. MCHSI is transmitted with this periodicity. Must be >= common_sf_alloc_period. For

the first PMCH, must be <= mcch_repetition_period. Note: only release 12 UEs support the value 4, so the effective range to support all UEs is 8 to 1024.

`mbms_session_info_list`
: Array of objects. List of sessions in this PMCH. Each session has the following properties:

  `tmgi`
  : Object. Temporary Mobile Group Identity.

    `plmn`
    : String (5 or 6 digits). PLMN identity.

    `service_id`
    : 24 bit integer. Service identity.

  `logical_channel_identity`
  : Range: 0 to 28. MAC logical channel identity. Must be different for each session in the PMCH. 0 is reserved for the MCCH in the first PMCH.

6 Remote API

You can access LTEMBMSGW via a remote API.
Protocol used is WebSocket as defined in RFC 6455 (`https://tools.ietf.org/html/rfc6455`).

Note that Origin header is mandatory for the server to accept connections.
This behavior is determined by the use of `nopoll` library.
Any value will be accepted.

To learn how to use it, you can refer to our the following tutorial (`https://tech-academy.amarisoft.com/RemoteAPI.html`).

6.1 Messages

Messages exchanged between client and LTEMBMSGW server are in strict JSON format.

Each message is represented by an object. Multiple message can be sent to server using an array of message objects.

Time and delay values are floating number in seconds.

There are 3 types of messages:

- Request

  Message sent by client.
  Common definition:

  `message` String. Represent type of message. This parameter is mandatory and depending on its value, other parameters will apply.

  `message_id`
  Optional any type. If set, response sent by the server to this message will have same message_id. This is used to identify response as WebSocket does not provide such a concept.

  `start_time`
  Optional float. Represent the delay before executing the message.
  If not set, the message is executed when received.

  `absolute_time`
  Optional boolean (default = false). If set, `start_time` is interpreted as absolute.
  You can get current clock of system using `time` member of any response.

  `standalone`
  Optional boolean (default = false). If set, message will survive WebSocket disconnection, else, if socket is disconnected before end of processing, the message will be cancelled.

  `loop_count`
  Optional integer (default = 0, max = 1000000). If set, message will be repeated `loop_count` time(s) after `loop_delay` (From message beginning of event).
  Response will have a `loop_index` to indicate iteration number.

`loop_delay`
: Optional number (min = 0.1, max = 86400). Delay in seconds to repeat message from its `start_time`. Mandatory when `loop_count` is set > 0.

- Response

  Message sent by server after any request message as been processed.
  Common definition:

  `message` String. Same as request.

  `message_id`
  : Optional any type. Same as in request.

  `time` Number representing time in seconds since start of the process.
  Usefull to send command with absolute time.

  `utc` Number representing UTC seconds.

- Events

  Message sent by server on its own initiative.
  Common definition:

  `message` String. Event name.

  `time` Number representing time in seconds.
  Usefull to send command with absolute time.

6.2 Startup

When WebSocket connections is setup, LTEMBMSGW will send a first message with name set to `com_name` and type set to MBMSGW.

If authentication is not set, message will be `ready`:

```
{
    "message": "ready",
    "type": "MBMSGW",
    "name": <com_name>,
    "version": <software version>,
    "product": <Amarisoft product name (optional)>
}
```

If authentication is set, message will be `authenticate` :

```
{
    "message": "authenticate",
    "type": "MBMSGW",
    "name": <com_name>,
    "challenge": <random challenge>
}
```

To authenticate, the client must answer with a `authenticate` message and a `res` parameter where:

```
res = HMAC-SHA256( "<type>:<password>:<name>", "<challenge>" )
```

`res` is a string and HMAC-SHA256 refers to the standard algorithm (`https://en.wikipedia.org/wiki/HMAC`)

If the authentication succeeds, the response will have a `ready` field set to `true`.

```
{
    "message": "authenticate",
```

```
    "message_id": <message id>,
    "ready": true
}
```

If authentication fails, the response will have an `error` field and will provide a new challenge.

```
{
    "message": "authenticate",
    "message_id": <message id>,
    "error": <error message>,
    "type": "MBMSGW",
    "name: <name>,
    "challenge": <new random challenge>
}
```

If any other message is sent before authentication succeeds, the error `"Authentication not done"` will be sent as a response.

6.3 Errors

If a message produces an error, response will have an error string field representing the error.

6.4 Sample nodejs program

You will find in this documentation a sample program: `ws.js`.
It is located in `doc` subdirectory.
This is a nodejs program that allow to send message to LTEMBMSGW.
It requires nodejs to be installed:

```
dnf install nodejs npm
npm install nodejs-websocket
```

Use relevant package manager instead of NPM depending on your Linux distribution.

Then simply start it with server name and message you want to send:

```
./ws.js 127.0.0.1:9004 '{"message": "config_get"}'
```

6.5 Common messages

`config_get`
: Retrieve current config.

  Response definition:

  `type` Always "MBMSGW"

  `name` String representing server name.

  `logs` Object representing log configuration.
  With following elements:

  `layers` Object. Each member of the object represent a log layer configuration:

  `layer name`
  : Object. The member name represent log layer name and parameters are:

    `level` See [log_options], page 6,

`max_size` See [log_options], page 6,

`key` See [log_options], page 6,

`crypto` See [log_options], page 6,

`payload` See [log_options], page 6,

`count` Number. Number of bufferizer logs.

`rotate` Optional number. Max log file size before rotation.

`rotate_count`
Optional number. Max log count before rotation.

`path` Optional string. Log rotation path.

`bcch` Boolean. True if BCCH dump is enabled (eNB only).

`mib` Boolean. True if MIB dump is enabled (eNB only).

`locked` Optional boolean. If `true`, logs configuration can't be changed with `config_set` API.

`config_set`
Change current config.
Each member is optional.
Message definition:

`logs` Optional object. Represent logs configuration. Same structure as config_get (See [config_get logs member], page 15).
All elements are optional.
Layer name can be set to `all` to set same configuration for all layers.
If set and logs are locked, response will have `logs` property set to `locked`.

`log_get` Get logs.
This API has a per connection behavior. This means that the response will depend on previous calls to this API within the same WebSocket connection.
In practice, logs that have been provided in a response won't be part of subsequent request unless connection is reestablished. To keep on receiving logs, client should send a new `log_get` request as soon as the previous response has been received.
If a request is sent before previous request has been replied, previous request will be replied right now without considering specific min/max/timeout conditions.

Message definition:

`min` Optional number (default = 1). Minimum amount of logs to retrieve. Response won't be sent until this limit is reached (Unless timeout occurs).

`max` Optional number (default = 4096). Maximum logs sent in a response.

`timeout` Optional number (default = 1). If at least 1 log is available and no more logs have been generated for this time, response will be sent.

`allow_empty`
Optional boolean (default = false). If set, response will be sent after timeout, event if no logs are available.

`rnti` Optional number. If set, send only logs matching rnti.

`ue_id` Optional number. If set, send only logs with matching ue_id.

`layers` Optional Object. Each member name represents a log layer and values must be string representing maximum level. See [log_options], page 6. If *layers* is not set, all layers level will be set to *debug*, else it will be set to *none*.
Note also the logs is also limited by general log level. See [log_options], page 6.

`short` Optional boolean (default = false). If set, only first line of logs will be dumped.

`headers` Optional boolean. If set, send log file headers.

`start_timestamp`
Optional number. Is set, filter logs older than this value in milliseconds.

`end_timestamp`
Optional number. Is set, filter logs more recent than this value in milliseconds.

`max_size` Optional number (default = 1048576, i.e. 1MB). Maximum size in bytes of the generated JSON message. If the response exceeds this size, the sending of logs will be forced independently from other parameters.

Response definition:

`logs` Array. List of logs. Each item is a an object with following members:

- `data` Array. Each item is a string representing a line of log.
- `timestamp`
  Number. Milliseconds since January 1st 1970. Not present if `com_log_us` is set in configuration.
- `timestamp_us`
  Number. Microseconds since January 1st 1970. Only present if `com_log_us` is set in configuration.
- `layer` String. Log layer.
- `level` String. Log level: *error*, *warn*, *info* or *debug*.
- `dir` Optional string. Log direction: *UL*, *DL*, *FROM* or *TO*.
- `ue_id` Optional number. UE_ID.
- `cell` Optional number (only for PHY layer logs). Cell ID.
- `rnti` Optional number (only for PHY layer logs). RNTI.
- `frame` Optional number (only for PHY layer logs). Frame number (Subframe is decimal part).
- `channel` Optional string (only for PHY layer logs). Channel name.
- `src` String. Server name.
- `idx` Integer. Log index.
- `headers` Optional array. Array of strings.

`discontinuity`
: Optional number. If set, this means some logs have been discarded due to log buffer overflow.

`microseconds`
: Optional boolean. Present and set to true if `com_log_us` is set in configuration file.

`log_set` Add log.
Message definition:

`log` Optional string. Log message to add. If set, *layer* and *level* are mandatory.

`layer` String. Layer name. Only mandatory if *log* is set.

`level` String. Log level: *error*, *warn*, *info* or *debug*. Only mandatory if *log* is set.

`dir` Optional string. Log direction: *UL*, *DL*, *FROM* or *TO*.

`ue_id` Optional number. UE_ID.

`flush` Optional boolean (default = false). If set, flushes fog file.

`rotate` Optional boolean (default = false). If set, forces log file rotation.

`cut` Optional boolean (default = false). If set, forces log file reset.

`log_reset`
Resets logs buffer.

`license` Retrieves license file information.
Response definition:

`products` String. List of products, separated by commas.

`user` String. License username.

`validity` String. License end of validity date.

`id` Optional string. License ID.

`id_type` Optional string. License ID type. Can be `host_id` or `dongle_id`

`uid` Optional string. License unique ID.

`filename` Optional string. License filename.

`server` Optional string. License server URL.

`server_id`
: Optional string. License server ID.

`quit` Terminates ltembmsgw.

`help` Provides list of available messages in *messages* array of strings and events to register in *events* array of strings.

`stats` Report statistics for LTEMBMSGW.
Every time this message is received by server, statistics are reset.
Warning, calling this message from multiple connections simultaneously will modify the statistics sampling time.
Response definition:

`cpu` Object. Each member name defines a type and its value cpu load in % of one core.

`instance_id`
: Number. Constant over process lifetime. Changes on process restart.

6.6 LTE messages

`service_start`
: Start a service.
Message definition:

  `service_id`
  : Integer. Identifier of service to start.

`service_stop`
: Stop a service.
Message definition:

  `service_id`
  : Integer. Identifier of service to stop.

7 Log file format

7.1 M2AP and GTP-U layers

When a message is dumped, the format is:

```
time layer - message
```

When a data PDU is dumped (debug level), the format is:

```
time layer dir ip_address short_content
        long_content
```

`time` Time using the selected format.

`layer` Indicate the layer (`[M2AP]` or `[GTPU]` here).

`dir` Direction: `TO` or `FROM`.

`ip_address`
source or destination IP address, depending on the `dir` field.

`short_content`
Single line content.

`long_content`

- M2AP: full ASN.1 content of the M2AP message if `layer.max_size > 0`.
- GTPU: hexadecimal dump of the message if `layer.max_size > 0`.

8 Change history

8.1 Version 2024-09-13

- added `license` remote API
- `com_logs_lock` parameter is renamed to `com_log_lock`. `com_logs_lock` is still supported for backward compatibility
- added `com_log_us` parameter

8.2 Version 2024-06-14

- OpenSSL library is upgraded to 1.1.1w

8.3 Version 2023-12-15

- added `loop_count` and `loop_delay` to remote API messages
- added `sim_events`, `sim_events_loop_count` and `sim_events_loop_delay`
- added `com_ssl_ca` parameter for SSL verification

8.4 Version 2023-06-10

- `com_logs_lock` parameter added to disable logs configuration change via remote API

8.5 Version 2023-03-17

- `com_addr` parameter now uses [::] address instead of 0.0.0.0 in the delivered configuration file to allow IPv6 connection

8.6 Version 2022-12-16

- `utc` parameter is added to remote API response messages

8.7 Version 2022-06-17

- OpenSSL library is upgraded to 1.1.1n
- `m2ap_bind_addr` parameter description is added
- `start_timestamp` and `end_timestamp` are added to `log_get` API

8.8 Version 2021-12-17

- `license` monitor command is added

8.9 Version 2021-09-17

- the minimum GLIBC version is now 2.17
- logs can be displayed with microseconds precision

9 License

Abbreviations

MBMS	Multimedia Broadcast Multicast Service
SYNC	MBMS synchronisation protocol
TMGI	Temporary Mobile Group Identity