LTE IMS Server

Version: 2020-12-16

Table of Contents

1 Introduction **1**

2 Features **2**

3 Requirements **3**

3.1 Hardware requirements 3
3.2 Known compatible UE 3
3.3 Software requirements 3

4 Installation **4**

4.1 Fedora setup 4
4.2 License key installation 4
4.3 Initial testing 4
4.4 Samsung S5 configuration 4

5 Configuration reference **7**

5.1 Configuration file syntax 7
5.2 Properties 8
5.2.1 User database options 13

6 Remote API **16**

6.1 Messages 16
6.2 Startup 17
6.3 Errors 18
6.4 Sample nodejs program 18
6.5 Common messages 18
6.6 LTE messages 22
6.7 LTE events 25
6.8 Examples 25

7 Command line monitor reference **27**

8 Log file format **28**

9 License **29**

Abbreviations **30**

1 Introduction

LTEIMS is an IMS standalone simple server. It has a built-in P-CSCF, I-CSCF, S-CSCF, HSS. It also allows SMS handling including SMS over SG by connecting to the Amarisoft MME.

2 Features

- Implements P-CSCF with built-in I-CSCF, S-CSCF and HSS.
- Support of SIP protocol.
- Support of MD5, AKAv1 and AKAv2 authentication.
- Support of ISIM cards using the XOR, Milenage or TUAK authentication algorithm.
- Support of IPSec (ESP/transport).
- Support of voice, video calls: MO and MT.
- Support of voice echo test.
- Support of hold.
- Support of SMS (GSM 3.40) using SIP MESSAGE and SMS over SG.
- Support of IPv4 and IPv6.
- Support of precondition and dedicated bearer using Rx interface.
- Support of emergency call.
- Configurable user database.
- External authentication using Cx interface.
- Command line monitor.
- Remote API using WebSocket.

3 Requirements

3.1 Hardware requirements

- LTEIMS 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.
- A VoLTE compatible UE is necessary (See [VoLTE Call], page 5, note that it may depends on UE).
- A test USIM with ISIM application should be plugged into the UE. IMSI and secret key must be known. A standard USIM may also work but it depends on the UE implementation.

3.2 Known compatible UE

The Amarisoft IMS server has been tested with the following UE models:

- Samsung S5
- LG MS870

3.3 Software requirements

- A 64 bit Linux distribution. Fedora 30 is the officially supported distribution.
  The following distributions are known as compatible:
  - Fedora 17 to 30
  - Cent OS 7
  - Ubuntu 12 to 18

4 Installation

The network access thru the Gigabit Ethernet port must be correctly configured.

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

4.1 Fedora setup

If you want to use SMS over SG with the Amarisoft MME, you need support of SCTP protocol for which the necessary packages are not usually installed. In order to install them, do as root user:

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

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

4.2 License key installation

LTEIMS needs a license key file to run. *It is associated to your PC, so if you replace it or change its hardware configuration you must contact Amarisoft to get a new license key.*

The following steps are needed to get this license file:

- Run LTEIMS:

  ```
  ./lteims config/ims.cfg
  ```

  It says that the license key is not present and prints a 16 digit hexadecimal code.
- Send by mail to `delivery@amarisoft.com` this hexadecimal code to your contact at Amarisoft. You will get back the `ltemme.key` license key file.
- Copy the `ltemme.key` file to the `${HOME}/.amarisoft/` directory (`${HOME}` is the home directory of the `root` user). You can use the shell variable `AMARISOFT_PATH` to change this path.

Once the license key is installed, `lteims` should start normally.

4.3 Initial testing

- Edit the file `config/ims.cfg` to set the address of the SIP interface. Normally it is the address of the Ethernet interface that will receive SIP packets.
  You can keep the current config if you use it with the Amarisoft MME and its `config/mme-ims.cfg` config file.
- Start the program as root with:

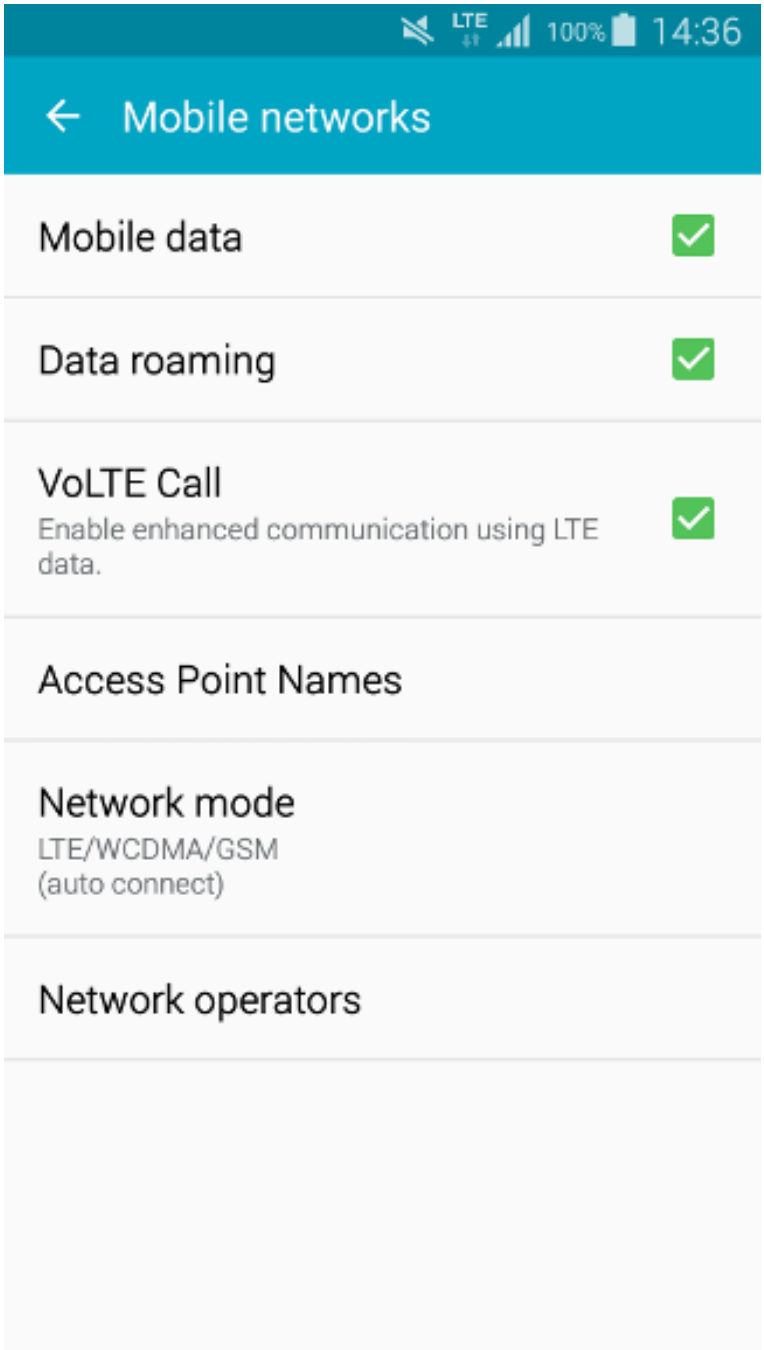
  ```
  ./lteims config/ims.cfg
  ```

  [The root access is only needed if you want IPSec support.]
- The command line interface is used to monitor the operation of LTEIMS and to change the logging options.
  Use `help` to get the list of commands and `quit` to stop the program.
- Use `users` to list the user database and registering state.

4.4 Samsung S5 configuration

Your UE must run at least Android 5.0 (Even if Android 5.0 is installed, try to update software (several times) as a sub-release is necessary).
If not, please update it.

To check your UE is configured for VoLTE, please go to `Settings/More networks/Mobile networks` of your handset and check `VoLTE Call` is checked:



We assume you are using the system with Amarisoft MME and `config/mme-ims.cfg` config file.
As there are two PDN defined, you must add them to the UE.

- Go to Settings/More networks/Mobile networks
- Turn on Data roaming
- Check VoLTE Call (If not present, it means your device is not up to date or does not support VoLTE).
- Go to Network operators, search for networks and select Amarisoft network.

- Go back to Mobile network.
- Add the first APN with the following parameters:
  - Name = Internet
  - APN = internet
  - APN type = default
- Save it and select it.
- Add second APN with following parameters:
  - Name = IMS
  - APN = ims
  - APN type = ims
- Save it and do not select it (This APN may not be displayed).
- Reboot your phone

5 Configuration reference

5.1 Configuration file syntax

The main configuration file uses a syntax very similar to the Javascript Object Notation (JSON) with few extensions.

1. Supported types:
   - 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, .... ]`
2. The basic operations `+`, `-`, `*` and `/` are supported with numbers and complex numbers. `+` also concatenates strings. The operators `!`, `||`, `&&`, `==`, `!=`, `<`, `<=`, `>=`, `>` are supported too.
3. The numbers `0` and `1` are accepted as synonyms for the boolean values `false` and `true`.
4. {} at top level are optional.
5. " for property names are optional.
6. Properties can be duplicated.
   Merge will be done by recursively overriding values considering reading direction.

```
{
    value: "foo",
    value: "bar",
    sub: {
        value: "foo"
    },
    sub: {
        value: "bar"
    }
}
```

   Will be equivalent to:

```
{
    value: "bar",
    sub: {
        value: "bar"
    }
}
```

7. Files can be included using *include* keyword (must not be quoted) followed by a string (without :) representing the file to include (path is relative to current file) and terminating by a comma.
   Arrays can't be included.
   Merge will be done as for duplicate properties.
   If *file1.cfg* is:

```
value: "foo",
include "file2.cfg",
foo: "foo"
```

   And *file2.cfg* is:

```
value: "bar",
```

```
    foo: "bar"
```

Final config will be:

```
{
    value: "bar",
    foo: "foo"
}
```

8. A C like preprocessor is supported. The following preprocessor commands are available:

   `#define` *var expr*
   : Define a new variable with value *expr*. *expr* must be a valid JSON expression. Note that unlike the standard C preprocessor, *expr* is evaluated by the preprocessor.

   `#undef` *var*
   : Undefine the variable *var*.

   `#include` *expr*
   : Include the file whose filename is the evaluation of the string expression *expr*.

   `#if` *expr* Consider the following text if *expr* is true.

   `#else` Alternative of `#if` block.

   `#elif` Composition of `#else` and `#if`.

   `#endif` End of `#if` block.

   `#ifdef` *var*
   : Shortcut for `#if defined(var)`

   `#ifndef` *var*
   : Shortcut for `#if !defined(var)`

   In the JSON source, every occurrence of a defined preprocessor variable is replaced by its value.

9. Backquote strings: JSON expression can be inserted in backquote delimited strings with the `${expr}` syntax. Example: `` `abc${1+2}d` `` is evaluated as the string `"abc3d"`. Preprocessor variables can be used inside the expression.

The System Information Blocks use the ASN.1 GSER syntax defined in RFC 3641 (Generic String Encoding Rules for ASN.1 Types). The description of the exact content of the System Information Blocks can be found in 3GPP TS 36.331 (RRC).

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 27.

`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, RRC and PCDP layers).
- time=[sec|short|full]. Display the time as seconds, time only or full date and time (default = time only).
- file=cut. Close current file log and open a new one.
- file.rotate=now. Rename current log with timestamp and open new one.
- file.rotate=*size*. Rename current log every time it reaches *size* bytes open new one. Size is an integer and can be followed by K, M or G.
- file.path=*path*. When log rotation is enabled, 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: `ims`, `sip`, `media`, `rx`, `cx`

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

`sip_addr` Array. Each item is an object representing a SIP server socket defined as follow:

`addr` String. Set the IP address (and an optional port) on which IMS server will listen for SIP packets. The default port is 5060.

`bind_addr`
Optional string. Defines network interface on which IMS will listen.
If not specified, the addr parameter is used.

`port_min` Optional integer (Default is 10000). Defines lower bound of UDP media socket.

`port_max` Optional integer (Default is 20000). Defines upper bound of UDP media socket.

NB:

- SIP socket object can be represented by a simple string. Thus, it will represent `addr` parameter and all other parameters will use default value.
- For legacy, `sip_addr` can be a single SIP socket (Object or String) instead of an Array.

`sctp_addr`
String. Set the IP address (and an optional port) for MME connection.
This is only necessary for SMS over SG feature.

`cx_server_addr`
String. Set the IP address (and optional port) of Cx SCTP connection to the HSS.
The default port is 3868.

`cx_bind_addr`
Optional string. IP address and optional port on which the Cx SCTP connection is bound. If not set, `sctp_addr` is used.

`cx_origin_realm`
: Optional string. Defines the string sent in the Origin-Realm AVP for Cx messages. Default is set to `amarisoft.com`.

`cx_origin_host`
: Optional string. Defines the string sent in the Origin-Host AVP for Cx messages. Default is set to `ims.amarisoft.com`.

`cx_watchdog_duration`
: Optional integer (range 0 to 36000000, default = 0). Tw watchdog timer in milliseconds to send the Diameter Device Watchdog Request message. The value 0 deactives the watchdog.

`rx_server_addr`
: Optional string. Set the IP address (and optional port) of Rx SCTP connection to the MME. The default port is 3868. If not set, `cx_server_addr` is used.

`rx_bind_addr`
: Optional string. IP address and optional port on which the Rx SCTP connection is bound. If not set, `cx_bind_addr` is used.

`rx_origin_realm`
: Optional string. Defines the string sent in the Origin-Realm AVP for Rx messages. Default is set to `amarisoft.com`.

`rx_origin_host`
: Optional string. Defines the string sent in the Origin-Host AVP for Rx messages. Default is set to `ims.amarisoft.com`.

`rx_watchdog_duration`
: Optional integer (range 0 to 36000000, default = 0). Tw watchdog timer in milliseconds to send the Diameter Device Watchdog Request message. The value 0 deactives the watchdog.

`domain`
: String. Global SIP domain used for IMPU and authentication. May be overriden at user level.
This parameter is not used to recover IMPU.

`tcp_threshold`
: Optional integer (default = 1300). Set packet threshold in bytes to use TCP instead of UDP.

`session_expires`
: Optional integer (default = 3600); Set session expires header value in seconds.

`100rel`
: Optional Boolean (default = true). Enable/disable 100rel support.

`precondition`
: Boolean. If true, precondition with QoS will be handled by IMS.
A Rx connection is necessary to allow dedicated bearer establishment.

`p_called_party_id`
: Optional boolean (default is false). Enable P-Called-Party-ID header for INVITE and MESSAGE requests.

`ipsec`
: Optional boolean (default is true). Enable/disable support of ipsec.

`ipsec_aalg_list`
: Array of strings. Each string represent IPSec authentication algorithm supported by IMS.
"null" may be used to indicate no authentication.

`ipsec_ealg_list`
: Array of strings. Each string represent IPSec encryption algorithm supported by IMS. "null" may be used to indicate no encryption.

`dialog_timeout`
: Optional integer (default = 30). Time in seconds of call session. Stop call if no activity has been detected during this time.

`auth_on_register_only`
: Optional boolean (default = false). If true, don't try to authenticate other request than register.

`com_addr` Optional string. Address of the WebSocket server remote API. See [Remote API], page 15.
If set, the WebSocket server for remote API will be enabled and bound to this address.
Default port is 9003.
Setting IP address to 0.0.0.0 will make remote API reachable through all network interfaces.

`com_name` Optional string. Sets server name. IMS 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_auth` Optional object. If set, remote API access will require authentication.
Authentication meachanism is describe in [Remote API Startup], page 17, 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.

`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"
}
```

`sms_expires`
: Integer (default = 86400). Delay in seconds before SMS is removed from database

`sms_hook_only`
: Optional boolean (default = false). If set, when SMS is received and at least one WebSocket client has registered to `sms` event, don't process SMS internally (Only CP/RP layer will be handled).

`binding_expires`
: Integer (default = 3600). Default duration in seconds for registration.

`subscribe_expires`
: Integer (default = 0, max = 864000). Subscription expiration. If set to 0, use value sent by UE.

`user_agent`
: String. SIP user agent.

`timer_t1` Optional number (default = 2). SIP T1 timer duration in seconds.

`timer_t2` Optional number (default = 16). SIP T2 timer duration in seconds.

`timer_t4` Optional number (default = 17). SIP T4 timer duration in seconds.

`custom_headers`
: Array of object. Each object represents a custom header to add to requests and/or responses, defined as follows:

  `name` Header name

  `value` Header value

  `methods` String or array of strings of the SIP method on which to apply custom headers. `*` can be used for all methods.

  `codes` Number or array of numbers of the SIP responses on which to apply custom headers. `0` can be used for all codes.

`sms_retry_delay`
: Integer. Time in s to retry SMS sending.

`echo` String. If set, this defines the phone number for echo service.

`mt_call_sdp_file`
: String. File to use as SDP when using MT call.

`sms_message_filter`
: Optional object. Allows to define the IMS behavior for a list of SMS related messages.
  Each property name represents a SMS message type. The ones currently supported are `cp_data`, `cp_ack`, `rp_data` and `rp_ack`.
  Each property value is an enum: `treat` (message is processed), `ignore` (message is ignored) or `reject` (message is rejected).
  By default all procedures are treated.

Example:

```
sms_message_filter: {
    cp_data: "treat",
    rp_ack: "reject"
}
```

`sms_forced_cp_cause`
: Optional integer (range 0 to 255). Allows to override the CP error cause selected by the IMS with the one configured. Set to 0 to deactivate the override.

`sms_forced_rp_cause`
: Optional integer (range 0 to 255). Allows to override the RP error cause selected by the IMS with the one configured. Set to 0 to deactivate the override.

`mms_server_bind_addr`
: Defines network interface on which MMS server will listen.

`mms_expires`
: Optional integer (default = 86400). Delay in seconds before MMS is removed from database.

5.2.1 User database options

`ue_db`

Array of objects. Configure the user database. Each element is an entry for one user. Note that this part can be shared between Amarisoft MME and IMS. The following properties are available:

`imsi` String. Set the IMSI.

`multi_sim`
: Optional boolean (default = false). If true, allow several UEs to have the same IMSI (useful when using several identifical test SIM cards in different UEs at the same time). They are distinguished with their IMEI. Note: it is only allowed with the XOR authentication algorithm.

`impi` String. Defines user IMPI. Must be fully filled with hostname if necessary.
If you don't know your IMPI, please look at IMS logs inside *REGISTER* request. The IMPI must match the *username* argument inside *Authorization* header.

`impu` Array of string or object. Each string represent an IMPU and can be a sip URI or a telephone number.
Note that sip URI must not include hostname.
If IMPU does not start by a scheme, it is assumed to be a sip URI.
Ex:

- sip:user
- user
- tel:+33123456789

If impu is an object, it has following members:

`impu` IMPU as defined above.

`imei` IMEI associated to this IMPU. Allows to filter calls and SMS for a specific UE.
Only relevant if multi_sim is set to true.

`code` Number. Only relevant for echo impu. Server will use this as SIP answer code.

`content` String. If *code* is set, response body will be filled with *content* file.

`content_type`
String. Mandatory if *code* and *content]* are set, will define response content type.

`anonymous`
Optional boolean (default is false). If true, allow Anonymous connection (Emergency call).

`authentication`
Optional boolean (default is true). If false, disable authentication.

`ring_only`
Optional boolean (default is false). If true, IMS will go up to ringing state but not further.

`precondition`
Optional string. Values can be "on", "off" or "silent".
On mode: IMS will try to guess precondition from supported header, SDP content and/or VoLTE compatibility of client.
Off mode: no precondition and no dedicated bearer establishment.
Silent mode: dedicated bearers will be established whatever the SIP and SDP content.

`100rel` Optional boolean (default = true). Enable/disable 100rel support for this IMPU.

`asserted` Optional boolean (default: false). If set, this `impu` will be used for asserted identity.

`associated`
Optional boolean (default: true). If set to false, this `impu` will not be used for associated URI.

`display_name`
Optional string. If set, SIP headers will use this field for display name.

`domain` Optional string. Used to override user or global config.

`p_called_party_id`
Optional string. If set, forces P-Called-Party-ID header for INVITE and MESSAGE requests, no matter if `p_called_party_id` global parameter is set or not.

`res_len` Optional integer (default = 8). Defines length of response in bytes during authentication. For TUAK authentication algorithm, the value must be 4, 8 or 16 bytes long.

`authent_type`
Optional string (default = AKAv1). Defines minimum authentication level.

If client does no specify authentication algo, server will use this value. Else, server will allow authentication only if client provided algo is at least the one specified by this parameter.
Values are (from lowest security to highest):

`none` Disable authentication.

`MD5` MD5 digest authentication.

`AKAv1` AKAv1 authentication.

`AKAv2` AKAv2 authentication.

`pwd` Optional string. Password set for MD5 authentication. If set and *authent_type* is not set, *authent_type* is set to MD5.

`mt_call_sdp_file`
Optional string. File to use as SDP when using MT call. Overrides global paramater.

`domain` Optional string. If set, overrides global config.

`auth_on_register_only`
Optional boolean. If set, overrides global config.

`force_sms_over_sg`
Optional boolean. If set, forces use of SMS over SG.

`ue_db_filename`
Optional string. If present, store the current IMS state in a persistent file. The IMS state contains in particular the registration info and pending SMS.

6 Remote API

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

6.1 Messages

Messages exchanged between client and LTEIMS 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 double. 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.

- 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.
  Usefull to send command with absolute time.

- 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, LTEIMS will send a first message with name and type of `PROG`.

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

```
{
    "message": "ready",
    "type": "IMS",
    "name: <name>
}
```

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

```
{
    "message": "authenticate",
    "type": "IMS",
    "name: <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": "IMS",
    "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 LTEIMS.
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 127.0.0.1:9003 '{"message": "config_get"}'
```

6.5 Common messages

`config_get`

Retrieve current config.

Response definition:

`type` Always "IMS"

`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 8,

`max_size` See [log_options], page 8,

`count` Number. Number of bufferizer logs.

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

`path` Optional string. Log rotation path.

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

`cch` Boolean. True if CCH dump is enabled (UE only).

`signal` Boolean. True if PHY layer signal dump is enabled.

`config_set`

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

`logs` Object. Represent logs configuration. Same structure as config_get (See [config_get logs member], page 18).

All elements are optional.
Layer name can be set to `all` to set same configuration for all layers.

`precondition`
: Optional boolean (default is false). If true, precondition with QoS will be handled by IMS.
IMS must be connected to MME to allow dedicated bearer establishment.

`sms_retry_delay`
: Integer. Time in s to retry SMS sending.

`sms_expires`
: Integer (default = 86400). Delay in seconds before SMS is removed from database

`binding_expires`
: Integer (default = 3600). Default duration in seconds for registration.

`subscribe_expires`
: Integer (default = 0, max = 864000). Subscription expiration. If set to 0, use value sent by UE.

`auth_on_register_only`
: Optional boolean (default = false). If true, don't try to authenticate other request than register.

`dialog_timeout`
: Optional integer (default = 30). Time in seconds of call session. Stop call if no activity has been detected during this time.

`p_called_party_id`
: Optional boolean (default is false). Enable P-Called-Party-ID header for INVITE and MESSAGE requests.

`sms_message_filter`
: Optional object. Allows to define the IMS behavior for a list of SMS related messages.
Each property name represents a SMS message type. The ones currently supported are `cp_data`, `cp_ack`, `rp_data` and `rp_ack`.
Each property value is an enum: `treat` (message is processed), `ignore` (message is ignored) or `reject` (message is rejected).
By default all procedures are treated.

Example:

```
sms_message_filter: {
    cp_data: "treat",
    rp_ack: "reject"
}
```

`sms_forced_cp_cause`
: Optional integer (range 0 to 255). Allows to override the CP error cause selected by the IMS with the one configured. Set to 0 to deactivate the override.

`sms_forced_rp_cause`
: Optional integer (range 0 to 255). Allows to override the RP error cause selected by the IMS with the one configured. Set to 0 to deactivate the override.

`log_get` Get logs.
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.

`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 8.
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 8.

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

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. Depends on log time configuration (See [log_options], page 8):
If time is set to *short*, milliseconds since start of the day.
If time is set to *full*, milliseconds since January 1st 1970.
If time is set to *sec*, milliseconds since start of the LTEIMS.

`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.

Note that only one request can be sent by client.
If a request is sent before previous one has returned, previous one will be sent without matchine min/max/timeout conditions.

`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.

`quit` Terminates lteims.

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

`stats` Report statistics for LTEIMS.
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.
- `counters` Object. List of counters, with following sub members:
  - `messages` Object. Each member name is the message name and its value is its occurence.
    To get list of message, type *cevent help msg* in LTEIMS monitor.
  - `errors` Object. Each member name is the error name and its value is its occurence.
    To get list of message, type *cevent help msg* in LTEIMS monitor.

`register` Register client to message generated by server. Message definition:

- `register` String or array of string. List of message to register to.
  Can be `users_update`, `sms`
- `unregister` String or array of string. List of message to unregister.
  Can be `users_update`, `sms`

6.6 LTE messages

`users_get`
: Get users state.
Response definition:

`users` Array of object. Each item represents a user with following parameters:

`impi` String. IMPI of user (IP Multimedia Private identity).

`force_sms_over_sg`
Optional boolean. Current SMS over SG forcing state.

`bindings` Array of object. One for each contact binding:

`uri` String. Contact URI.

`impu` Array of strings. List of associated IMPUs.

`q` Number. Contact priority.

`video` Optional boolean. Video support.

`sms` Optional boolean. SMS pending.

`imei` Optional string. IMEI.

`expires` Integer. Number of seconds before binding expiration.

`dialogs` Array of object. One for each current dialog:

`remote` String. IMPI of remote user.

`sms` Integer. Number of pending SMS.

`users_add`
: Add users.
Message definition:

`users.` Array of object. Same as info in configuration file: See [ue_db], page 13.

`user_set` Configure user.
Message definition:

`impi` String. IMPI of user to configure.

`force_sms_over_sg`
Optional boolean. Set/unset forcing of SMS over SG.

`sms` Send SMS.
Message definition:

`impi` Optional string. IMPI of user (IP Multimedia Private identity).

`impu` Optional string. If IMPI is not set, try to get user from IMPU (IP Multimedia Public identity).

`text` String. SMS text to send.

`sender` Optional string. Sets SMS sender.

`validity` Optional integer (Default = 86400). Validity period in seconds.

`binary` Optional string. If set (and `text` is not set), must be a base64 string representing binary data of the TP-User-Data.

`binary_hex`
: Optional string. If set (and `text` is not set), must be an hexadecimal string representing binary data of the TP-User-Data.

`tp_udl`
: Optional integer. Used when `binary` is set. If present, it sets the TP-User-Data-Length field. If not present, the TP-User-Data-Length field is set to the number of octets of the `binary` field.

`tp_udhi_present`
: Optional boolean (default is false). When `binary` is set, indicates if TP-User-Data start with a user-data header or not.

`pid`
: Optional integer (default is 0). Defines protocol identifier.

`dcs`
: Optional integer (default is 4). Defines data coding scheme. If the `text` parameter was provided, it's up to the user to ensure that the `dcs` value is coherent with the encoding automatically selected (7 bit default GSM alphabet or UCS2).

`mms`
: Send MMS.
Message definition:

`filename`
: String. File name to send. Extensions jpg, jpeg, png, gif and txt are supported.

`from`
: String. Sender phone number.

`to`
: String. Receiver phone number.

`mt_call`
: Initiate a mobile terminating call.
Message definition:

`impi`
: String. IMPI (IP Multimedia Private identity) of user to call.

`impu`
: String. IMPU (IP Multimedia Public identity) of user to call.

`contact`
: String. Contact SIP uri of user to call.

`sip_file`
: Optional string. Define file to use as sdp. Will override *mt_call_sdp_file* parameter.

`caller`
: Optional string. Use it to force caller IMPU. If IMPU is in user database, the P-Asserted-Identity header will be added.

`duration`
: Optional number. If set, call duration in seconds (The server will close the dialog).

Response definition:

`session_id`
: String. If call has started, provides its session ID.

`dialog_get`
: Get list of current pending dialogs.
Dialog will persist 30s after being stopped. Message definition:

`session_id`
: Optional string. If set, filter on session ID.

Response definition:

`dialogs`
: Array of object representing dialogs as follow:

`session_id`
: String. Dialog session ID.

`state` String. Dialog state, can be `init`, `ringing`, `start`, `hold` or `stop`.

`type` String. Dialog type, can be `server`, `echo` or `mt call`

`to` Callee IMPU.

`from` Caller IMPU.

`mt_dialog`
Optional string. In case of server dialog, session id of associated MT dialog.

`mo_dialog`
Optional string. In case of client dialog, session id of associated MO dialog.

`date` Integer. Dialog creation time in seconds since 1st January 1970.

`duration` Number. Number of seconds since dialog has started.

`event_list`
Array of object representing events that has occured during dialog lifetime.
Each element have the following definition:

- `type` String. Event type, can be `state`, when a state change occurs, `send` and `recv` when receiving or sending message.
- `timestamp`
  Number. Event time in seconds since dialog creation.
- `state` String. Dialog state when event has occured as defined above.

`medias` Array of object representing media state.
Each media is an object having following definition:

- `type` String. Media type, can be `audio` or `video`.
- `qos` String. QoS state, can be:
  - disabled: QoS not enabled, IETF mode used.
  - required: QoS required but not yet initiated.
  - pending: QoS dedicated bearer establishment in progress.
  - erab_set: QoS done.
- `dir` String. Media current direction, can be `sendrecv`, `sendonly`, `recvonly` or `inactive`.
- `rtp_addr` String. RTP packets destination address.
- `rtp_recv_count`
  Integer. Number of RTP packets received.
- `rtp_send_count`
  Integer. Number of RTP packets sent.

`rtcp_addr`
String. RTCP packets destination address.

`rtcp_recv_count`
Integer. Number of RTCP packets received.

`rtcp_send_count`
Integer. Number of RTCP packets sent.

`dialog_stop`
Forces termination of a dialog.
Message definition:

`session_id`
String. Session ID of dialog to stop.

`reinvite` Forces sending of INVITE of a started dialog.
Message definition:

`session_id`
String. Session ID of dialog.

`unregister`
Force a network deregistration of a binding. Message definition:

`uri` String. Binding URI (Address of Record)

6.7 LTE events

Following events are sent by IMS if they have been registered on WebSocket.

`sms` Generated by SMS reception:

`sender` String. SMS originator.

`destination`
String. SMS destination.

`text` String. SMS text.

`binary` String. If `text` is not set, base64 encoded string of SMS data.

`dcs` Integer. Data coding scheme.

`users_update`
Event generated when a change occurs on a user (Registration, call, sms...).

`users_update`
Array of object. Each item represents a user (See [users_get], page 22).

6.8 Examples

1. Config
    1. Client sends

```
{
    "message": "config_get",
    "message_id": "foo"
}
```

    2. Server replies

```
{
    "message_id": "foo",
```

```
    "message": "config_get",
    "name": "UE",
    "logs": {
        "phy": {
            "level": "error",
            "max_size": 0
        },
        ...
        "rrc": {
            "level": "debug",
            "max_size": 1
        }
    }
}
```

2. Error
    1. Client sends

    ```
    {
        "message": "bar",
        "message_id": "foo"
    }
    ```

    2. Server replies

    ```
    {
        "message_id": "foo",
        "message": "bar",
        "error": "Unknown message: bar"
    }
    ```

7 Command line monitor reference

The following commands are available:

`help` Display the help. Use `help` *`command`* to have a more detailed help about a command.

`log [`*`log_options`*`]`
Display the current log state. If *log_options* are given, change the log options. The syntax is the same as the `log_options` configuration property.

`mme` Lists MME connections

`sms` *`impi`* `or` *`impu`* *`text`*
Send a SMS to the user identified by *impi* or *impu* if *impi* has not been found.

`sms_flush` *`impi`*
Flush pending SMS.

`mms` *`filename`* *`from`* *`to`*
Send a MMS to the user identified by *to*. Extensions jpg, jpeg, png, gif and txt are supported. For any other extension value, the content type is interpreted as octet stream.

`mt_call` *`callee`* `[-d` *`duration`*`]` `[`*`sip_file`*`]` `[`*`caller`*`]`
Initiate a mobile terminating call.
callee can be IMPI, IMPU or contact URI.
sip_file Define file to use as sdp. Will override *mt_call_sdp_file* parameter.
caller can be used to force caller IMPU. If IMPU is in user database, *duration* duration of the call in seconds before server closes it.
the P-Asserted-Identity header will be added.

`dialog` Lists all dialogs.

`dialog_stop` *`dialog`* *`index`*
Stops dialog.

`reinvite` *`dialog`* *`index`*
Forces sending of INVITE of a started dialog.

`quit` Stop the program and exit.

8 Log file format

9 License

Abbreviations

APN	Access Point Name
IMPU	IP Multimedia Public Identity
IMPI	IP Multimedia Private Identity