LMT IoT Shortcut SDK API

1.1.1

Chapter 1

File Index

1.1 File List

Here is a list of all files with brief descriptions:

Chapter 2

File Documentation

2.1 lmt_coap_manager.h File Reference

Macros

- #define APP_COAP_MAX_MSG_LEN 1280

Enumerations

- enum MailerWaitModes { WAIT_ON_TIMEOUT = 0 , WAIT_FOREVER = 1 }

  Enum for mailer wait control.

Functions

- void getDeviceSN (char ∗sn_buffer, size_t ∗buffer_size)

  Reads device serial number (SN).
- bool isSocketConnected (void)

  Checks if the device is currently connected to the IoT network.
- bool isModemInitialized (void)

  Checks if the modem is initialized.
- MailerWaitModes getMailerWaitMode (void)

  Returns the mailer wait mode.
- void setMailerWaitMode (MailerWaitModes mode)

  Sets the mailer wait mode.
- uint32_t getPacketCounter (void)

  Retrieves the current packet counter value.
- void setPacketCounter (uint32_t counter)

  Sets the packet counter to a specific value.
- void resetPacketCounter (void)

  Resets the packet counter to zero.
- uint32_t getPacketCounterLimit (void)

  Retrieves the packet counter limit that will triger the radio data update.
- void setPacketCounterLimit (uint32_t limit)

  Sets a new packet counter limit that will triger the radio data update.
- int getNetworkQuality (int32_t ∗rsrp, int32_t ∗rsrq, int32_t ∗snr)

Function gets the cached signal quality from the modem.

- void triggerDataPacking (bool set_include_radio_parms)

  Enable packer thread to create a CoAP message.

- void triggerMailer (bool trigger_radio_pata_packing)

  Enable mailer thread; first data packet containing radio data can be created.

- void modemShutdown (void)

  Function to close the socket and power off the modem.

- void sendEventCmdRes (int res)

  Set and send terminal command result.

2.1.1 Macro Definition Documentation

2.1.1.1 APP_COAP_MAX_MSG_LEN

```
#define APP_COAP_MAX_MSG_LEN 1280
```

2.1.2 Enumeration Type Documentation

2.1.2.1 MailerWaitModes

```
enum MailerWaitModes
```

Enum for mailer wait control.

Enumerator

WAIT_ON_TIMEOUT	
WAIT_FOREVER	

```
00017 {
00018     WAIT_ON_TIMEOUT = 0,
00019     WAIT_FOREVER    = 1
00020 } MailerWaitModes;
```

2.1.3 Function Documentation

2.1.3.1 getDeviceSN()

```
void getDeviceSN (
            char * sn_buffer,
            size_t * buffer_size)
```

Reads device serial number (SN).

Parameters

sn_buffer	Pointer to store the SN.
buffer_size	Pointer to buffer size; updated with actual SN length.

2.1.3.2 getMailerWaitMode()

```
MailerWaitModes getMailerWaitMode (
            void )
```

Returns the mailer wait mode.

Returns

WAIT_ON_TIMEOUT (false) if the mailer waits on timeout, WAIT_FOREVER (true) if the mailer has to be started by user

2.1.3.3 getNetworkQuality()

```
int getNetworkQuality (
            int32_t * rsrp,
            int32_t * rsrq,
            int32_t * snr)
```

Function gets the cached signal quality from the modem.

Parameters

rsrp	Pointer to store the RSRP value
rsrq	Pointer to store the RSRQ value
snr	Pointer to store the SNR value

Returns

0 if values values, -EINVAL otherwise

2.1.3.4 getPacketCounter()

```
uint32_t getPacketCounter (
            void )
```

Retrieves the current packet counter value.

Returns

Current packet counter value.

2.1.3.5 getPacketCounterLimit()

```
uint32_t getPacketCounterLimit (
            void )
```

Retrieves the packet counter limit that will triger the radio data update.

Returns

The maximum allowed packet counter value.

2.1.3.6 isModemInitialized()

```
bool isModemInitialized (
            void )
```

Checks if the modem is initialized.

Returns

true if the modem is initialized, false otherwise.

2.1.3.7 isSocketConnected()

```
bool isSocketConnected (
            void )
```

Checks if the device is currently connected to the IoT network.

Returns

true if the device is connected, false otherwise.

2.1.3.8 modemShutdown()

```
void modemShutdown (
            void )
```

Function to close the socket and power off the modem.

2.1.3.9 resetPacketCounter()

```
void resetPacketCounter (
            void )
```

Resets the packet counter to zero.

2.1.3.10 sendEventCmdRes()

```
void sendEventCmdRes (
            int res)
```

Set and send terminal command result.

Parameters

res	Result of the command, 0 if successful, negative value otherwise

2.1.3.11 setMailerWaitMode()

```
void setMailerWaitMode (
            MailerWaitModes mode)
```

Sets the mailer wait mode.

Parameters

mode	WAIT_ON_TIMEOUT (false) if the mailer has to wait on timeout, WAIT_FOREVER (true) if the user controls the mailer start

2.1.3.12 setPacketCounter()

```
void setPacketCounter (
            uint32_t counter)
```

Sets the packet counter to a specific value.

Parameters

counter	The new value for the packet counter.

2.1.3.13 setPacketCounterLimit()

```
void setPacketCounterLimit (
            uint32_t limit)
```

Sets a new packet counter limit that will triger the radio data update.

Parameters

limit	The new upper limit for the packet counter.

2.1.3.14 triggerDataPacking()

```
void triggerDataPacking (
            bool set_include_radio_parms)
```

Enable packer thread to create a CoAP message.

Parameters

set_include_radio_parms	when set, radio data will be added

2.1.3.15 triggerMailer()

```
void triggerMailer (
            bool trigger_radio_pata_packing)
```

Enable mailer thread; first data packet containing radio data can be created.

Parameters

trigger_radio_pata_packing	when set, radio data packet will be added

2.2 lmt_common.h File Reference

Macros

- #define FIRST_USER_STATUS_BIT 16
- #define LAST_USER_STATUS_BIT 31

Functions

- void criticalError (void)

  Calls assert and triggers system reset in case of critical error.
- bool checkBootOkMask (uint32_t mask)

  Returns the flag if the given mask is set in BootOK status.
- void setUserBootOkMask (uint32_t mask)

  Set user app boot ok mask.
- void setBootOkBit (uint32_t bit)

  Set appropriate status bit.
- void resetStatusBit (uint32_t bit)

  Reset appropriate status bit.

2.2.1 Macro Definition Documentation

2.2.1.1 FIRST_USER_STATUS_BIT

```
#define FIRST_USER_STATUS_BIT 16
```

2.2.1.2 LAST_USER_STATUS_BIT

```
#define LAST_USER_STATUS_BIT 31
```

2.2.2 Function Documentation

2.2.2.1 checkBootOkMask()

```
bool checkBootOkMask (
            uint32_t mask)
```

Returns the flag if the given mask is set in BootOK status.

Parameters

mask	The mask of the interested flags

Returns

true if all flags are set, false otherwise

2.2.2.2 criticalError()

```
void criticalError (
            void )
```

Calls assert and triggers system reset in case of critical error.

2.2.2.3 resetStatusBit()

```
void resetStatusBit (
            uint32_t bit)
```

Reset appropriate status bit.

Parameters

bit	The number of bit that has to be reset

2.2.2.4 setBootOkBit()

```
void setBootOkBit (
            uint32_t bit)
```

Set appropriate status bit.

Parameters

bit	The number of bit that has to be set

2.2.2.5 setUserBootOkMask()

```
void setUserBootOkMask (
            uint32_t mask)
```

Set user app boot ok mask.

Parameters

mask	The mask of the interested flags

2.3 lmt_proto_handler.h File Reference

Macros

- #define MAX_TRACKS_COUNT 12
- #define MAX_PERIODS_COUNT 3
- #define MAX_COLUMNS_COUNT 50
- #define MAX_TAPE_COUNT 1
- #define MAX_ACTION_PARAMETERS_SIZE 256
- #define I_TAPE 0

Functions

- void dumpMemory (uint8_t *ptr, uint16_t size)

  Dumps in HEX the memory region at the given address of the given size.
- bool isDataChanged (void)

  Returns flag signalising that data is updated in comparison to the last encoded message.
- void updatePeriod (uint8_t i_tape, uint32_t value)

  Function to add a new period to the Periods array of a given Tape. During operation first add the period, then measurement. The duplicate value entries will be not added. The entries withouth measurements will be overwritten. On overflow it will reset the Periods array. Corrects also the Columns_count if Columns_count >= MAX_COLUMNS↩ _COUNT.
- uint32_t getLastPeriod (uint8_t i_tape)

  Returns the last defined period of the Tape instance given by pointer.
- int addColumnToTape (uint8_t i_tape, uint32_t period, int32_t *p_measurements)

  Adds a measuremens column and it period to the given Tape.
- pb_size_t getTapeRecordsCount (uint8_t i_tape)

  Returns actual column count in Tape.
- void rewindTape (uint8_t i_tape)

  Resets the Tape keeping the last period entry.
- void restartMeasurements (void)

  Clear sensor measurements keeping the last measuremet periods.
- bool encodeMessage (void)

  Encodes uplink message.
- uint16_t getEncodedMsgLen (void)

  Get encoded message length.
- const uint8_t * getEncodedMsgBuffer (void)

  Get pointer to encoded message buffer.
- bool isCompressionCheckRequired (void)

  Checks if the data will fit UDP packet with no compression.
- bool isUdpPacketFull (void)

  Checks if the UDP packet is full.
- bool decodeMessage (const uint8_t *p_buffer)

  Decodes downlink message.

2.3.1 Macro Definition Documentation

2.3.1.1 I_TAPE

```
#define I_TAPE 0
```

2.3.1.2 MAX_ACTION_PARAMETERS_SIZE

```
#define MAX_ACTION_PARAMETERS_SIZE 256
```

2.3.1.3 MAX_COLUMNS_COUNT

```
#define MAX_COLUMNS_COUNT 50
```

2.3.1.4 MAX_PERIODS_COUNT

```
#define MAX_PERIODS_COUNT 3
```

2.3.1.5 MAX_TAPE_COUNT

```
#define MAX_TAPE_COUNT 1
```

2.3.1.6 MAX_TRACKS_COUNT

```
#define MAX_TRACKS_COUNT 12
```

2.3.2 Function Documentation

2.3.2.1 addColumnToTape()

```
int addColumnToTape (
            uint8_t i_tape,
            uint32_t period,
            int32_t * p_measurements)
```

Adds a measuremens column and it period to the given Tape.

Parameters

i_tape	the Tape index
period	the period value
p_measurements	pinter to the measurements array in count of MAX_TRACKS_COUNT

Returns

number of remaining empty columns, -EINVAL if i_tape is out of range

2.3.2.2 decodeMessage()

```
bool decodeMessage (
            const uint8_t * p_buffer)
```

Decodes downlink message.

Parameters

p_buffer	pointer to incoming message

Returns

success flag

2.3.2.3 dumpMemory()

```
void dumpMemory (
            uint8_t * ptr,
            uint16_t size)
```

Dumps in HEX the memory region at the given address of the given size.

Parameters

∗ptr	The starting memory address
size	The size of the memory dump

2.3.2.4 encodeMessage()

```
bool encodeMessage (
            void )
```

Encodes uplink message.

Returns

success flag

2.3.2.5 getEncodedMsgBuffer()

```
const uint8_t * getEncodedMsgBuffer (
            void )
```

Get pointer to encoded message buffer.

Returns

Constant pointer to encoded message buffer

2.3.2.6 getEncodedMsgLen()

```
uint16_t getEncodedMsgLen (
            void )
```

Get encoded message length.

Returns

Current encoded message length

2.3.2.7 getLastPeriod()

```
uint32_t getLastPeriod (
            uint8_t i_tape)
```

Returns the last defined period of the Tape instance given by pointer.

Parameters

i_tape	the Tape index

Returns

the last defined period or 0 if period is not set

2.3.2.8 getTapeRecordsCount()

```
pb_size_t getTapeRecordsCount (
            uint8_t i_tape)
```

Returns actual column count in Tape.

Parameters

i_tape	the Tape index

Returns

Column count in Tape

2.3.2.9 isCompressionCheckRequired()

```
bool isCompressionCheckRequired (
            void )
```

Checks if the data will fit UDP packet with no compression.

Returns

true if data will not fit, false otherwise

2.3.2.10 isDataChanged()

```
bool isDataChanged (
            void )
```

Returns flag signalising that data is updated in comparison to the last encoded message.

Returns

A flag signalising that data had been changed

2.3.2.11 isUdpPacketFull()

```
bool isUdpPacketFull (
            void )
```

Checks if the UDP packet is full.

Returns

true if full, false otherwise

2.3.2.12 restartMeasurements()

```
void restartMeasurements (
            void )
```

Clear sensor measurements keeping the last measuremet periods.

2.3.2.13 rewindTape()

```
void rewindTape (
            uint8_t i_tape)
```

Resets the Tape keeping the last period entry.

Parameters

i_tape	the Tape index

2.3.2.14 updatePeriod()

```
void updatePeriod (
            uint8_t i_tape,
            uint32_t value)
```

Function to add a new period to the Periods array of a given Tape. During operation first add the period, then measurement. The duplicate value entries will be not added. The entries withouth measurements will be overwritten. On overflow it will reset the Periods array. Corrects also the Columns_count if Columns_count >= MAX_COLUMNS↩ _COUNT.

Parameters

i_tape	the Tape index
value	the period value

2.4 lmt_sdk_api.h File Reference

Functions

- void lmtInit (void)
  Initializes the SDK.
- void setup (void)
  User application setup code.
- void loop (void)
  User application main control loop.

2.4.1 Function Documentation

2.4.1.1 lmtInit()

```
void lmtInit (
             void )
```

Initializes the SDK.

2.4.1.2 loop()

```
void loop (
             void )
```

User application main control loop.

2.4.1.3 setup()

```
void setup (
             void )
```

User application setup code.

2.5 lmt_settings.h File Reference

Enumerations

- enum LogLevel { LOG_ERRORS = 0 , LOG_WARNINGS , LOG_INFORMATIVE }
  Log level options for system logging.
- enum ActiveSim { ACTIVATE_ESIM = 0 , ACTIVATE_SIM = 1 }
  Enum for active SIM control.

Functions

- void printCoap (void)

  Prints coap structure.
- int scanForCoapKeys (const uint8_t *p_coap_data, unsigned int size)

  Scans a buffer for COAP key/value pairs stored as C strings (key\0value\0).
- void setCoapServerPort (uint16_t port)

  Set the COAP server port number.
- void setCoapTxResource (const char *resource)

  Set the COAP transmission resource path.
- void setCoapTxFileResource (const char *resource)

  Set the COAP file transmission resource path.
- void setCoapTxFwResource (const char *resource)

  Set the COAP firmware transmission resource path.
- void setCoapServerHostname (const char *hostname)

  Set the hostname of the main COAPS server.
- uint16_t getCoapServerPort (void)

  Get the currently configured COAP server port.
- const char * getCoapTxResource (void)

  Get the configured COAP transmission resource path.
- const char * getCoapTxFileResource (void)

  Get the configured COAP file transmission resource path.
- const char * getCoapTxFwResource (void)

  Get the configured COAP firmware transmission resource path.
- void getCoapDeviceName (char *device_name, unsigned *len)

  Get the configured device name used in COAP.
- const char * getCoapServerHostname (void)

  Get the configured main COAPS server hostname.
- int setLogFileMaxSize (int32_t size)

  Set the maximum size of the log file.
- int setLTEConnectionTimeout (uint16_t timeout)

  Set network connection timeout.
- int setUplinkTimeout (uint16_t timeout)

  Set the device uplink timeout.
- int setNoPsmUplinkTimeout (uint16_t timeout)

  Set the device uplink timeout when PSM not available.
- int setResendPacketInitialTimeout (uint8_t timeout)

  Set the initial timeout for resending packets.
- int setMaxResendTimeout (uint8_t timeout)

  Set the maximum timeout for resending packets.
- int setMaxResendAttempts (uint8_t attempts)

  Set the maximum number of resend attempts.
- int setLogRotationFrequency (uint8_t frequency)

  Set the frequency of log rotation checks.
- int setResponseWaitTimeout (uint8_t timeout)

  Set the CoAP response wait timeout.
- int setFileUlRetries (uint8_t retries)

  Set the number of retries for file uploads.
- int setNumOfLogFiles (uint8_t file_count)

  Set the maximum number of log files on flash before starting file rotation.
- int setLogLevel (LogLevel level)

Set the log level for the application.

- int setActiveSim (ActiveSim sim_source)

  Sets the current SIM source.
- int32_t getLogFileMaxSize (void)

  Get the maximum size of the log file.
- uint16_t getLTEConnectionTimeout (void)

  Get network connection timeout.
- uint16_t getUplinkTimeout (void)

  Get the device uplink timeout.
- uint16_t getNoPsmUplinkTimeout (void)

  Get the device uplink timeout when no PSM available.
- uint8_t getResendPacketInitialTimeout (void)

  Get the initial timeout for resending packets.
- uint8_t getMaxResendTimeout (void)

  Get the maximum timeout for resending packets.
- uint8_t getMaxResendAttempts (void)

  Get the maximum number of resend attempts.
- uint8_t getLogRotationFrequency (void)

  Get the frequency of log rotation checks.
- uint8_t getResponseWaitTimeout (void)

  Get the CoAP response wait timeout.
- uint8_t getFileUlRetries (void)

  Get the number of retries for file uploads.
- uint8_t getNumOfLogFiles (void)

  Get the maximum number of log file count in flash.
- LogLevel getLogLevel (void)

  Get the log level for the application.
- int getActiveSim (ActiveSim ∗sim_source)

  Returns the current SIM source.
- uint8_t disableSerialLog (void)

  Disable serial logging output by suspending the UART device.
- uint8_t enableSerialLog (void)

  Enable serial logging output by resuming the UART device.

2.5.1 Enumeration Type Documentation

2.5.1.1 ActiveSim

```
enum ActiveSim
```

Enum for active SIM control.

Enumerator

ACTIVATE_ESIM	
ACTIVATE_SIM	

```
00023 {
00024     ACTIVATE_ESIM = 0,
00025     ACTIVATE_SIM  = 1
00026 } ActiveSim;
```

2.5.1.2 LogLevel

```
enum LogLevel
```

Log level options for system logging.

This enum defines the available log levels for controlling the verbosity of saved system logs.

Enumerator

LOG_ERRORS	
LOG_WARNINGS	
LOG_INFORMATIVE	

```
00013 {
00014     LOG_ERRORS = 0,
00015     LOG_WARNINGS,
00016     LOG_INFORMATIVE
00017 } LogLevel;
```

2.5.2 Function Documentation

2.5.2.1 disableSerialLog()

```
uint8_t disableSerialLog (
            void )
```

Disable serial logging output by suspending the UART device.

Suspends the UART device using power management to disable serial output for logging and debugging. This can help reduce power consumption when serial output is not needed.

Returns

0 on success, negative error code on failure. Returns -EALREADY if the device is already suspended (not treated as error).

2.5.2.2 enableSerialLog()

```
uint8_t enableSerialLog (
            void )
```

Enable serial logging output by resuming the UART device.

Resumes the UART device using power management to enable serial output for logging and debugging. This allows real-time monitoring of device operation after the UART was previously suspended.

Returns

0 on success, negative error code on failure. Returns -EALREADY if the device is already active (not treated as error).

2.5.2.3 getActiveSim()

```
int getActiveSim (
            ActiveSim * sim_source)
```

Returns the current SIM source.

Parameters

sim_source	Pointer to the source of the SIM

Returns

0 on success, negative error code on failure.

2.5.2.4 getCoapDeviceName()

```
void getCoapDeviceName (
            char * device_name,
            unsigned * len)
```

Get the configured device name used in COAP.

Parameters

device_name	Pointer to a buffer where the device name will be stored.
len	Length pointer holds in/out buffer size.

2.5.2.5 getCoapServerHostname()

```
const char * getCoapServerHostname (
            void )
```

Get the configured main COAPS server hostname.

Returns

Null-terminated string with the server hostname.

2.5.2.6 getCoapServerPort()

```
uint16_t getCoapServerPort (
            void )
```

Get the currently configured COAP server port.

Returns

Server port as uint16_t.

2.5.2.7 getCoapTxFileResource()

```
const char * getCoapTxFileResource (
             void )
```

Get the configured COAP file transmission resource path.

Returns

Null-terminated string representing the file resource path.

2.5.2.8 getCoapTxFwResource()

```
const char * getCoapTxFwResource (
             void )
```

Get the configured COAP firmware transmission resource path.

Returns

Null-terminated string representing the firmware resource path.

2.5.2.9 getCoapTxResource()

```
const char * getCoapTxResource (
             void )
```

Get the configured COAP transmission resource path.

Returns

Null-terminated string representing the resource path.

2.5.2.10 getFileUlRetries()

```
uint8_t getFileUlRetries (
             void )
```

Get the number of retries for file uploads.

Returns

Number of retries for file uploads.

2.5.2.11 getLogFileMaxSize()

```
int32_t getLogFileMaxSize (
            void )
```

Get the maximum size of the log file.

Returns

Maximum size of the log file in bytes.

2.5.2.12 getLogLevel()

```
LogLevel getLogLevel (
            void )
```

Get the log level for the application.

Returns

Log level for the application.

2.5.2.13 getLogRotationFrequency()

```
uint8_t getLogRotationFrequency (
            void )
```

Get the frequency of log rotation checks.

Returns

Frequency of log rotation checks in wakeup cycles.

2.5.2.14 getLTEConnectionTimeout()

```
uint16_t getLTEConnectionTimeout (
            void )
```

Get network connection timeout.

Returns

Device network connection timeout in seconds.

2.5.2.15 getMaxResendAttempts()

```
uint8_t getMaxResendAttempts (
            void )
```

Get the maximum number of resend attempts.

Returns

Maximum number of resend attempts before the socket is closed.

2.5.2.16 getMaxResendTimeout()

```
uint8_t getMaxResendTimeout (
            void )
```

Get the maximum timeout for resending packets.

Returns

Maximum timeout for resending packets in hours.

2.5.2.17 getNoPsmUplinkTimeout()

```
uint16_t getNoPsmUplinkTimeout (
            void )
```

Get the device uplink timeout when no PSM available.

Returns

Device uplink timeout when PSM not available in hours.

2.5.2.18 getNumOfLogFiles()

```
uint8_t getNumOfLogFiles (
            void )
```

Get the maximum number of log file count in flash.

Returns

Number of maximum count of log files stored in flash.

2.5.2.19 getResendPacketInitialTimeout()

```
uint8_t getResendPacketInitialTimeout (
            void )
```

Get the initial timeout for resending packets.

Returns

Initial timeout for resending packets in minutes.

2.5.2.20 getResponseWaitTimeout()

```
uint8_t getResponseWaitTimeout (
            void )
```

Get the CoAP response wait timeout.

Returns

Timeout for waiting for CoAP response in seconds.

2.5.2.21 getUplinkTimeout()

```
uint16_t getUplinkTimeout (
            void )
```

Get the device uplink timeout.

Returns

Device uplink timeout in minutes.

2.5.2.22 printCoap()

```
void printCoap (
            void )
```

Prints coap structure.

2.5.2.23 scanForCoapKeys()

```
int scanForCoapKeys (
            const uint8_t * p_coap_data,
            unsigned int size)
```

Scans a buffer for COAP key/value pairs stored as C strings (key\0value\0).

Parameters

p_coap_data	a Pointer
size	Size to scan.

Returns

0 on success, negative error code on failure.

2.5.2.24 setActiveSim()

```
int setActiveSim (
            ActiveSim sim_source)
```

Sets the current SIM source.

Parameters

sim_source	Source of the SIM

Returns

0 on success, negative error code on failure.

2.5.2.25 setCoapServerHostname()

```
void setCoapServerHostname (
            const char * hostname)
```

Set the hostname of the main COAPS server.

Parameters

hostname	Null-terminated string with the server's hostname or IP address.

2.5.2.26 setCoapServerPort()

```
void setCoapServerPort (
            uint16_t port)
```

Set the COAP server port number.

Parameters

port	COAP server port (typically 5683 for COAP, 5684 for COAPS).

2.5.2.27 setCoapTxFileResource()

```
void setCoapTxFileResource (
            const char * resource)
```

Set the COAP file transmission resource path.

Parameters

resource	Null-terminated string for the file resource (e.g. "upload").

2.5.2.28 setCoapTxFwResource()

```
void setCoapTxFwResource (
            const char * resource)
```

Set the COAP firmware transmission resource path.

Parameters

resource	Null-terminated string for the firmware update resource.

2.5.2.29 setCoapTxResource()

```
void setCoapTxResource (
            const char * resource)
```

Set the COAP transmission resource path.

Parameters

resource	Null-terminated string representing the resource path (e.g. "sensor/data").

2.5.2.30 setFileUlRetries()

```
int setFileUlRetries (
            uint8_t retries)
```

Set the number of retries for file uploads.

Parameters

retries	Number of retries for file uploads. 1 <= retries <= 10.

Returns

0 on success, -EINVAL otherwise.

2.5.2.31 setLogFileMaxSize()

```
int setLogFileMaxSize (
            int32_t size)
```

Set the maximum size of the log file.

Parameters

size	Maximum size of the log file in bytes. 1024 (KB) <= size <= 1048576 (1MB).

Returns

0 on success, -EINVAL otherwise.

2.5.2.32 setLogLevel()

```
int setLogLevel (
            LogLevel level)
```

Set the log level for the application.

Parameters

level	Log level for the application. LOG_ERRORS, LOG_WARNINGS, LOG_INFORMATIVE.

Returns

0 on success, -EINVAL otherwise.

2.5.2.33 setLogRotationFrequency()

```
int setLogRotationFrequency (
            uint8_t frequency)
```

Set the frequency of log rotation checks.

Parameters

frequency	Frequency of log rotation checks in wakeup cycles. 1 <= frequency <= 50.

Returns

0 on success, -EINVAL otherwise.

2.5.2.34 setLTEConnectionTimeout()

```
int setLTEConnectionTimeout (
            uint16_t timeout)
```

Set network connection timeout.

Parameters

timeout	Device network connection timeout in seconds. 1 (sec) <= timeout <= 1800 (30 min).

Returns

0 on success, -EINVAL otherwise.

2.5.2.35 setMaxResendAttempts()

```
int setMaxResendAttempts (
            uint8_t attempts)
```

Set the maximum number of resend attempts.

Parameters

attempts	Maximum number of resend attempts before the socket is closed. 1 <= attempts <= 10.

Returns

0 on success, -EINVAL otherwise.

2.5.2.36 setMaxResendTimeout()

```
int setMaxResendTimeout (
            uint8_t timeout)
```

Set the maximum timeout for resending packets.

Parameters

timeout	Maximum timeout for resending packets in hours. 1 (h) <= timeout <= 24 (1d).

Returns

0 on success, -EINVAL otherwise.

2.5.2.37 setNoPsmUplinkTimeout()

```
int setNoPsmUplinkTimeout (
            uint16_t timeout)
```

Set the device uplink timeout when PSM not available.

Parameters

timeout	Device uplink timout in hours 1 <= timeout <= 24 (hours).

Returns

0 on success, -EINVAL otherwise.

2.5.2.38 setNumOfLogFiles()

```
int setNumOfLogFiles (
            uint8_t file_count)
```

Set the maximum number of log files on flash before starting file rotation.

Parameters

file_count	Number of log files on flash. 1 <= file_count <= 20.

Returns

0 on success, -EINVAL otherwise.

2.5.2.39 setResendPacketInitialTimeout()

```
int setResendPacketInitialTimeout (
            uint8_t timeout)
```

Set the initial timeout for resending packets.

Parameters

timeout	Initial timeout for resending packets in minutes. 1 (min) <= timeout <= 60 (1h).

Returns

0 on success, -EINVAL otherwise.

2.5.2.40 setResponseWaitTimeout()

```
int setResponseWaitTimeout (
            uint8_t timeout)
```

Set the CoAP response wait timeout.

Parameters

timeout	Timeout for waiting for CoAP response in seconds. 1 <= timeout <= 60.

Returns

0 on success, -EINVAL otherwise.

2.5.2.41 setUplinkTimeout()

```
int setUplinkTimeout (
            uint16_t timeout)
```

Set the device uplink timeout.

Parameters

timeout	Device uplink timeout in minutes. 5 (min) <= timeout <= 1440 (24h).

Returns

0 on success, -EINVAL otherwise.

2.6 lmt_som_event_emitter.h File Reference

Typedefs

- typedef void(∗ EventHandler) (void ∗p_data, int i_data)

Enumerations

- enum SomEvent {
  EVENT_DEVICE_INIT_OK,
  EVENT_LOGGER_INIT_OK,
  EVENT_PACKER_INIT_OK,
  EVENT_MAILER_INIT_OK,
  EVENT_DROPPING_OLDEST,
  EVENT_PACKER_STARTED,
  EVENT_PACKING_FAILED, EVENT_ENQUEUE_FAILED,
  EVENT_PACKER_DONE_OK,
  EVENT_UL_START,
  EVENT_UL_MAX_RETRY,
  EVENT_UL_RETRY,
  EVENT_UL_DONE,
  EVENT_COAP_START,
  EVENT_COAP_FAIL, EVENT_COAP_NOACK,
  EVENT_COAP_OK,
  EVENT_LOG_ERROR,
  EVENT_LOG_WARNING,
  EVENT_LOG_INFO,
  EVENT_TERMINAL_CMD,
  EVENT_COUNT
  }

  Events emitted by the LMT SOM library.

Functions

- void handleSomEvent (SomEvent event, void *p_data, int i_data)

  Centralized event handler for the library. This is a weak function that can be overridden by the user application.

2.6.1 Typedef Documentation

2.6.1.1 EventHandler

```
typedef void(* EventHandler) (void *p_data, int i_data)
```

2.6.2 Enumeration Type Documentation

2.6.2.1 SomEvent

```
enum SomEvent
```

Events emitted by the LMT SOM library.

This enumeration defines all possible events that can be emitted by the library to notify the application about various system and application-level occurrences, such as initialization, uplink status, logging, and terminal commands.

Enumerator

EVENT_DEVICE_INIT_OK	Device initialization completed successfully.
EVENT_LOGGER_INIT_OK	Logger module initialized successfully.
EVENT_PACKER_INIT_OK	Packer module initialized successfully.
EVENT_MAILER_INIT_OK	Mailer module initialized successfully.
EVENT_DROPPING_OLDEST	Uplink queue is full, the oldest message(s) are dropped.
EVENT_PACKER_STARTED	Packer started.
EVENT_PACKING_FAILED	Data packing failed.
EVENT_ENQUEUE_FAILED	Packer failed to enqueue packet.
EVENT_PACKER_DONE_OK	Package created and enqueued successfully.
EVENT_UL_START	Uplink process started.
EVENT_UL_MAX_RETRY	Uplink retry limit reached.
EVENT_UL_RETRY	Uplink retry is being attempted.
EVENT_UL_DONE	Uplink completed.
EVENT_COAP_START	COAP packet uplink started.
EVENT_COAP_FAIL	COAP packet uplink failed.
EVENT_COAP_NOACK	No ACK received for the COAP packet.
EVENT_COAP_OK	COAP packet uplink completed successfully.
EVENT_LOG_ERROR	An error log event occurred.
EVENT_LOG_WARNING	A warning log event occurred.
EVENT_LOG_INFO	An informational log event occurred.
EVENT_TERMINAL_CMD	A terminal command event occurred.
EVENT_COUNT	Event count.

```
00014 {
00015     EVENT_DEVICE_INIT_OK,
00016     EVENT_LOGGER_INIT_OK,
00017     EVENT_PACKER_INIT_OK,
00018     EVENT_MAILER_INIT_OK,
00019
00020     EVENT_DROPPING_OLDEST,
00021
00022     EVENT_PACKER_STARTED,
00023     EVENT_PACKING_FAILED,
00024     EVENT_ENQUEUE_FAILED,
00025     EVENT_PACKER_DONE_OK,
00026
00027     EVENT_UL_START,
00028     EVENT_UL_MAX_RETRY,
00029     EVENT_UL_RETRY,
00030     EVENT_UL_DONE,
00031
00032     EVENT_COAP_START,
00033     EVENT_COAP_FAIL,
00034     EVENT_COAP_NOACK,
00035     EVENT_COAP_OK,
00036
00037     EVENT_LOG_ERROR,
00038     EVENT_LOG_WARNING,
00039     EVENT_LOG_INFO,
00040     EVENT_TERMINAL_CMD,
00041     EVENT_COUNT
00042 } SomEvent;
```

2.6.3 Function Documentation

2.6.3.1 handleSomEvent()

```
void handleSomEvent (
            SomEvent event,
            void * p_data,
            int i_data)
```

Centralized event handler for the library. This is a weak function that can be overridden by the user application.

Parameters

event	The event type.
p_data	Optional data pointer associated with the event (can be NULL).
i_data	Optional integer data associated with the event if specified in the event description (e.g., length of the data).

2.7 lmt_storage_manager.h File Reference

Functions

- int logError (char ∗text, int code)

  Writes error to app.log file.
- int logWarning (char ∗text)

  Writes warning to app.log file.
- void logStringHex (uint8_t ∗payload, uint8_t message_length)

  Logs the given buffer as hex string.

- int logInfoFormatted (char ∗text,...)

  Variadic function for writing formatted info to app.log file.
- int logInfo (char ∗text)

  Function for writing informative string to app.log file. Creating two separate functions because most of the time we don't need to format the string.
- int saveSettings (void)

  Writes current device settings in JSON format to settings.txt file.
- void settingsFileRead (void)

  Schedules settings.txt read.
- int checkFwUpgradeStatus (void)

  Checks if firmware upgrade was successful or failed. Detects successful upgrades by checking if running unconfirmed image. Detects failed upgrades by checking for rejected images in secondary slot.
- int eraseFlash (void)

  Wrapper function for filesystem flashErase()

2.7.1 Function Documentation

2.7.1.1 checkFwUpgradeStatus()

```
int checkFwUpgradeStatus (
            void )
```

Checks if firmware upgrade was successful or failed. Detects successful upgrades by checking if running unconfirmed image. Detects failed upgrades by checking for rejected images in secondary slot.

Returns

1 if firmware upgrade was successful, 0 if no upgrade occurred, -1 if upgrade failed

2.7.1.2 eraseFlash()

```
int eraseFlash (
            void )
```

Wrapper function for filesystem flashErase()

Returns

0 on success, negative error code on fail

2.7.1.3 logError()

```
int logError (
            char * text,
            int code)
```

Writes error to app.log file.

Parameters

text	Error string.
code	Error code.

Returns

0 on success, negative value on fail

2.7.1.4 logInfo()

```
int logInfo (
            char * text)
```

Function for writing informative string to app.log file. Creating two separate functions because most of the time we don't need to format the string.

Parameters

text	info message string.

Returns

0 on success, negative value on fail

2.7.1.5 logInfoFormatted()

```
int logInfoFormatted (
            char * text,
             ...)
```

Variadic function for writing formatted info to app.log file.

Parameters

text	info message string.

Returns

0 on success, negative value on fail

2.7.1.6 logStringHex()

```
void logStringHex (
            uint8_t * payload,
            uint8_t message_length)
```

Logs the given buffer as hex string.

Parameters

payload	pointer to the buffer
message_length	buffer size

2.7.1.7 logWarning()

```
int logWarning (
            char * text)
```

Writes warning to app.log file.

Parameters

text	Warning string.

Returns

0 on success, negative value on fail

2.7.1.8 saveSettings()

```
int saveSettings (
            void )
```

Writes current device settings in JSON format to settings.txt file.

Returns

0 on success, negative value on fail

2.7.1.9 settingsFileRead()

```
void settingsFileRead (
            void )
```

Schedules settings.txt read.

Index