

LS Player MQTT API v1.2.4

Архивная версия MQTT API

1. Управление проигрыванием и получение статистики

Описывает MQTT API сервиса.

Сервис осуществляет проигрывание анимаций.

SUB **lm/player**

Принимает команды управления проигрыванием.

Play

Payload command format

```
{
  "cmd": 'play',
  "what_playing": Union['playlist', 'cue'],
  "entity": Union[int, str],
  "count": Optional[int],
  "priority": int,
}
```

Example

```
{
  "cmd": "play",
  "what_playing": "playlist",
  "entity": 19,
  "count": Null,
  "priority": 4,
}
```

- **cmd** - Название команды.
- **what_playing** - Тип сущности для воспроизведения. Принимает два значения "playlist" и "cue".
- **entity** - ID или наименование проигрываемой сущности.
- **count** - Опциональный параметр. Количество повторений проигрывания. Если не задан или значение равно Null то проигрывание продолжится до получения следующей команды с равным или более высоким приоритетом.
- **priority** - Приоритет команды. Значение от 1 до 100. Чем больше значение - тем выше приоритет. Команда с более низким приоритетом не может отменять команду с более высоким приоритетом. Текущие сопоставления приоритетов: Расписание - 60, Триггер - 50, Ручной запуск - 40.

Stop

Payload stop command format

```
{
  "cmd": 'stop',
  "priority": int,
}
```

Example

```
{
  "cmd": "stop",
  "priority": 4,
}
```

- **cmd** - Название команды.
- **priority** - Приоритет команды. Значение от 1 до 100. Чем больше значение - тем выше приоритет. Команда с более низким приоритетом не может отменять команду с более высоким приоритетом. Текущие сопоставления приоритетов: Расписание - 60, Триггер - 50, Ручной запуск - 40.

PUB **lm/statistic/playing_progress_info**

Публикует статистику проигрывания.

Зная текущее значение fps можно перевести значения во время.

Например при fps равном 40 frame_count равном 1000 и frame_number равном 120 мы получим:
 $1 / 40 * 1000 = 25$ - Общая продолжительность анимации в секундах. $1 / 40 * 120 = 3$ - На текущий момент анимация проиграла 3 секунды.

Payload format

Представляет из себя строку в формате "{frame_count}, {frame_number}"

Example

"1000, 35"

- **frame_count** - Общее количество фреймов.
- **frame_number** - Сколько фреймов проиграно на текущий момент.

PUB **lm/statistic/playing_ent_info**

Публикует Наименования того, что сейчас проигрывается.

Payload format

```
{
```

```
"playlist": Optional[str],
'scene': Optional[int],
'cue': Optional[str],
}
```

Example

```
{
  "playlist": "NewYearPlaylist",
  "scene": 1,
  "cue": "BLUE.cue",
}
```

- **playlist** - Наименование проигрываемого плейлиста. Может быть None.
- **scene** - Порядковый номер в плейлисте. Может быть None.
- **cue** - Наименование проигрываемой анимации. Может быть None.

PUB **lm/statistic/current_playing_priority**

Публикует текущий приоритет проигрывания.

Payload format

int

Example

60

2. Управление настройками проигрывания и сущностей

PUB **lm/settings/location/coordinates**

Публикует координаты плеера.

Payload command format

```
{
  "latitude": float,
  "longitude": float,
}
```

Example

```
{
  "latitude": "56.821019190097616",
  "longitude": "60.59559633825789"
```

```
}
```

PUB **lm/settings/location/address**

Публикует адрес устройства.

Payload format

```
{  
  "address": str  
}
```

Example

```
{  
  "address": "Yekaterinburg"  
}
```

PUB **lm/settings/datetime/timezone**

Публикует часовой пояс плеера.

Payload format

```
{  
  "timezone": str  
}
```

Example

```
{  
  "timezone": "Asia/Yekaterinburg"  
}
```

- **timezone** - Часовой пояс плеера.

PUB **lm/settings/player/fps**

Публикует настройки fps.

Payload format

```
{  
  "fps": int,  
}
```

Example

```
{  
  "fps": 40  
}
```

PUB `lm/settings/player/artsync`

Публикует статус отправки artsync.

Payload format

```
{  
  "artsync": bool,  
}
```

Example

```
{"artsync": false}
```

PUB `lm/settings/player/blackout_between_playing_command`

Публикует настройку необходимости blackout между событиями проигрывания.

Payload format

```
{  
  "blackout_between_playing_command": bool,  
}
```

Example

```
{  
  "blackout_between_playing_command": false  
}
```

PUB `lm/settings/player/playing_priority`

Публикует приоритеты проигрывания плеера.

Payload command format

```
{  
  "buttons": int,  
  "triggers": int,  
  "scheduler": int,  
}
```

```
}
```

Example

```
{
  "buttons": 4,
  "triggers": 5,
  "scheduler": 6,
}
```

Приоритет представляет из себя целое число от 1 до 100. Чем выше число тем меньше приоритет.

PUB **lm/settings/player/universes**

Публикует настройки вселенных плеера.

Payload format

```
[
  {
    "number": int,
    "device": {
      "name": str,
      "description": str,
      "network_mode": str,
      "ip": str,
      "port": int,
    } | None
  }
]
```

Example

```
[
  {
    "number": 1,
    "device": {
      "name": "artnet_device_1",
      "description": "Main ArtNet converter",
      "network_mode": "unicast",
      "ip": "192.168.1.100",
      "port": 6454
    }
  },
  {
    "number": 2,
    "device": null
  }
]
```

- **number** - Номер вселенной (0-32768).
- **device** - Настройки ArtNet устройства для данной вселенной. Может быть null если устройство не назначено.
 - **name** - Уникальное имя ArtNet устройства (до 32 символов).
 - **description** - Описание устройства (до 255 символов, может быть пустым).
 - **network_mode** - Режим работы сети ("unicast" или "broadcast").
 - **ip** - IP адрес устройства.
 - **port** - Порт устройства (по умолчанию 6454, диапазон 1-65534).

PUB **lm/cues**

Публикует список cue файлов загруженных на плеер

Payload format

```
[
  {
    "id": int,
    "filename": str,
    "uni_count": int,
    "frame_count": int,
    "created": str,
  }
]
```

Example

```
[
  {
    "id": 47,
    "filename": "00-5.cue",
    "uni_count": 1,
    "frame_count": 220,
    "created": "2024-03-07T08:30:16.926447Z"
  }
]
```

- **id** - Уникальный идентификатор анимации.
- **filename** - Имя файла.
- **uni_count** - Количество вселенных в файле.
- **frame_count** - Количество фреймов в файле.
- **created** - Время загрузки анимации в ISO формате.

PUB **lm/playlists**

Публикует список cue файлов загруженных на плеер

Payload format

```
[
  {
    "id": int,
    "name": str,
    "scenes": [
      {
        "id": int,
        "order": int,
        "cue": {
          "created": str,
          "filename": str,
          "frame_count": int,
          "id": int,
          "uni_count": int
        },
        "fade_in": float,
        "fade_out": float,
        "transition_time": float,
        "repeat_value": int,
      }
    ]
  }
]
```

Example

```
[
  {
    "id": 19,
    "name": "Test",
    "scenes": [
      {
        "id": 71,
        "order": 0,
        "cue": {
          "created": "2024-03-07T08:27:23.567083Z",
          "filename": "5-8.cue",
          "frame_count": 220,
          "id": 51,
          "uni_count": 1
        },
        "fade_in": 1.0,
        "fade_out": 0.0,
        "transition_time": 2.0,
        "repeat_value": 3600
      }
    ]
  }
]
```

- **id** - Уникальный идентификатор плейлиста.
- **name** - Название плейлиста.

- **scenes** - Сцены. В сценах содержится вся информация об эффектах примененных к сцене и порядковый номер воспроизведения внутри плейлиста.
 - **id** - Уникальный идентификатор сцены.
 - **order** - Порядковый номер воспроизведения внутри плейлиста.
 - **cue** - Параметры анимации. [Подробнее](#)
 - **fade_in** - Время fade_in.
 - **fade_out** - Время fade_out.
 - **transition_time** - Время перехода.
 - **repeat_value** - Количество повторений.

3. Управление расписанием

PUB **lm/scheduler/error**

Публикует ошибки.

Выставляет заголовок **Correlation data** если он был установлен в запросе.

Payload format

```
{  
  msg: str  
  data: Any  
}
```

- **msg** - contain error message
- **data** - contain related error data

PUB **lm/scheduler/events**

Публикует список всех событий календаря.

Payload format

```
[  
  {  
    "id": str,  
    "title": str,  
    "priority": int,  
    "actions": {  
      "player": Optional[{  
        "cmd": Literal['play'],  
        "entity_type": Union['playlist', 'cue'],  
        "entity_id": int,  
      }],  
    },  
  },  
]
```

```

    "do1": Optional[{
      "state": Literal[0, 1],
    }],
    "do2": Optional[{
      "state": Literal[0, 1],
    }],
    "do3": Optional[{
      "state": Literal[0, 1],
    }],
  },
  "rrule": {
    "freq": Union['YEARLY', 'MONTHLY', 'WEEKLY', 'DAILY', 'HOURLY'],
    "interval": int,
    "start_date": str,
    "start_time_type": Union['sunset', 'sunrise', 'time'],
    "start_time": Optional[str],
    "start_time_offset": Optional[int],
    "until_date": Optional[str],
    "until_time_type": Optional[Union['sunset', 'sunrise', 'time']],
    "until_time": Optional[str],
    "until_time_offset": Optional[int],
    "count": Optional[int],
    "from_time_type": Optional[Union['sunset', 'sunrise', 'time']],
    "from_time": Optional[str],
    "from_time_offset": Optional[int],
    "to_time_type": Optional[Union['sunset', 'sunrise', 'time']],
    "to_time": Optional[str],
    "to_time_offset": Optional[int],
    "bymonth": Optional[
      list[
        Union[
          'January', 'February', 'March', 'April', 'May', 'June',
'July',
          'August', 'September', 'October', 'November', 'December',
        ],
      ],
    ],
    "bymonthday": Optional[list[int]],
    "byweekday": Optional[list[Union['MO', 'TU', 'WE', 'TH', 'FR', 'SA',
'SU']]],
    "from_min": Optional[int],
    "to_min": Optional[int],
  }
}
]

```

Example

```

[
  {
    "id": "abe4c633-8e3f-4938-94e2-efd135d993fc",

```

```

"title": "holiday",
"priority": 1,
"actions": {
  "player": {
    "cmd": "play",
    "entity_type": "playlist",
    "entity_id": 19
  },
  "do1": {
    "state": 1
  },
  "do2": null,
  "do3": null
},
"rrule": {
  "freq": "DAILY",
  "interval": 1,
  "start_date": "2024-01-20",
  "start_time_type": "time",
  "start_time": "00:00",
  "start_time_offset": null,
  "count": 1,
  "until_date": null,
  "until_time_type": null,
  "until_time": null,
  "until_time_offset": null,
  "from_time_type": "sunset",
  "from_time": null,
  "from_time_offset": 0,
  "to_time_type": "sunset",
  "to_time": null,
  "to_time_offset": 0,
  "bymonth": null,
  "bymonthday": null,
  "byweekday": null,
  "from_min": null,
  "to_min": null
}
}
]

```

- **id** - Уникальный идентификатор события (UUID).
- **title** - Название события.
- **priority** - Приоритет события. Чем выше значение тем выше приоритет.
- **actions** - Действия которые должны быть выполнены при наступлении события.
- **player** - Действие для плеера. Содержит команду воспроизведения.
- **cmd** - Команда для плеера. Всегда равна 'play'.
- **entity_type** - Тип сущности для воспроизведения. Может принимать значения 'playlist', 'cue'.
- **entity_id** - Уникальный идентификатор сущности для воспроизведения.
- **do1** - Действие для цифрового выхода DO1.
- **do2** - Действие для цифрового выхода DO2.
- **do3** - Действие для цифрового выхода DO3.

- **state** - Состояние цифрового выхода. Может принимать значения 0 (выключен) или 1 (включен).
- **rrule** - Правила повторения события (recurrence rule).
- **freq** - Частота повторений события. Может принимать значения: 'YEARLY', 'MONTHLY', 'WEEKLY', 'DAILY', 'HOURLY'.
- **interval** - Периодичность повторения события.
- **start_date** - Дата старта события. Формат YYYY-mm-dd.
- **start_time_type** - Тип времени старта события. Может принимать значения: 'sunset', 'sunrise', 'time'.
- **start_time** - Время старта события. Формат: %H:%M. Заполнено если start_time_type равен 'time'.
- **start_time_offset** - Сдвиг времени старта события. Может принимать отрицательные значения. Заполнено если start_time_type равен 'sunset' или 'sunrise'.
- **count** - Количество повторений события. Не может быть заполнен одновременно с полем until_date. Если оба поля не заполнены то событие не никогда не завершается.
- **until_date** - Дата завершения события. Формат YYYY-mm-dd. Не может быть заполнен одновременно с полем count. Если оба поля не заполнены то событие не никогда не завершается.
- **until_time_type** - Тип времени завершения события. Может принимать значения: 'sunset', 'sunrise', 'time'. Заполнено если заполнено поле until_date.
- **until_time** - Время завершения события. Формат: %H:%M. Заполнено если заполнено поле until_date и until_time_type равен 'time'.
- **until_time_offset** - Сдвиг времени завершения события. Заполнено если заполнено поле until_date и until_time_type равен 'sunset' или 'sunrise'.
- **from_time_type** - Тип времени начала события. Может принимать значения: 'sunset', 'sunrise', 'time'. Заполнено если поле freq не равно 'HOURLY'.
- **from_time** - Время начала события. Формат: %H:%M. Заполнено если поле freq не равно 'HOURLY' и from_time_type равен 'time'.
- **from_time_offset** - Сдвиг времени начала события. Может принимать отрицательные значения. Заполнено если поле freq не равно 'HOURLY' и from_time_type равен 'sunset' или 'sunrise'.
- **to_time_type** - Тип времени окончания события. Может принимать значения: 'sunset', 'sunrise', 'time'. Заполнено если поле freq не равно 'HOURLY'.
- **to_time** - Время окончания события. Формат: %H:%M. Заполнено если заполнено поле freq не равно 'HOURLY' и to_time_type равен 'time'.
- **to_time_offset** - Сдвиг времени завершения события. Заполнено если заполнено поле freq не равно 'HOURLY' и to_time_type равен 'sunset' или 'sunrise'.
- **bymonth** - Месяцы в которые событие активно. Заполнено если поле freq равно 'YEARLY'.
- **bymonthday** - Дни месяца в которые событие активно. Заполнено если поле freq равно 'MONTHLY'.
- **byweekday** - Дни недели в которые событие активно. Заполнено если поле freq равно 'WEEKLY'.
- **from_min** - Минута с которой начинается событие. Заполнено если поле freq равно 'HOURLY'.
- **to_min** - Минута окончания события. Заполнено если поле freq равно 'HOURLY'.

SUB `lm/scheduler/events/add`

Добавляет новое событие.

Payload format

```
{
  "title": str,
  "priority": int,
  "actions": {
    "player": Optional[{
      "cmd": Literal['play'],
      "entity_type": Union['playlist', 'cue'],
```

```

        "entity_id": int,
    }],
    "do1": Optional[{
        "state": Literal[0, 1],
    }],
    "do2": Optional[{
        "state": Literal[0, 1],
    }],
    "do3": Optional[{
        "state": Literal[0, 1],
    }],
},
"rrule": {
    "freq": Union['YEARLY', 'MONTHLY', 'WEEKLY', 'DAILY', 'HOURLY'],
    "interval": int,
    "start_date": str,
    "start_time_type": Union['sunset', 'sunrise', 'time'],
    "start_time": Optional[str],
    "start_time_offset": Optional[int],
    "until_date": Optional[str],
    "until_time_type": Optional[Union['sunset', 'sunrise', 'time']],
    "until_time": Optional[str],
    "until_time_offset": Optional[int],
    "count": Optional[int],
    "from_time_type": Optional[Union['sunset', 'sunrise', 'time']],
    "from_time": Optional[str],
    "from_time_offset": Optional[int],
    "to_time_type": Optional[Union['sunset', 'sunrise', 'time']],
    "to_time": Optional[str],
    "to_time_offset": Optional[int],
    "bymonth": Optional[
        list[
            Union[
                'January', 'February', 'March', 'April', 'May', 'June',
'July',
                'August', 'September', 'October', 'November', 'December',
            ],
        ],
    ],
    "bymonthday": Optional[list[int]],
    "byweekday": Optional[list[Union['MO', 'TU', 'WE', 'TH', 'FR', 'SA',
'SU']]],
    "from_min": Optional[int],
    "to_min": Optional[int],
}
}

```

Example

```

{
    "title": "holiday",

```

```

"priority": 1,
"actions": {
  "player": {
    "cmd": "play",
    "entity_type": "playlist",
    "entity_id": 19
  },
  "do1": {
    "state": 1
  },
  "do2": null,
  "do3": null
},
"rrule": {
  "freq": "DAILY",
  "interval": 1,
  "start_date": "2024-01-20",
  "start_time_type": "time",
  "start_time": "00:00",
  "start_time_offset": null,
  "count": 1,
  "until_date": null,
  "until_time_type": null,
  "until_time": null,
  "until_time_offset": null,
  "from_time_type": "sunset",
  "from_time": null,
  "from_time_offset": 0,
  "to_time_type": "sunset",
  "to_time": null,
  "to_time_offset": 0,
  "bymonth": null,
  "bymonthday": null,
  "byweekday": null,
  "from_min": null,
  "to_min": null
}
}

```

- **title** - Название события.
- **priority** - Приоритет события. Чем выше значение тем выше приоритет.
- **actions** - Действия которые должны быть выполнены при наступлении события.
- **player** - Действие для плеера. Содержит команду воспроизведения.
- **cmd** - Команда для плеера. Всегда равна 'play'.
- **entity_type** - Тип сущности для воспроизведения. Может принимать значения 'playlist', 'cue'.
- **entity_id** - Уникальный идентификатор сущности для воспроизведения.
- **do1** - Действие для цифрового выхода DO1.
- **do2** - Действие для цифрового выхода DO2.
- **do3** - Действие для цифрового выхода DO3.
- **state** - Состояние цифрового выхода. Может принимать значения 0 (выключен) или 1 (включен).
- **rrule** - Правила повторения события (recurrence rule).
- **freq** - Частота повторений события. Может принимать значения: 'YEARLY', 'MONTHLY', 'WEEKLY', 'DAILY',

'HOURLY'.

- **interval** - Периодичность повторения события.
- **start_date** - Дата старта события. Формат YYYY-mm-dd.
- **start_time_type** - Тип времени старта события. Может принимать значения: 'sunset', 'sunrise', 'time'.
- **start_time** - Время старта события. Формат: %H:%M. Заполнено если start_time_type равен 'time'.
- **start_time_offset** - Сдвиг времени старта события. Может принимать отрицательные значения. Заполнено если start_time_type равен 'sunset' или 'sunrise'.
- **count** - Количество повторений события. Не может быть заполнен одновременно с полем until_date. Если оба поля не заполнены то событие не никогда не завершается.
- **until_date** - Дата завершения события. Формат YYYY-mm-dd. Не может быть заполнен одновременно с полем count. Если оба поля не заполнены то событие не никогда не завершается.
- **until_time_type** - Тип времени завершения события. Может принимать значения: 'sunset', 'sunrise', 'time'. Заполнено если заполнено поле until_date.
- **until_time** - Время завершения события. Формат: %H:%M. Заполнено если заполнено поле until_date и until_time_type равен 'time'.
- **until_time_offset** - Сдвиг времени завершения события. Заполнено если заполнено поле until_date и until_time_type равен 'sunset' или 'sunrise'.
- **from_time_type** - Тип времени начала события. Может принимать значения: 'sunset', 'sunrise', 'time'. Заполнено если поле freq не равно 'HOURLY'.
- **from_time** - Время начала события. Формат: %H:%M. Заполнено если поле freq не равно 'HOURLY' и from_time_type равен 'time'.
- **from_time_offset** - Сдвиг времени начала события. Может принимать отрицательные значения. Заполнено если поле freq не равно 'HOURLY' и from_time_type равен 'sunset' или 'sunrise'.
- **to_time_type** - Тип времени окончания события. Может принимать значения: 'sunset', 'sunrise', 'time'. Заполнено если поле freq не равно 'HOURLY'.
- **to_time** - Время окончания события. Формат: %H:%M. Заполнено если заполнено поле freq не равно 'HOURLY' и to_time_type равен 'time'.
- **to_time_offset** - Сдвиг времени завершения события. Заполнено если заполнено поле freq не равно 'HOURLY' и to_time_type равен 'sunset' или 'sunrise'.
- **bymonth** - Месяцы в которые событие активно. Заполнено если поле freq равно 'YEARLY'.
- **bymonthday** - Дни месяца в которые событие активно. Заполнено если поле freq равно 'MONTHLY'.
- **byweekday** - Дни недели в которые событие активно. Заполнено если поле freq равно 'WEEKLY'.
- **from_min** - Минута с которой начинается событие. Заполнено если поле freq равно 'HOURLY'.
- **to_min** - Минута окончания события. Заполнено если поле freq равно 'HOURLY'.

SUB **lm/scheduler/events/delete**

Удаляет событие.

Payload format

```
{  
  id: str  
}
```

Example

```
{  
  "id": "abe4c633-8e3f-4938-94e2-efd135d993fc",  
}
```

- **id** - Уникальный идентификатор события. ____

SUB **lm/scheduler/events/update**

Обновляет параметры события.

Payload format

```
{
  "id": str,
  "title": str,
  "priority": int,
  "actions": {
    "player": Optional[{
      "cmd": Literal['play'],
      "entity_type": Union['playlist', 'cue'],
      "entity_id": int,
    }],
    "do1": Optional[{
      "state": Literal[0, 1],
    }],
    "do2": Optional[{
      "state": Literal[0, 1],
    }],
    "do3": Optional[{
      "state": Literal[0, 1],
    }],
  },
  "rrule": {
    "freq": Union['YEARLY', 'MONTHLY', 'WEEKLY', 'DAILY', 'HOURLY'],
    "interval": int,
    "start_date": str,
    "start_time_type": Union['sunset', 'sunrise', 'time'],
    "start_time": Optional[str],
    "start_time_offset": Optional[int],
    "until_date": Optional[str],
    "until_time_type": Optional[Union['sunset', 'sunrise', 'time']],
    "until_time": Optional[str],
    "until_time_offset": Optional[int],
    "count": Optional[int],
    "from_time_type": Optional[Union['sunset', 'sunrise', 'time']],
    "from_time": Optional[str],
    "from_time_offset": Optional[int],
    "to_time_type": Optional[Union['sunset', 'sunrise', 'time']],
    "to_time": Optional[str],
    "to_time_offset": Optional[int],
    "bymonth": Optional[
      list[
        Union[
          'January', 'February', 'March', 'April', 'May', 'June',
```

```

'July',
        'August', 'September', 'October', 'November', 'December',
    ],
],
],
    "bymonthday": Optional[list[int]],
    "byweekday": Optional[list[Union['MO', 'TU', 'WE', 'TH', 'FR', 'SA',
'SU']]],
    "from_min": Optional[int],
    "to_min": Optional[int],
}
}

```

Example

```

{
    "id": "abe4c633-8e3f-4938-94e2-efd135d993fc",
    "title": "holiday",
    "priority": 1,
    "actions": {
        "player": {
            "cmd": "play",
            "entity_type": "playlist",
            "entity_id": 19
        },
        "do1": {
            "state": 1
        },
        "do2": null,
        "do3": null
    },
    "rrule": {
        "freq": "DAILY",
        "interval": 1,
        "start_date": "2024-01-20",
        "start_time_type": "time",
        "start_time": "00:00",
        "start_time_offset": null,
        "count": 1,
        "until_date": null,
        "until_time_type": null,
        "until_time": null,
        "until_time_offset": null,
        "from_time_type": "sunset",
        "from_time": null,
        "from_time_offset": 0,
        "to_time_type": "sunset",
        "to_time": null,
        "to_time_offset": 0,
        "bymonth": null,
        "bymonthday": null,
    }
}

```

```

    "byweekday": null,
    "from_min": null,
    "to_min": null
}
}

```

- **id** - Уникальный идентификатор события (UUID).
- **title** - Название события.
- **priority** - Приоритет события. Чем выше значение тем выше приоритет.
- **actions** - Действия которые должны быть выполнены при наступлении события.
- **player** - Действие для плеера. Содержит команду воспроизведения.
- **cmd** - Команда для плеера. Всегда равна 'play'.
- **entity_type** - Тип сущности для воспроизведения. Может принимать значения 'playlist', 'cue'.
- **entity_id** - Уникальный идентификатор сущности для воспроизведения.
- **do1** - Действие для цифрового выхода DO1.
- **do2** - Действие для цифрового выхода DO2.
- **do3** - Действие для цифрового выхода DO3.
- **state** - Состояние цифрового выхода. Может принимать значения 0 (выключен) или 1 (включен).
- **rrule** - Правила повторения события (recurrence rule).
- **freq** - Частота повторений события. Может принимать значения: 'YEARLY', 'MONTHLY', 'WEEKLY', 'DAILY', 'HOURLY'.
- **interval** - Периодичность повторения события.
- **start_date** - Дата старта события. Формат YYYY-mm-dd.
- **start_time_type** - Тип времени старта события. Может принимать значения: 'sunset', 'sunrise', 'time'.
- **start_time** - Время старта события. Формат: %H:%M. Заполнено если start_time_type равен 'time'.
- **start_time_offset** - Сдвиг времени старта события. Может принимать отрицательные значения. Заполнено если start_time_type равен 'sunset' или 'sunrise'.
- **count** - Количество повторений события. Не может быть заполнен одновременно с полем until_date. Если оба поля не заполнены то событие не никогда не завершается.
- **until_date** - Дата завершения события. Формат YYYY-mm-dd. Не может быть заполнен одновременно с полем count. Если оба поля не заполнены то событие не никогда не завершается.
- **until_time_type** - Тип времени завершения события. Может принимать значения: 'sunset', 'sunrise', 'time'. Заполнено если заполнено поле until_date.
- **until_time** - Время завершения события. Формат: %H:%M. Заполнено если заполнено поле until_date и until_time_type равен 'time'.
- **until_time_offset** - Сдвиг времени завершения события. Заполнено если заполнено поле until_date и until_time_type равен 'sunset' или 'sunrise'.
- **from_time_type** - Тип времени начала события. Может принимать значения: 'sunset', 'sunrise', 'time'. Заполнено если поле freq не равно 'HOURLY'.
- **from_time** - Время начала события. Формат: %H:%M. Заполнено если поле freq не равно 'HOURLY' и from_time_type равен 'time'.
- **from_time_offset** - Сдвиг времени начала события. Может принимать отрицательные значения. Заполнено если поле freq не равно 'HOURLY' и from_time_type равен 'sunset' или 'sunrise'.
- **to_time_type** - Тип времени окончания события. Может принимать значения: 'sunset', 'sunrise', 'time'. Заполнено если поле freq не равно 'HOURLY'.
- **to_time** - Время окончания события. Формат: %H:%M. Заполнено если заполнено поле freq не равно 'HOURLY' и to_time_type равен 'time'.
- **to_time_offset** - Сдвиг времени завершения события. Заполнено если заполнено поле freq не равно 'HOURLY' и to_time_type равен 'sunset' или 'sunrise'.
- **bymonth** - Месяцы в которые событие активно. Заполнено если поле freq равно 'YEARLY'.
- **bymonthday** - Дни месяца в которые событие активно. Заполнено если поле freq равно 'MONTHLY'.
- **byweekday** - Дни недели в которые событие активно. Заполнено если поле freq равно 'WEEKLY'.

- **from_min** - Минута с которой начинается событие. Заполнено если поле freq равно 'HOURLY'.
- **to_min** - Минута окончания события. Заполнено если поле freq равно 'HOURLY'.

PUB `lm/scheduler/events/changes`

Публикует вновь созданные/измененные/удаленные события.

Payload format

```
{
  status: Literal['created', 'updated', 'deleted'],
  event: {
    "id": str,
    "title": str,
    "priority": int,
    "actions": {
      "player": Optional[{
        "cmd": Literal['play'],
        "entity_type": Union['playlist', 'cue'],
        "entity_id": int,
      }],
      "do1": Optional[{
        "state": Literal[0, 1],
      }],
      "do2": Optional[{
        "state": Literal[0, 1],
      }],
      "do3": Optional[{
        "state": Literal[0, 1],
      }],
    },
  },
  "rrule": {
    "freq": Union['YEARLY', 'MONTHLY', 'WEEKLY', 'DAILY', 'HOURLY'],
    "interval": int,
    "start_date": str,
    "start_time_type": Union['sunset', 'sunrise', 'time'],
    "start_time": Optional[str],
    "start_time_offset": Optional[int],
    "until_date": Optional[str],
    "until_time_type": Optional[Union['sunset', 'sunrise', 'time']],
    "until_time": Optional[str],
    "until_time_offset": Optional[int],
    "count": Optional[int],
    "from_time_type": Optional[Union['sunset', 'sunrise', 'time']],
    "from_time": Optional[str],
    "from_time_offset": Optional[int],
    "to_time_type": Optional[Union['sunset', 'sunrise', 'time']],
    "to_time": Optional[str],
    "to_time_offset": Optional[int],
  },
}
```

```

        "bymonth": Optional[
            list[
                Union[
                    'January', 'February', 'March', 'April', 'May', 'June',
'July',
                    'August', 'September', 'October', 'November',
'December',
                ],
            ],
        "bymonthday": Optional[list[int]],
        "byweekday": Optional[list[Union['MO', 'TU', 'WE', 'TH', 'FR', 'SA',
'SU']]],
        "from_min": Optional[int],
        "to_min": Optional[int],
    }
}

```

Example

```

{
  "status": "created",
  "event": {
    "id": "abe4c633-8e3f-4938-94e2-efd135d993fc",
    "title": "holiday",
    "priority": 1,
    "actions": {
      "player": {
        "cmd": "play",
        "entity_type": "playlist",
        "entity_id": 19
      },
      "do1": {
        "state": 1
      },
      "do2": null,
      "do3": null
    },
    "rrule": {
      "freq": "DAILY",
      "interval": 1,
      "start_date": "2024-01-20",
      "start_time_type": "time",
      "start_time": "00:00",
      "start_time_offset": null,
      "count": 1,
      "until_date": null,
      "until_time_type": null,
      "until_time": null,
      "until_time_offset": null,
    }
  }
}

```

```

    "from_time_type": "sunset",
    "from_time": null,
    "from_time_offset": 0,
    "to_time_type": "sunset",
    "to_time": null,
    "to_time_offset": 0,
    "bymonth": null,
    "bymonthday": null,
    "byweekday": null,
    "from_min": null,
    "to_min": null
  }
}
}

```

- **status** - Тип изменения. Может принимать значения 'created', 'updated', 'deleted'.
- **event** - Событие со всеми параметрами в формате SchedulerEvent. ____

SUB **lm/scheduler/events/periods**

Принимает запрос на публикацию всех одиночных событий за указанный период.

Запрос должен содержать cor data для последующей идентификации ответа. Запрос может содержать resp_topic. В противном случае ответ будет опубликован в топик

lm/scheduler/events/periods/response.

Payload format

```

{
  from_datetime: str,
  to_datetime: str,
  filters: Optional[{
    player: bool,
    do1: bool,
    do2: bool,
    do3: bool,
  }]
}

```

Example

```

{
  "from_datetime": "2024-02-25T05:00:00",
  "to_datetime": "2024-04-08T05:00:00",
  "filters": {
    "player": true,
    "do1": false,
    "do2": false,
    "do3": false
  }
}

```

```
}
```

- **from_datetime** - Дата и время начала диапазона в iso формате.
- **to_datetime** - Дата и время окончания диапазона в iso формате.
- **filters** - Опциональные фильтры для типов действий. Если не указаны, возвращаются события со всеми типами действий.
- **player** - Включать события с действиями плеера.
- **do1** - Включать события с действиями для цифрового выхода DO1.
- **do2** - Включать события с действиями для цифрового выхода DO2.
- **do3** - Включать события с действиями для цифрового выхода DO3.

PUB `lm/scheduler/events/periods/response`

Публикует список одиночных событий календаря за указанный период. Период задается в запросе. Запрос принимается на топик `lm/scheduler/events/periods`

Payload format

```
[
  {
    id: str
    title: str
    start: str
    end: str
    priority: int
    duration: float
  }
]
```

Example

```
[
  {
    "id": "abe4c633-8e3f-4938-94e2-efd135d993fc",
    "title": "holiday",
    "priority": 1,
    "start": "2024-02-29T12:00:00+03:00",
    "end": "2024-03-02T12:00:00+03:00",
    "duration": 259200.0
  }
]
```

- **id** - Уникальный идентификатор события.
- **title** - Название события.
- **priority** - Приоритет события. Чем выше значение тем выше приоритет.
- **start** - Дата и время начала события в ISO формате.
- **end** - Дата и время окончания события в ISO формате.
- **duration** - Продолжительность события в секундах.

PUB `lm/scheduler/player/status`

Публикует текущее активное событие плеера если оно есть.

Payload formatСобытие есть:

```
{
  status: Literal['running'],
  event: {
    id: str,
    title: str,
    action: {
      cmd: Literal['play']
      entity_type: Literal['playlist', 'cue']
      entity_id: int
    }
  }
}
```

Example

```
{
  "status": "running",
  "event": {
    "id": "abe4c633-8e3f-4938-94e2-efd135d993fc",
    "title": "holiday",
    "action": {
      "cmd": "play",
      "entity_type": "playlist",
      "entity_id": 19
    }
  }
}
```

События нет:

```
{
  status: Literal['no_event'],
}
```

Example

```
{
  "status": "no_event"
}
```

- **status** - Текущий статус расписания. Может принимать значения 'running', 'no_event'.
- **event** - Активное событие со всеми параметрами. Присутствует только когда status равен 'running'.
- **id** - Уникальный идентификатор события.
- **title** - Название события.

- **action** - Действие которое должно быть выполнено для данного события.
- **cmd** - Команда для выполнения. Всегда равна 'play'.
- **entity_type** - Тип сущности для воспроизведения. Может принимать значения 'playlist', 'cue'.
- **entity_id** - Уникальный идентификатор сущности для воспроизведения.

PUB **lm/scheduler/do/*/status**

Публикует текущее активное событие управления цифровым выходом DO1 если оно есть.

PUB **lm/scheduler/do/1/status** PUB **lm/scheduler/do/2/status** PUB
lm/scheduler/do/3/status

Payload format Событие есть:

```
{
  status: Literal['running'],
  event: {
    id: str,
    title: str,
    action: {
      state: Literal[0, 1]
    }
  }
}
```

Example

```
{
  "status": "running",
  "event": {
    "id": "abe4c633-8e3f-4938-94e2-efd135d993fc",
    "title": "holiday",
    "action": {
      "state": 1
    }
  }
}
```

События нет:

```
{
  status: Literal['no_event'],
}
```

Example

```
{
  "status": "no_event"
```

```
}
```

- **status** - Текущий статус расписания для DO1. Может принимать значения 'running', 'no_event'.
- **event** - Активное событие со всеми параметрами. Присутствует только когда status равен 'running'.
- **id** - Уникальный идентификатор события.
- **title** - Название события.
- **action** - Действие которое должно быть выполнено для данного события.
- **state** - Состояние цифрового выхода. Может принимать значения 0 (выключен) или 1 (включен).

SUB **lm/settings/datetime/timezone**

Получает текущую таймзону.

Payload format

```
{  
    timezone: str  
}
```

Example

```
{  
    "timezone": "Europe/Moscow",  
}
```

SUB **lm/settings/location/coordinates**

Получает координаты устройства для расчета солнечного времени.

Payload format

```
{  
    latitude: float  
    longitude: float  
}
```

Example

```
{  
    latitude: 56.821019190097616  
    longitude: 60.59559633825789  
}
```

4 Управление устройствами Art-Net

Сервис осуществляет мониторинг и управления ArtNet и RDM устройствами.

PUB **lm/artnet_devices_management_service/error**

Публикует ошибки.

Выставляет заголовок **Correlation data** если он был установлен в запросе.

Payload format

```
{  
  msg: str  
  data: Any  
}
```

- **msg** - contain error message
- **data** - contain related error data

PUB **lm/artnet_devices_management_service/artnet/devices/changes**

Публикует вновь созданные/измененные/удаленные ArtNet устройства.

Payload format

```
{  
  status: Literal['created', 'updated', 'deleted']  
  device: {  
    mac_address: str  
    ip_address: str  
    subnet_mask: str  
    default_gateway: str  
    dhcp_status: bool  
    name: str  
    style: str  
    firmware_version: str  
    ports: dict[  
      int,  
      {  
        bind_index: int  
        is_input: bool  
        is_output: bool  
        port_type: Literal[  
          'DALI',  
          'ArtNet',  
          'ADB',  
          'Colortran_CMX',  
          'Avab',  
          'MIDI',  
          'DMX512',  
        ]  
      }  
    ]  
  }  
}
```

```

        name: str
        universe: int
        is_rdm_on: bool
        physical_port: Optional[int]
        out_signal: Optional[Literal['DMX', 'SPI']]
        is_data_transmitting: bool
    }
]
status: str
dev_mode: Optional[str]
spi_settings: Optional[
    {
        chip: str
        mode: str
        period: int
        time_high_0: int
        time_high_1: int
        time_reset: int
        gamma: int
        bit_mode: str
    }
]
dmx_settings: Optional[
    {
        break_time: int
        mab_time: int
        chan_time: int
        pause_time: int
        chan_num: int
    }
]
rdm_devices_count: int
}

```

PUB **lm/artnet_devices_management_service/rdm/devices/changes**

Публикует вновь созданные/измененные/удаленные RDM устройства.

Payload format

```

{
    status: Literal['created', 'updated', 'deleted']
    device: {
        uid: str
        art_net_device_mac: str
        art_net_device_ip: str
        port: int
        supported_params: dict[str, Any]
    }
}

```

```
}  
}
```

- **uid** - Уникальный идентификатор устройства.
- **art_net_device_mac** - Mac адрес ArtNet устройства к которому подключено данное rdm устройство.
- **art_net_device_ip** - IP адрес ArtNet устройства к которому подключено данное rdm устройство.
- **port** - Номер порта ArtNet устройства к которому подключено данное rdm устройство.
- **supported_params** - Словарь параметров и их значений.

PUB **lm/artnet_devices_management_service/cmd_response**

Публикует результаты выполнения асинхронных команд.

Используется для уведомления о завершении длительных операций, которые выполняются в фоновом режиме. Клиент получает **transaction_uid** при инициации команды и может отслеживать её статус через данный топик.

Payload format

```
{  
  "transaction_uid": "string",  
  "status": "string"  
}
```

- **transaction_uid** - Уникальный идентификатор транзакции, возвращаемый при инициации асинхронной команды
- **status** - Статус выполнения команды. Возможные значения: "done", "error"

Example

```
{  
  "transaction_uid": "550e8400-e29b-41d4-a716-446655440000",  
  "status": "done"  
}
```

5 Управление триггерами

PUB **'lm/trigger_service/trigger/trigger_list'**

Публикует список всех триггеров. Топик всегда содержит актуальный список.

Payload format

```
[
```

```
{
    name: str
    tr_type: str
    params: dict[str, Any]
}
```

- **name** - Имя триггера.
- **tr_type** - Тип триггера.
- **params** - Словарь с параметрами триггера.

Example

```
[
    {
        "name": "TriggerFromMqtt",
        "tr_type": "RawUDP",
        "params": {
            "network_type": "udp",
            "listen_ip": "0.0.0.0",
            "listen_port": "5555",
            "data": "any"
        }
    }
]
```

PUB `lm/trigger_service/action/action_list`

Публикует список всех action.

Топик всегда содержит актуальный список.

Payload format

```
[
    {
        name: str
        action_type: str
        params: dict[str, Any]
    }
]
```

- **name** - Имя action.
- **action_type** - Тип action.
- **params** - Словарь с параметрами action.

Example

```
[
    {
```

```

        "name": "default",
        "action_type": "send_trigger_to_mqtt",
        "params": {
            "topic": "lm/trigger_service/trigger/",
            "payload": "",
            "retain": false
        }
    }
]

```

PUB **lm/trigger_service/relation_list**

Публикует список всех связей между триггером и action.

Payload format

```

[
    {
        trigger: {
            name: str
            tr_type: str
            params: dict[str, Any]
        }
        action: {
            name: str
            action_type: str
            params: dict[str, Any]
        }
    }
]

```

- **trigger** - Словарь с триггером.
- **action** - Словарь с action.

Example

```

[
    {
        "trigger": {
            "name": "TriggerFromMqtt",
            "tr_type": "RawUDP",
            "params": {
                "network_type": "udp",
                "listen_ip": "0.0.0.0",
                "listen_port": "5555",
                "data": "any"
            }
        },
        "action": {

```

```

        "name": "default",
        "action_type": "send_trigger_to_mqtt",
        "params": {
            "topic": "lm/trigger_service/trigger/",
            "payload": "",
            "retain": false
        }
    }
}
]

```

SUB **lm/trigger_service/trigger/add**

Добавляет новый триггер.

На данный момент доступны три типа триггера: **RawUDP** и **ArtNet** и **Mqtt**.

- RawUDP - Срабатывает при получении UDP пакета удовлетворяющего заданным параметрам.
- ArtNet - Срабатывает при получении ArtNet пакета удовлетворяющего заданным параметрам.
- Mqtt - Срабатывает при получении Mqtt сообщения удовлетворяющего заданным параметрам.

Payload format

```

{
  name: str
  tr_type: str
  params: dict[str, Any]
}

```

- **name** - Имя триггера.
- **tr_type** - Тип триггера.
- **params** - Словарь с параметрами триггера. Параметры отличаются в зависимости от типа триггера.

Example

```

{
  "name": "TriggerFromMqtt",
  "tr_type": "RawUDP",
  "params": {
    "network_type": "udp",
    "listen_ip": "0.0.0.0",
    "listen_port": "5555",
    "data": "any"
  }
}

```

Ожидаемые Параметры

Параметры для триггера с типом RawUDP

```
{
    network_type: Literal['udp']
    listen_ip: str
    listen_port: int
    data: str
}
```

- **network_type** - Тип сети. Должен быть 'udp'.
- **listen_ip** - Прослушиваемый ip.
- **listen_port** - Прослушиваемый порт.
- **data** - Полезная нагрузка. Принимает строку полностью отражающую полезную нагрузку UDP пакета.

Example RawUDP params

```
{
    "network_type": "udp",
    "listen_ip": "0.0.0.0",
    "listen_port": "5555",
    "data": "any"
}
```

Параметры для триггера с типом ArtNet

```
{
    network_type: Literal['tcp', 'udp']
    listen_ip: str
    listen_port: int
    universe: int
    channel: int
    min_level: int
    max_level: int
}
```

- **network_type** - Тип сети. Принимает значения 'tcp' или 'udp'.
- **listen_ip** - Прослушиваемый ip.
- **listen_port** - Прослушиваемый порт.
- **universe** - Отражает значение параметра subuni из ArtNet пакета.
- **channel** - Номер канала в ArtNet пакете.
- **min_level** - Минимальное значение в канале для срабатывания триггера.
- **max_level** - Максимальное значение в канале для срабатывания триггера.

```
{
    "network_type": "udp",
    "listen_ip": "0.0.0.0",
    "listen_port": "6454",
    "universe": 3,
    "channel": 5,
    "min_level": 1,
    "max_level": 124
}
```

Параметры для триггера с типом Mqtt

```
{
  topic: str
  payload: str
}
```

- **topic** - Mqtt топик для отслеживания.
- **payload** - Полезная нагрузка mqtt сообщения в виде байт. Должна точно совпадать.Example Mqtt params

```
{
  "topic": "lm/di/port/1",
  "payload": "\x01"
}
```

SUB **lm/trigger_service/trigger/delete**

Удаляет триггер.

Payload format

```
{
  name: str
}
```

- **name** - Имя триггера.Example

```
{
  "name": "TriggerFromMqtt",
}
```

SUB **lm/trigger_service/action/add**

Добавляет новый action.

На данный момент доступны два типа action: **send_mqtt_msg_raw** и **send_trigger_to_mqtt**.

- **send_mqtt_msg_raw** - Отправляет по mqtt сообщение записанное в параметрах не внося в него никаких изменений.
- **send_trigger_to_mqtt** - Отправляет по mqtt сообщение в теле которого находится сработавший триггер.

Payload format

```
{
  name: str
}
```

```
    action_type: str
    params: dict[str, Any]
}
```

- **name** - Имя action.
- **action_type** - Тип action.
- **params** - Словарь с параметрами action. Различается в зависимости от типа action.Example

```
{
    "name": "default",
    "action_type": "send_trigger_to_mqtt",
    "params": {
        "topic": "lm/trigger_service/trigger/",
        "payload": "",
        "retain": false
    }
}
```

Ожидаемые Параметры

Параметры для actions с типом **send_trigger_to_mqtt** и **send_trigger_to_mqtt** совпадают.

```
{
    topic: str
    payload: str
    retain: bool
}
```

- **topic** - Mqtt topic в который будет отправлено сообщение.
- **payload** - Mqtt payload. Полезная нагрузка сообщения.
- **retain** - Mqtt retain param.

Типа **send_trigger_to_mqtt** игнорирует поля **payload** и **retain** но в сообщении они должны присутствовать.

Example params

```
{
    "topic": "lm/trigger_service/trigger/",
    "payload": "",
    "retain": false
}
```

SUB **lm/trigger_service/action/delete**

Удаляет action.

Payload format

```
{
  name: str
}
```

- **name** - Имя action.

Example

```
{
  "name": "default",
}
```

SUB **lm/trigger_service/set_trigger_to_action_relation**

Создает связь между триггером и action.

Payload format

```
trigger: {
  name: str
  tr_type: str
  params: dict[str, Any]
}
action: {
  name: str
  action_type: str
  params: dict[str, Any]
}
```

- **trigger** - Словарь с триггером.
- **action** - Словарь с action.

Example

```
{
  "trigger": {
    "name": "TriggerFromMqtt",
    "tr_type": "RawUDP",
    "params": {
      "network_type": "udp",
      "listen_ip": "0.0.0.0",
      "listen_port": "5555",
      "data": "any"
    }
  },
  "action": {
    "name": "default",
    "action_type": "send_trigger_to_mqtt",
    "params": {
```

```

        "topic": "lm/trigger_service/trigger/",
        "payload": "",
        "retain": false
    }
}

```

SUB **lm/trigger_service/delete_trigger_to_action_relation**

Удаляет связь между триггером и action.

Payload format

```

trigger: {
    name: str
    tr_type: str
    params: dict[str, Any]
}
action: {
    name: str
    action_type: str
    params: dict[str, Any]
}

```

- **trigger** - Словарь с триггером.
- **action** - Словарь с action.

Example

```

{
    "trigger": {
        "name": "TriggerFromMqtt",
        "tr_type": "RawUDP",
        "params": {
            "network_type": "udp",
            "listen_ip": "0.0.0.0",
            "listen_port": "5555",
            "data": "any"
        }
    },
    "action": {
        "name": "default",
        "action_type": "send_trigger_to_mqtt",
        "params": {
            "topic": "lm/trigger_service/trigger/",
            "payload": "",
            "retain": false
        }
    }
}

```

```
}
```

PUB **lm/trigger_service/error**

Публикует ошибки.

Выставляет заголовок **Correlation data** если он был установлен в запросе.

Payload format

```
{  
  msg: str  
  data: Any  
}
```

- **msg** - contain error message
- **data** - contain related error data

SUB **lm/trigger_service/delete_trigger_with_related_actions**

Удаляет триггер и все связанные с ним действия.

Payload format

```
{  
  name: str  
}
```

- **name** - Имя триггера.Example

```
{  
  "name": "TriggerFromMqtt",  
}
```

Б. Настройки системы

Сервис осуществляет конфигурирование системных настроек ОС.

PUB **lm/system_configurator/error**

Публикует ошибки.

Выставляет заголовок **Correlation data** если он был установлен в запросе.

Payload format

```
{
  msg: str
  data: Any
}
```

- **msg** - contain error message
- **data** - contain related error data

PUB **lm/system_settings/external_access/certificates**

Публикует список всех x509 сертификатов.
Топик всегда содержит актуальный список.

Payload format

```
[
  {
    name: str
    cert_type: str
    public_bytes: str
    params: dict[str, Any]
  }
]
```

- **name** - Имя сертификата.
- **cert_type** - Тип сертификата. Может принимать значения 'csr' или 'certificate'
- **params** - Словарь с параметрами сертификата. Набор параметров отличается в зависимости от [типа](#) сертификата.

Example

```
[
  {
    "cert_type": "certificate",
    "name": "cert_name",
    "params": {
      "issuer": "OU=test ou,CN=domain.com,O=test o,L=123,ST=st,C=UA",
      "san": "IP=192.168.0.3",
      "subject": "OU=test ou,CN=domain.com,O=test o,L=123,ST=st,C=UA",
      "valid_from": "1664440221.0",
      "valid_to": "1759048221.0"
    },
    "public_bytes": "-----BEGIN CERTIFICATE-----\n"
      "-----END CERTIFICATE-----\n"}
]
```

PUB **lm/system_settings/external_access/web_access_settings**

Публикует список настроек web доступа.
Топик всегда содержит актуальный список.

Payload format

```
{
  http_port: int
  https_port: int
  is_https_enabled: bool
  is_http_redirected: bool
  cert_name: str
}
```

- **http_port** - Http порт. По умолчанию 80.
- **https_port** - Https порт. По умолчанию 443.
- **is_https_enabled** - Индикатор включен ли https.
- **is_http_redirected** - Индикатор включена ли переадресация http to https.
- **cert_name** - Имя сертификата сервера.

Example

```
{
  "http_port": 80,
  "https_port": 443,
  "is_https_enabled": false,
  "is_http_redirected": true,
  "cert_name": ""
}
```

SUB **lm/system_settings/external_access/change_web_access_settings**

Меняет настройки web доступа.

Payload format

```
{
  http_port: int
  https_port: int
  is_https_enabled: bool
  is_http_redirected: bool
  cert_name: str
}
```

- **http_port** - Http порт. По умолчанию 80.
- **https_port** - Https порт. По умолчанию 443.
- **is_https_enabled** - Индикатор включен ли https.
- **is_http_redirected** - Индикатор включена ли переадресация http to https.

- **cert_name** - Имя сертификата сервера.

Example

```
{
  "http_port": 80,
  "https_port": 443,
  "is_https_enabled": false,
  "is_http_redirected": true,
  "cert_name": ""
}
```

SUB **lm/system_settings/certificates/upload_certificate**

Загружает сертификат и его ключ для дальнейшего использования в настройках доступа.

Payload format

```
{
  cert_name: str
  certificate: bytes
  key: bytes
  intermediate: bytes
}
```

- **cert_name** - Читаемое имя сертификата.
- **certificate** - x.509 сертификат в pem формате.
- **key** - Приватный ключ в pem формате.
- **intermediate** - (Опционально) промежуточный сертификат.

SUB

lm/system_settings/certificates/upload_certificate_corresponding_csr

Загружает сертификат относящийся к сформированному ранее csr.

Payload format

```
{
  cert_name: str
  certificate: bytes
}
```

- **cert_name** - Имя csr сертификата.
- **certificate** - x.509 сертификат в pem формате.

SUB **lm/system_settings/certificates/delete_certificate**

Удаляет сертификат и все связанные с ним файлы.

Payload format

```
{
  id: int
  name: str
  cert_type: str
  public_bytes: str
  params: dict[str, Any]
}
```

- **id** - (Опционально) Идентификатор сертификата.
- **name** - Имя сертификата.
- **cert_type** - Тип сертификата. Может принимать значения 'csr' или 'certificate'
- **public_bytes** - Открытый ключ сертификата.
- **params** - Словарь с параметрами сертификата. Набор параметров отличается в зависимости от [типа](#) сертификата.

Example

```
{
  "cert_type": "certificate",
  "name": "cert_name",
  "params": {
    "issuer": "OU=test ou,CN=domain.com,O=test o,L=123,ST=st,C=UA",
    "san": "IP=192.168.0.3",
    "subject": "OU=test ou,CN=domain.com,O=test o,L=123,ST=st,C=UA",
    "valid_from": "1664440221.0",
    "valid_to": "1759048221.0"
  },
  "public_bytes": "-----BEGIN CERTIFICATE-----\n"
    "-----END CERTIFICATE-----\n"}]
```

SUB **lm/system_settings/certificates/generate_csr**

Генерирует Certificate Signing Request.

Payload format

```
{
  cert_name: str
  cert_type: str
  key_size: int
  subject: str
  san: str
}
```

```
}
```

- **cert_name** - Имя сертификата.
- **cert_type** - Тип сертификата. Может принимать значения 'csr' или 'certificate'
- **key_size** - Размер ключа в байтах. Принимает значения 2048 и 4096.
- **subject** - Строка в формате rfc4514.
- **san** - Стока представляющее расширение SubjectAltName. Принимаются только ip адреса или dns имена идущие подряд через запятую без пробелов с префиксами **IP=** или **DNS=**.

Example

```
{  
  "cert_name": "ss_cert23",  
  "cert_type": "certificate",  
  "key_size": 2048,  
  "subject": "OU=test ou,CN=domain.com,O=test o,L=123,ST=st,C=UA",  
  "san": "IP=192.168.0.3,DNS=domain.com"  
}
```

SUB **lm/system_settings/certificates/generate_self_sign_certificate**

Генерирует самоподписанный сертификат.

Payload format

```
{  
  cert_name: str  
  cert_type: str  
  key_size: int  
  subject: str  
  san: str  
}
```

- **cert_name** - Имя сертификата.
- **cert_type** - Тип сертификата. Может принимать значения 'csr' или 'certificate'.
- **key_size** - Размер ключа в байтах. Принимает значения 2048 и 2096.
- **subject** - Строка в формате rfc4514.
- **san** - Стока представляющее расширение SubjectAltName. Принимаются только ip адреса или dns имена идущие подряд через запятую без пробелов с префиксами **IP=** или **DNS=**.

Example

```
{  
  "cert_name": "ss_cert23",  
  "cert_type": "certificate",  
  "key_size": 2048,  
  "subject": "OU=test ou,CN=domain.com,O=test o,L=123,ST=st,C=UA",  
  "san": "IP=192.168.0.3,DNS=domain.com"  
}
```

PUB `lm/system_settings/network/interfaces/wired/eth*/statistics`

PUB `lm/system_settings/network/interfaces/wired/eth0/statistics`

PUB `lm/system_settings/network/interfaces/wired/eth1/statistics`

Публикует информацию о проводном интерфейсе ethernet каждые 10 секунд.

Payload format

```
{
  status: str
  ip_assign_method: Literal['manual', 'dhcp']
  ip: str
  netmask: str
  gateway: str
  dns_assign_method: Literal['manual', 'dhcp']
  dns_servers: list[str]
  mac_address: str
}
```

- **status** - Статус интерфейса. Может быть **up** или **down**.
- **ip_assign_method** - Способ назначения ip адреса. Может быть **manual** или **dhcp**.
- **ip** - IP адрес интерфейса.
- **netmask** - Маска интерфейса.
- **gateway** - Шлюз по умолчанию.
- **dns_assign_method** - Способ назначения dns серверов. Может быть **manual** или **dhcp**.
- **dns_servers** - Список dns серверов.
- **mac_address** - MAC адрес интерфейса.

Example

```
{
  "status": "up",
  "ip_assign_method": "manual",
  "ip": "192.168.0.205",
  "netmask": "255.255.255.0",
  "gateway": "192.168.0.1",
  "dns_assign_method": "manual",
  "dns_servers": ["8.8.8.8", "8.8.4.4"],
  "mac_address": "e4:5f:01:a8:e0:6c"
}
```

SUB

`lm/system_settings/network/interfaces/wired/eth*/set_ip_credentials`

SUB `lm/system_settings/network/interfaces/wired/eth0/set_ip_credentials`
`lm/system_settings/network/interfaces/wired/eth1/set_ip_credentials`

Устанавливает ip адресацию и шлюз на интерфейс.

Поддерживает статическое назначение ip и назначение через dhcp.

Payload format

Статическая адресация:

```
{
  ip_assign_method: Literal['manual']
  static_ip: str
  static_netmask: str
  static_gateway: str
}
```

- **ip_assign_method** - Способ назначения ip адреса. Должно быть **manual**.
- **static_ip** - IPv4 адрес интерфейса
- **static_netmask** - Сетевая маска интерфейса.
- **static_gateway** - Шлюз по умолчанию.

Example

```
{
  "ip_assign_method": "manual",
  "static_ip": "192.168.0.205",
  "static_netmask": "255.255.255.0",
  "static_gateway": "192.168.0.1"
}
```

Динамическая адресация

```
{
  ip_assign_method: Literal['dhcp']
}
```

- **ip_assign_method** - Способ назначения ip адреса. Должно быть **dhcp**.

Example

```
{
  "ip_assign_method": "dhcp"
}
```

SUB

lm/system_settings/network/interfaces/wired/eth*/set_dns_credential

SUB **lm/system_settings/network/interfaces/wired/eth0/set_dns_credential**

SUB `lm/system_settings/network/interfaces/wired/eth1/set_dns_credential`

Назначение dns серверов на интерфейс.

Поддерживает статическое и динамическое (dhcp) назначение dns серверов.

Payload format

Статическое назначение:

```
{  
  dns_assign_method: Literal['manual']  
  static_dns_servers: list[str]  
}
```

- `dns_assign_method` - Способ назначения dns серверов. Должно быть **manual**.
- `static_dns_servers` - Список DNS серверов. Example

```
{  
  "dns_assign_method": "manual",  
  "static_dns_servers": ["8.8.8.8", "8.8.4.4"]  
}
```

Динамическое назначение:

```
{  
  dns_assign_method: Literal['dhcp']  
}
```

- `dns_assign_method` - Способ назначения dns серверов. Должно быть **dhcp**.

Example

```
{  
  "dns_assign_method": "dhcp"  
}
```

PUB `lm/system_settings/network/interfaces/modem/statistics`

Публикует информацию о модемном интерфейсе каждые 10 секунд.

Payload format

```
{  
  ip_assign_method: Literal['manual', 'dhcp']  
  ip: str  
  netmask: str  
  gateway: str  
  dns_assign_method: Literal['manual', 'dhcp']  
  dns_servers: list[str]
```

```

apn: {
    apn: str,
    username: str,
    password: str,
}
modem_status: {
    state: str,
    state_failed_reason: str,
    power_state: str,
    signal_quality: int,
    access_technologies: list[str]
}
}

```

- **status** - Статус интерфейса. Может быть **up** или **down**.
- **ip_assign_method** - Способ назначения ip адреса. Может быть **manual** или **dhcp**.
- **netmask** - IP адрес интерфейса.
- **gateway** - Шлюз по умолчанию.
- **dns_assign_method** - Способ назначения dns серверов. Может быть **manual** или **dhcp**.
- **dns_servers** - Список dns серверов.
- **apn**:
 - **apn**: APN сервер.
 - **username**: Имя пользователя для apn сервера.
 - **password**: Пароль для apn сервера.
- **modem_status**:
 - **state**: Состояние подключения.
 - **state_failed_reason**: Причина ошибки если таковая есть.
 - **power_state**: Состояние питания модема.
 - **signal_quality**: Качество сигнала в процентах.
 - **access_technologies**: Список текущих режимов (LTE, UMTS и т.д.).

Example

```

{
    "status": "up",
    "ip_assign_method": "manual",
    "ip": "192.168.0.205",
    "netmask": "255.255.255.0",
    "gateway": "192.168.0.1",
    "dns_assign_method": "manual",
    "dns_servers": ["8.8.8.8", "8.8.4.4"],
    "apn": {
        "apn": "internet.mts.ru",
        "username": "mts",
        "password": "mts"
    },
    "modem_status": {
        "state": "connected",
        "state_failed_reason": "--",
        "power_state": "on",
        "signal_quality": 81,
    }
}

```

```
        "access_technologies": ["LTE"]
    }
}
```

SUB **lm/system_settings/network/interfaces/modem/set_ip_credential**

Устанавливает ip адресацию и шлюз на интерфейс.

Поддерживает статическое назначение ip и назначение через dhcp.

Payload format

Статическая адресация

```
{
    ip_assign_method: Literal['manual']
    static_ip: str
    static_netmask: str
    static_gateway: str
}
```

- **ip_assign_method** - Способ назначения ip адреса. Должно быть **manual**.
- **static_ip** - IPv4 адрес интерфейса
- **static_netmask** - Сетевая маска интерфейса.
- **static_gateway** - Шлюз по умолчанию.Example

```
{
    "ip_assign_method": "manual",
    "static_ip": "192.168.0.205",
    "static_netmask": "255.255.255.0",
    "static_gateway": "192.168.0.1"
}
```

Динамическая адресация

```
{
    ip_assign_method: Literal['dhcp']
}
```

- **ip_assign_method** - Способ назначения ip адреса. Должно быть **dhcp**.

Example

```
{
    "ip_assign_method": "dhcp"
}
```

SUB `lm/system_settings/network/interfaces/modem/set_dns_credential`

Назначение dns серверов на интерфейс.

Поддерживает статическое и динамическое (dhcp) назначение dns серверов.

Payload format

Статическое назначение:

```
{
  dns_assign_method: Literal['manual']
  static_dns_servers: list[str]
}
```

- **dns_assign_method** - Способ назначения dns серверов. Должно быть **manual**.
- **static_dns_servers** - Список DNS серверов.

Example

```
{
  "dns_assign_method": "manual",
  "static_dns_servers": ["8.8.8.8", "8.8.4.4"]
}
```

Динамическое назначение

```
{
  dns_assign_method: Literal['dhcp']
}
```

- **dns_assign_method** - Способ назначения dns серверов. Должно быть **dhcp**.

Example

```
{
  "dns_assign_method": "dhcp"
}
```

SUB `lm/system_settings/network/interfaces/modem/set_apn_credential`

Назначение настроек apn на интерфейс.

Поддерживается только статическое назначение.

Payload format

Статическое назначение:

```
{
  apn: str
  username: str
  password: str
}
```

- **apn** - APN сервер.
- **username** - Имя пользователя если есть либо пустая строка.
- **password** - Пароль если есть либо пустая строка.

Example

```
{
  "apn": "internet.mts.ru",
  "username": "mts",
  "password": "mts"
}
```

PUB **lm/system_settings/datetime/rtc_status**

Публикует статус rtc модуля

Payload format

```
{
  is_active: bool
}
```

- **is_active** - Активен ли rtc модуль.

Example

```
{
  "is_active": true,
}
```

SUB **lm/system_settings/datetime**

Принимает [команды](#) на изменение даты и времени конфигурации системы.

Список принимаемых команд

Set Date

Description: > Set system date.

Values:

command: str > set_date

data: dict > date: str - date in format 'Y:M:D'

Example:

```
{'command': 'set_date', 'data': {'date': '1970:01:01'}}
```

Set Time

Description: > Set system time.

Values:

command: str > set_time

data: dict > time: str - time in format 'HH:mm:ss'

Example:

```
{'command': 'set_time', 'data': {'time': '13:00:00'}}
```

Set Datetime

Description: > Set system date and time.

Values:

command: str > set_datetime

data: dict > datetime: str - time in format 'Y:M:D HH:mm:ss'

Example:

```
{'command': 'set_datetime', 'data': {'datetime': '1970:01:01 13:00:00'}}
```

Change Ntp Status

Description: > Enable or disable ntp synchronization.

Values:

command: str > change_ntp_status

data: dict > ntp: bool - is ntp sync enable

Example:

```
{'command': 'change_ntp_status', 'data': {'ntp': True}}
```

Set Ntp Servers

Description: > Set ntp servers. > Generate ntp config, replace it then restart systemd-timesyncd.service > Accepts list

of ip addresses or domain names

Values:

command: str > set_ntp_servers

data: dict > ntp_servers: list[str] - list of servers ip addresses or dns names

Example:

```
{'command': 'set_ntp_servers', 'data': {'ntp_servers': ['192.168.0.2', 'ntp1.stratum2.com']}}
```

Set timezone

Description: > Set system timezone.

Values:

command: str > set_timezone

data: dict > timezone: str - timezone name

Example:

```
{'command': 'set_timezone', 'data': {'timezone': 'Europe/London'}}
```

Base format for command payload

```
{
  'command': str
  'data': dict[str, Any]
}
```

- **command** - command name
- **data** - any data for command

Example:

```
{'command': 'set_ip', 'data': {'ifname': 'eth0', 'ip': '192.168.0.1'}}
```

SUB **lm/system_settings/power_control**

Управляет питанием устройства

Payload format

```
{
  command: str
  delay: int
}
```

- **command** - Команда управления питанием. Может принимать значения "reboot" и "shutdown".

- **delay** - Задержка срабатывания команды в минутах.

Example

```
{
  "command": "reboot",
  "delay": "0",
}
```

Certificate params format

Параметры сертификата отличаются в зависимости от его типа. В данный момент поддерживается два типа сертификата x509: **certificate** и **csr**.

x509 certificate params format

```
{
  subject: str
  san: str
  issuer: str
  valid_from: float
  valid_to: float
}
```

- **subject** - Строка в формате rfc4514.
- **san** - Стока представляющее расширение SubjectAltName. Принимаются только ip адреса или dns имена идущие подряд через запятую без пробелов с префиксами **IP=** или **DNS=**.
- **issuer** - Строка в формате rfc4514.
- **valid_from** - Дата с которой сертификат действителен. Формат Posix timestamp.
- **valid_to** - Дата по которую сертификат действителен. Формат Posix timestamp.

Example

```
{
  "issuer": "OU=test ou,CN=domain.com,O=test o,L=123,ST=st,C=UA",
  "san": "IP=192.168.0.3",
  "subject": "OU=test ou,CN=domain.com,O=test o,L=123,ST=st,C=UA",
  "valid_from": "1664440221.0",
  "valid_to": "1759048221.0"
}
```

x509 csr params format

```
{
  subject: str
  san: str
}
```

- **subject** - Строка в формате rfc4514.
- **san** - Стока представляющее расширение SubjectAltName. Принимаются только ip адреса или dns

имена идущие подряд через запятую без пробелов с префиксами **IP=** или **DNS=**.

Example

```
{ "subject": "OU=test ou,CN=domain.com,O=test o,L=123,ST=st,C=UA", "san":  
"IP=192.168.0.3", }
```

7. Управление Di Do интерфейсами плеера

PUB **lm/di/port/***

PUB **lm/di/port/0** (player V1 only) PUB **lm/di/port/1** PUB **lm/di/port/2** (player V2 only) PUB
lm/di/port/3 (player V2 only)

Публикует состояние di порта

- **di_port_number** - Номер di порта.

Payload format

int

Example

1

- **int** - Статус Di порта. 1 - активен, 0 - неактивен.

PUB **lm/do/port/***

PUB **lm/do/port/0** (player V1 only)

PUB **lm/do/port/1** PUB **lm/do/port/2** (player V2 only) PUB **lm/do/port/3** (player V2 only)

Публикует состояние do порта

- **do_port_number** - Номер do порта.

Payload format

int

Example

1

- **int** - Статус DO порта. 1 - активен, 0 - неактивен.

SUB **lm/do/change_state**

Принимает команды для изменения состояния DO порта.

Payload command format

```
{
  "port": int,
  "state": int,
}
```

Example

```
{
  "port": 1,
  "state": 1,
}
```

- **port** - Номер do порта.
- **state** - Статус порта. 1 - активен, 0 - неактивен.

8. Управление RS485 интерфейсами плеера

PUB **lm/serialport_controller/error**

Публикует ошибки.

Выставляет заголовок **Correlation data** если он был установлен в запросе.

Payload format

```
{
  msg: str
  data: Any
}
```

- **msg** - contain error message
- **data** - contain related error data

PUB **lm/serialport_controller/ports**

Публикует список rs485 портов.

Payload format

```
[
  {
    name: str
    mode: Literal['rs485', 'dmx0ut']
  }
]
```

- **name** - Имя порта.
- **mode** - Предназначение порта.

Example

```
[
  {
    "name": "port1",
    "mode": "rs485",
  },
  {
    "name": "port2",
    "mode": "rs485",
  },
  {
    "name": "port3",
    "mode": "dmx0ut",
  },
  {
    "name": "port4",
    "mode": "dmx0ut",
  }
]
```

SUB **lm/serialport_controller/ports/change_mode**

Меняет предназначение порта.

Payload format

```
{
  name: str
  mode: Literal['rs485', 'dmx0ut']
}
```

- **name** - Имя порта.
- **mode** - Предназначение порта.

Example

```
{
  "name": "port1",
  "mode": "rs485",
}
```

9. Управление светодиодами плеера

PUB '**lm/leds/state'**

Публикует состояние диодов rs485 портов

Payload format

```
{
  Port1: {
    green: bool,
    red: bool,
  },
  Port2: {
    green: bool,
    red: bool,
  },
  Port3: {
    green: bool,
    red: bool,
  },
  Port4: {
    green: bool,
    red: bool,
  },
}
```

Example

```
{
  "Port1": {
    "green": true,
    "red": true,
  },
  "Port2": {
    "green": true,
    "red": true,
  },
  "Port3": {
    "green": true,
```

```

    "red": true,
  },
  "Port4": {
    "green": true,
    "red": true,
  },
}

```

- **green** - Статус зеленого светодиода.
- **red** - Статус красного светодиода.

SUB **lm/leds/change_state**

Принимает команды для изменения состояния диодов у rs485 порта.

Payload command format

```

{
  pub port: Literal['Port1', 'Port2', 'Port3', 'Port4'],
  green: bool,
  red: bool,
}

```

Example

```

{
  "port": "Port1",
  "green": true,
  "red": false,
}

```

- **port** - Имя rs485 порта.
- **green** - Статус зеленого светодиода.
- **red** - Статус красного светодиода.

SUB **lm/leds/blink**

Принимает команды для мигания всех светодиодов на всех rs485 портах.

Payload format

```

{
  times: int,
  interval: int,
}

```

Example

```

{
  "times": 5,
  "interval": 1000
}

```

```
}
```

- **times** - Количество миганий (от 1 до 255).
- **interval** - Интервал между миганиями в миллисекундах.

10. Обновление программного обеспечения плеера

PUB `lm/update_service/version/version_list`

Публикует список версий всех модулей. Топик всегда содержит актуальный список.

Payload format

```
[  
  {  
    id: int  
    version: str  
    subversion: Optional[str]  
    module: str  
    description: Optional[str]  
  }  
]
```

- **id** - version id
- **version** - version number
- **subversion** - (Optional) subversion.
- **module** - module name
- **description** - (Optional) description

Example

```
[  
  {  
    "id": 1,  
    "version": "20",  
    "subversion": null,  
    "module": "frontend",  
    "description": null  
  }  
]
```

PUB `lm/update_service/update/update_list`

Публикует список обновлений. Топик всегда содержит актуальный список.

Payload format

```
[
  {
    id: int
    version: str
    status: str
    filename: Optional[str]
    update_path: str
    extracted_path: Optional[str]
    backup_path: Optional[str]
    description: Optional[str]
  }
]
```

- **id** - update id.
- **version** - update version.
- **status** - update status.
- **filename** - (Optional) update filename.
- **update_path** - path to update file.
- **extracted_path** - path to extracted files.
- **backup_path** - (Optional) update version.
- **description** - (Optional) description.

Example

```
[
  {
    "id": 1,
    "version": "2022",
    "status": "installed",
    "filename": "lmp_2022.update",
    "update_path":
"/home/lightmaster/lightmaster/updater/lmp_2022.update",
    "extracted_path":
"/home/lightmaster/lightmaster/updates_store/lmp_2022",
    "backup_path":
"/home/lightmaster/lightmaster/backups_store/20220519181452_lmp_v0_full_backu
p",
    "description": "A error occurred during installation update.
Installation filed. None"
  }
]
```

SUB `lm/update_service/update/add_update`

Добавляет обновление в базу.

Payload format

```
{  
  file: str  
}
```

- **file: str** - путь до файла обновления

Example

```
{"file": "/home/lightmaster/projects/wess-group/lightmaster/updater/lmp_2022.update"}
```

SUB `lm/update_service/update/check_update`

Проверяет совместимость обновления.

Payload format

```
{  
  id: int  
}
```

- **id** - id обновления

Example

```
{'id': 5}
```

SUB `lm/update_service/update/initial_update`

Совмещает добавление обновления в базу и его проверку.

Payload format

```
{  
  file: str  
}
```

- **file: str** - путь до файла обновления

Example

```
{"file": "/home/lightmaster/projects/wess-group/lightmaster/updater/lmp_2022.update"}
```

SUB **lm/update_service/update/install_update**

Устанавливает обновление

Payload format

```
{  
  id: int  
}
```

- **id** - id обновления

Example

```
{'id': 5}
```

SUB **lm/update_service/update/restore_update**

Откатывает обновление на предыдущую версию.

Payload format

```
{  
  id: int  
}
```

- **id** - id обновления

Example

```
{'id': 5}
```

SUB **lm/update_service/update/delete_update**

Удаляет обновление и все связанные с ним файлы.

Payload format

```
{  
  id: int  
}
```

- **id** - id обновления

Example

```
{'id': 5}
```

SUB **lm/update_service/version/get_versions_list**

Запрос на публикацию списка версий всех модулей.

Публикация происходит в топик **lm/update_service/version/get_versions_list/response**

В заголовок запроса могут быть включены необязательные поля:

- Correlation data
- Response topic

Corelation data любой уникальный идентификатор запроса. Зеркально устанавливается в публикуемый ответ и служит для идентификации ответа со стороны клиента.

Response topic если установлен то ответ публикуется в указанный топик вместо стандартного.

PUB **lm/update_service/version/get_versions_list/response**

Публикует ответ на запрос из топика **lm/update_service/version/get_versions_list**.

Выставляет заголовок **Correlation data** если он был установлен в запросе.

Payload format

```
[
  {
    id: int
    version: str
    subversion: Optional[str]
    module: str
    description: Optional[str]
  }
]
```

- **id** - version id
- **version** - version number
- **subversion** - (Optional) subversion.
- **module** - module name
- **description** - (Optional) description

Example

```
[
  {
```

```
    "id": 1,
    "version": "20",
    "subversion": null,
    "module": "frontend",
    "description": null
  }
]
```

SUB `lm/update_service/version/get_module_version`

Публикует версию конкретного модуля.

Публикация происходит в топик `lm/update_service/version/get_module_version/response`

В заголовок запроса могут быть включены необязательные поля:

- Correlation data
- Response topic

Corelation data любой уникальный идентификатор запроса. Зеркально устанавливается в публикуемый ответ и служит для идентификации ответа со стороны клиента.

Response topic если установлен то ответ публикуется в указанный топик вместо стандартного.

Payload format

```
{
  module: str
}
```

- **module** - название модуля

Example

```
{'module': 'update_service'}
```

PUB `lm/update_service/version/get_module_version/response`

Публикует ответ на запрос из топика `lm/update_service/version/get_module_version`.

Выставляет заголовок **Correlation data** если он был установлен в запросе.

Payload format

```
{
  id: int
}
```

```
version: str
subversion: Optional[str]
module: str
description: Optional[str]
}
```

- **id** - version id
- **version** - version number
- **subversion** - (Optional) subversion.
- **module** - module name
- **description** - (Optional) description

Example

```
{
  "id": 1,
  "version": "20",
  "subversion": null,
  "module": "frontend",
  "description": null
}
```

SUB **lm/update_service/update/get_updates_list**

Запрос на публикацию списка всех обновлений добавленных в базу.

Публикация происходит в ветку **lm/update_service/update/get_updates_list/response**

В заголовок запроса могут быть включены необязательные поля:

- Correlation data
- Response topic

Corelation data любой уникальный идентификатор запроса. Зеркально устанавливается в публикуемый ответ и служит для идентификации ответа со стороны клиента.

Response topic если установлен то ответ публикуется в указанный топик вместо стандартного.

PUB **lm/update_service/update/get_updates_list/response**

Публикует ответ на запрос из топика **lm/update_service/update/get_updates_list**.

Выставляет заголовок **Correlation data** если он был установлен в запросе.

Payload format

```
[
  {
```

```

        id: int
        version: str
        status: str
        filename: Optional[str]
        update_path: str
        extracted_path: Optional[str]
        backup_path: Optional[str]
        description: Optional[str]
    }
]

```

- **id** - update id.
- **version** - update version.
- **status** - update status.
- **filename** - (Optional) update filename.
- **update_path** - path to update file.
- **extracted_path** - path to extracted files.
- **backup_path** - (Optional) update version.
- **description** - (Optional) description.

Example

```

[
  {
    "id": 1,
    "version": "2022",
    "status": "installed",
    "filename": "lmp_2022.update",
    "update_path":
"/home/lightmaster/lightmaster/updater/lmp_2022.update",
    "extracted_path":
"/home/lightmaster/lightmaster/updates_store/lmp_2022",
    "backup_path":
"/home/lightmaster/lightmaster/backups_store/20220519181452_lmp_v0_full_backu
p",
    "description": "A error occurred during installation update.
Installation filed. None"
  }
]

```

PUB **lm/update_service/error**

Публикует ошибки.

Выставляет заголовок **Correlation data** если он был установлен в запросе.

Payload format

```
{  
  msg: str  
  data: Any  
}
```

- **msg** - contain error message
- **data** - contain related error data

II. Управление внешними датчиками

Описывает MQTT API сервиса управления внешними датчиками.

PUB **lm/sensors/{sensor_id}/data**

Публикует данные датчика.

Payload format

```
{  
  str  
}
```

- str - данные датчика

Example

```
{  
  "34"  
}
```