

**Руководство пользователя
Topic Mapper**

Оглавление

1. Описание	4
2. Основные функции	4
3. Формат конфигурирования YAML	5
3.1 Структура файлов конфигурации	5
3.1.1 Настройка mqtt	6
3.1.2 Настройки логирования	7
3.1.3 Настройка алгоритмов	8
3.2 Валидация конфигурации	8
3.3 Алгоритм 1	9
3.3.1 Описание алгоритма	9
3.3.2 Логика OR	10
3.3.3 Логика AND	10
3.3.4 Настройка Алгоритма 1	10
3.3.5 Пример описания в YAML	11
3.3.6 Пример топика /payload	11
3.3.7 Пример обработки управления	11
3.4 Алгоритм 2	12
3.4.1 Описание алгоритма	12
3.4.2 Пример корневого файла конфигурации Алгоритма 2	12
3.4.3 Настройка Алгоритма 2	12
3.4.4 Примеры конфигурации topics.topic_file	12
3.4.5 Настройка маппинга Алгоритма 2 (topic.topic_file)	15
3.4.6 Настройка mapping	16
3.4.7 Пример конфигурации index_tag	18
3.4.8 Настройка index_tag	20
3.4.9 Настройка bit_mapping	21
3.4.10 Настройка payload	22
3.4.11 Selector (селектор)	22
3.4.12 Правила и ограничения по пользованию Selector	22
3.4.13 Синтаксис Selector'а	23
3.5 Алгоритм 3	23
3.5.1 Описание алгоритма	23
3.5.2 Общая структура топиков	24
3.5.3 Пример. Сложная структура с полным идентификатором устройства	24

3.5.4 Пример. Структура с сокращенным идентификатором параметра	24
3.5.5 Дополнительный идентификатор	25
3.5.6 Пример описания YAML	25
3.5.7 Настройка Алгоритма 3.....	25
4. Демо-режим.....	25
5. Аргументы запуска	26



Универсальный Topic Mapper — сервис, предназначенный для преобразования (маппинга) наименований MQTT-топиков. Сервис подписывается на указанные исходные топики, получает сообщения от MQTT-брокера и перенаправляет их содержимое в новые топики, указанные в конфигурационном файле. Это позволяет гибко управлять структурой MQTT-сообщений и адаптировать их под требования различных систем.

1. Описание

В текущей реализации поддерживаются три типа реализованных алгоритмов:

1. **Алгоритм 1** - предназначен для преобразования MQTT-топиков, независимо от MQTT payload.
2. **Алгоритм 2** - предназначен для преобразования содержимого payload (в формате JSON) MQTT топиков, с последующей выдачей топиков в соответствии с конфигурацией.
3. **Алгоритм 3** – предназначен для обработки и пересылки сообщений MQTT с корректировкой топиков.

2. Основные функции

1. **Подписка на исходные топики:** сервис подписывается на топики, указанные в конфигурационных файлах, и ожидает входящих сообщений.
2. **Гибкая конфигурация:** конфигурация сервиса задается через файл, что позволяет легко адаптировать его под различные сценарии использования.
3. **Обработка сообщений:** в зависимости от алгоритма, сервис обрабатывает payload и перенаправляет значения в целевые топики.
4. **Автопубликация данных:** задает интервал в секундах, через который сервис будет проверять состояние топиков и автоматически публиковать их значения, если изменения не были обнаружены. Это полезно для поддержания актуальности данных в системах, где топики могут не обновляться длительное время.
5. **Логирование:** логирует входящие и исходящие сообщения, ошибки и дополнительную информацию для отладки и мониторинга в зависимости от конфигурации логгера.

ВАЖНО!

При работе с сервисом важно учитывать следующие особенности:

1. Если в конфигурации **алгоритма 1** или **алгоритма 2** присутствуют одинаковые входные топики, то параметры для маппинга(преобразования) будут использованы из конфигурации топика, который был указан последним.
2. Для **алгоритма 2** важно понимать риски использования индексов, а не фильтров полей. Поскольку, содержание и порядок элементов в JSON объекте или массиве определяется отправителем данного JSON, то фильтры полей являются более надёжным вариантом (если применимо).

3. Формат конфигурирования YAML

3.1 Структура файлов конфигурации

mapper.yaml – основной (корневой) файл конфигурации, в данном примере:

```
# mapper_root_config.yaml

logger:
  level_log: 3
  path_log: "/opt/topic_mapper/logs/topic_mapper.log"
  path_json_log: "/opt/topic_mapper/logs/topic_mapper.json"
  max_size_log: 2
  max_backups_log: 30
  max_age_log: 10

mqtt:
  broker: "tcp://127.0.0.1:1883" # URL брокера tcp://127.0.0.1:1883 или Unix Socket
  "unix:///var/run/mosquitto/mosquitto.sock"
  client_id: "topic_mapper"
  keepalive: 20
  username: "test"
  password: "test_password"
  max_retries: 5000
  retry_interval: 5
  keepalive: 20
  ping_timeout: 60
  connect_timeout: 60
  clean_session: true
  autoreconnect: true
  pubQoS: 1
  subQoS: 2
  retained: false
  retained: false # TLS конфигурация. Опционально
  tls:
    enabled: false
    ca_cert: ""
    cert_file: ""
    key_file: ""
    insecure_skip_verify: false

Algorithm1:
  enabled: true # Флаг, определяющий, активен ли алгоритм
  config_file: "algorithm_1.yaml" # Путь к файлу конфигурации алгоритма 1

Algorithm2:
  enabled: true # Флаг, определяющий, активен ли алгоритм
  config_file: "algorithm_2.yaml" # Путь к файлу конфигурации алгоритма 2

Algorithm3:
```

```
enabled: true # Флаг, определяющий, активен ли алгоритм
config_file: "algorithm_3.yaml" # Путь к файлу конфигурации алгоритма 3

активен ли алгоритм
config_file: "algorithm_3.yaml" # Путь к файлу конфигурации алгоритма 3
```

3.1.1 Настройка mqtt

YAML-ключ	Обязательное / функциональное поле	Название	Допустимые значения	Значения по умолчанию
broker	Обязательное поле	URL брокера	tcp://, unix://, ssl://, mqtt:// Например: tcp://localhost:1883 ssl://127.0.0.1:8883 unix:///var/run/mosquitto/mosquitto.sock	tcp://localhost:1883
client_id	Необязательное	Идентификатор клиента	Строка	smartgate.topic_mapper
keepalive	Обязательное поле	Параметр keepalive в секундах	int32 в диапазоне [20;120]	30
username	Необязательное	Имя пользователя для подключения к MQTT-брокеру	Строка	—
password	Необязательное. При наличии mqtt.username - обязательное.	Пароль для подключения к MQTT-брокеру	Строка	—
max_retries	Необязательное	Количество повторов попыток подключения к брокеру	int32, которое строго больше 0	5000
retry_interval	Необязательное	Интервал между попытками подключения к брокеру	int32, которое строго больше 1	5
clean_session	Необязательное	Чистая сессия с брокером	true, false	true
autoreconnect	Необязательное	Переподключение к брокеру при потере соединения	true, false	true
ping_timeout	Необязательное	Таймаут ожидания ответа на сообщение keepalive	int32 в диапазоне: [1;60]	10
connect_timeout	Необязательное	Время ожидания установления первого соединения с брокером	int32 в диапазоне: [1;60]	30
pubQoS	Необязательное	QoS (Quality of Service) - гарантии доставки сообщений: QoS 0 ("хоть бы раз", доставка не гарантируется), QoS 1 (доставка "как минимум один раз", возможны дубли) и QoS 2	byte из: 0, 1, 2	0

		("ровно один раз", доставка без дубликатов)		
subQoS	Необязательно	QoS (Quality of Service) - гарантии доставки сообщений: QoS 0 ("хоть бы раз", доставка не гарантируется), QoS 1 (доставка "как минимум один раз", возможны дубли) и QoS 2 ("ровно один раз", доставка без дубликатов)	byte из: 0, 1, 2	0
retained	Необязательно	Сохранение топика в брокере	true, false	false
tls.enabled	Обязательное	Режим работы. Служит для включения, отключения безопасного подключения к MQTT-брокеру.	true, false	—
tls.insecure_skip_verify	Обязательное	Если задан true, то полностью отключает проверку подлинности сертификата сервера, позволяя приложению устанавливать соединение даже с самоподписанными, просроченными или недействительными сертификатами.	true, false	—
tls.ca_cert	Обязательное, если mqtt.tls.insecure_skip_verify задан false	Путь к файлу корневого сертификата удостоверяющего центра	Строка	""
tls.cert_file	Обязательное, если указан mqtt.tls.key_file	Путь к файлу клиентского сертификата для взаимной аутентификации	Строка	""
tls.key_file	Обязательное, если указан mqtt.tls.cert_file	Путь к файлу приватного ключа клиента	Строка	""

3.1.2 Настройки логирования

YAML-ключ	Обязательное/функциональное поле	Описание	Допустимые значения	Значения по умолчанию
level_log	Функциональное поле	Уровень логирования	int32 в диапазоне [1;5]	1
path_log	Функциональное поле	Путь к лог-файлу	Строка	""
path_json_log	Функциональное поле	Путь к JSON лог-файлу	Строка	""

<code>max_size_log</code>	Функциональное поле	Максимальный размер файла в мегабайтах	int32 в диапазоне [1;5]	2
<code>max_backups_log</code>	Функциональное поле	Максимальное количество резервных копий	int32 в диапазоне [1;300]	10
<code>max_age_log</code>	Функциональное поле	Максимальный возраст файлов в днях	int32 в диапазоне [1;300]	1

3.1.2.1 Приоритеты настройки логгера

Настройки логгера также можно задать через аргументы командной строки — раздел [Аргументы запуска](#). Ниже приведены правила приоритизации настроек для логгера.

- Конфигурация из файла имеет высший приоритет.
- Если файл не задан, используются параметры командной строки.
- Если ничего не задано, применяются значения по умолчанию.

3.1.3 Настройка алгоритмов

YAML-ключ	Обязательное/функциональное поле	Описание	Допустимые значения	Значения по умолчанию
<code>enabled</code>	Обязательное поле	Опция, позволяющая включить или отключить алгоритм	true, false	False
<code>config_file</code>	Обязательное поле	Путь к файлу конфигурации алгоритма	*.yaml	-

3.2 Валидация конфигурации

При загрузке конфигурации сервис проверяет, что все включённые алгоритмы можно безопасно запустить. Если проверка не проходит, сервис завершает загрузку конфигурации с ошибкой.

Проверяются следующие условия:

- Обязательные поля и типы значений:** Проверяется наличие обязательных секций и полей: настройки MQTT, логирования, `enabled`, `config_file`, `root_topic_mqtt`, `refresh_time`, списки `topics`, `mapping` и другие поля, необходимые выбранным алгоритмам.
- Корректность MQTT-топиков:** В топиках, где сервис ожидает конкретный топик, запрещены wildcard-символы `+` и `#`. Это относится к входным и выходным топикам `Algorithm1/Algorithm2`, а также к `root_topic_mqtt`.
- Корректность `refresh_time`:** Проверяется, что значение `refresh_time` задано в допустимом формате. Значение 0 отключает автопубликацию, положительное значение задаёт период автопубликации в секундах.

4. **Корректность файлов алгоритмов:** Если алгоритм включён, сервис загружает указанный `config_file` и проверяет его содержимое. Для Algorithm2 дополнительно проверяются все файлы, указанные в `topics[ ].topic_file`.
5. **Корректность правил Algorithm2:** Проверяется синтаксис `selector`, настройки `index_tag`, `bit_mapping` и выходные топики, в которые Algorithm2 будет публиковать результат.
6. **Коллизии входных топики:** Сервис проверяет, что один и тот же входной топик не используется несколькими обработчиками одновременно. Для Algorithm1 учитываются не только `topics[ ].in`, но и служебные подписки `abs(topics[ ].out) + "/on"`.
7. **Коллизии выходных топики:** Сервис проверяет пересечения выходных топики Algorithm1/Algorithm2. Несколько входов Algorithm1 могут публиковать в общий `or` или `and` топик, но остальные пересечения считаются ошибкой конфигурации.
8. **Рекурсия топики:** Сервис проверяет, что публикация одного правила не приводит обратно во входной топик этого же или другого правила. Проверка выполняется для Algorithm1 и Algorithm2. Algorithm3 в этой проверке не участвует.

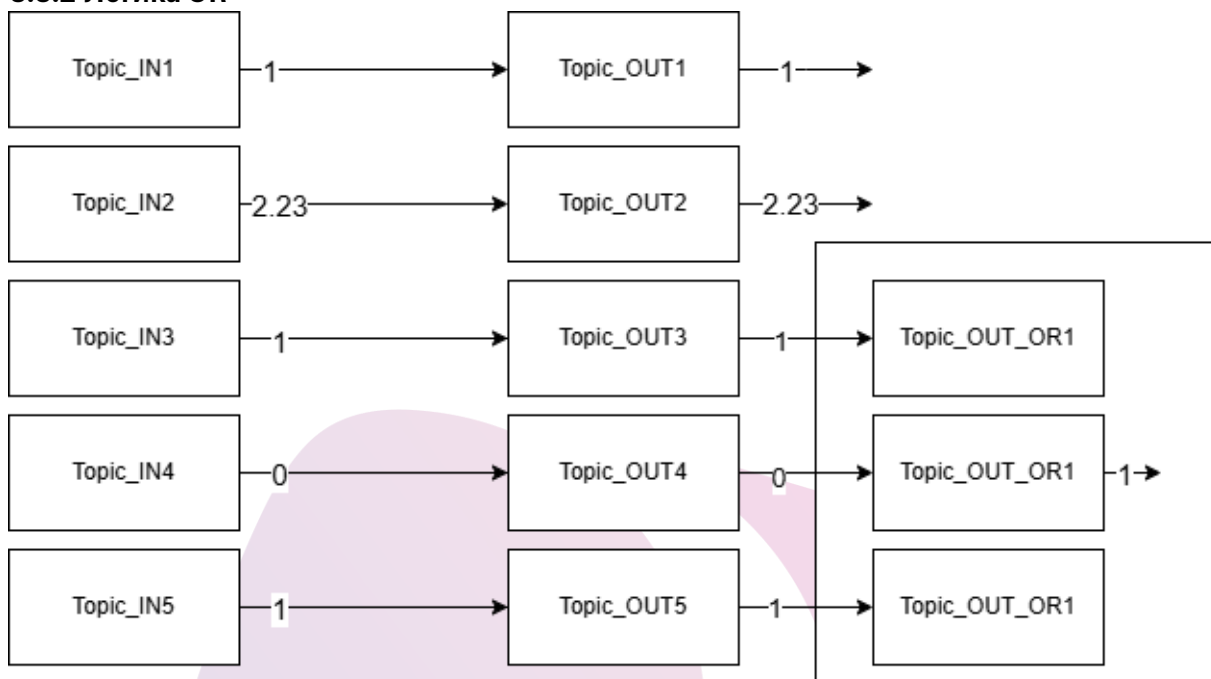
3.3 Алгоритм 1

3.3.1 Описание алгоритма

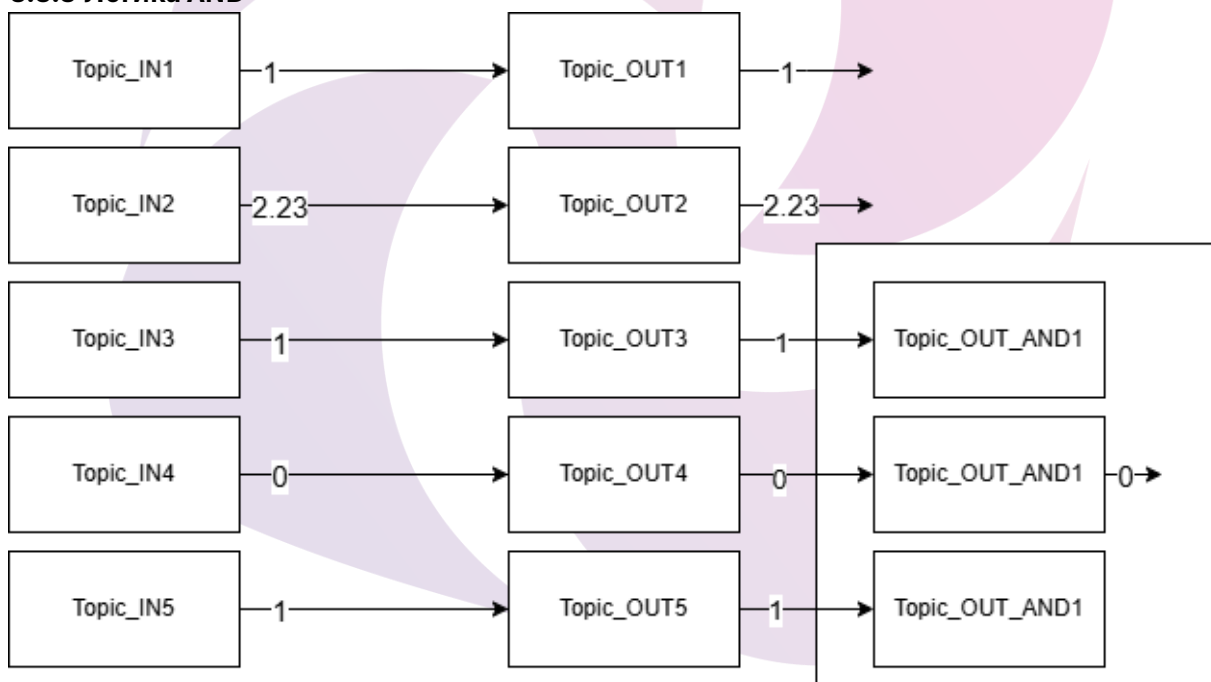
Данный алгоритм работает по следующей логике:

1. **Подписка на исходные топики:** сервис читает конфигурационный файл и подписывается на все исходные топики (`topic_in`), указанные в нем. Дополнительно для каждого заданного `topics.out` сервис подписывается на служебный топик `abs(topics.out) + "/on"`. Эта подписка используется для обратного ремапинга команд управления: сообщение, пришедшее в выходной топик с суффиксом `/on`, преобразуется обратно во входной топик Algorithm1.
2. **Обработка сообщений:** при получении сообщения из `topic_in` сервис извлекает его содержимое. Если в топике присутствует `"/meta/error"`, то пишется соответствующий лог, сообщение не публикуется. Содержимое сообщения перенаправляется в целевые топики (`topic_out`, `topic_out_or`, `topic_out_and`). Если указано несколько целевых топики, применяется логика OR или AND (например, для бинарных значений 0 и 1).
3. **Публикация сообщений:** сервис публикует преобразованные сообщения в указанные целевые топики через локальный MQTT-брокер.
4. **Автопубликация данных:** задает интервал в секундах, через который сервис будет проверять состояние топики и автоматически публиковать их значения, если изменения не были обнаружены. Это полезно для поддержания актуальности данных в системах, где топики могут не обновляться длительное время.
5. **Обратный ремапинг топики для управления.**

3.3.2 Логика OR



3.3.3 Логика AND



3.3.4 Настройка Алгоритма 1

YAML-ключ	Обязательное/ функциональное поле	Описание	Примеры
<code>root_topic_mqtt</code>	Обязательное поле	Корневой топик MQTT, используется для формирования целевого исходящего топика	elevator, TEST_TOPIC, bAf0r0_bAf0r0_0_n1_climate
<code>refresh_time</code>	Обязательное поле	Интервал в секундах, через который сервис будет проверять состояние топиков и автоматически публиковать их	30, 0

		значения, если изменения не были обнаружены. Это полезно для поддержания актуальности данных в системах, где топики могут не обновляться длительное время. Для отключения логики автопубликации - выставить 0.	
topics.in	Обязательное поле	Исходный топик, на который подписывается сервис.	/devices/ensystec_virtual/controls/GET_LEAK1
topics.out	Обязательное поле, если отсутствует topics.or, иначе опциональное	Целевой топик, в который перенаправляются сообщения.	/devices/b1f5r56_b1f5r56_30_n1_water-leak/controls/leak
topics.or	Обязательное поле, если отсутствует topics.out и topics.and, иначе опциональное	Если указано несколько целевых топиков, сервис поддерживает логику OR.	/devices/b1f5r56_b1f5r56_30_all1_water-leak/controls/leak
topics.and	Обязательное поле, если отсутствует topics.out и topics.or, иначе опциональное	Если указано несколько целевых топиков, сервис поддерживает логику AND.	/devices/b1f5r56_b1f5r56_30_all1_water-leak/controls/leak

3.3.5 Пример описания в YAML

```
root_topic_mqtt: test # Обязательное поле
refresh_time: 30      # Обязательное поле

topics: # Список топиков для обработки, обязательное поле
- in: /devices/Climate_Living_room/controls/OnOff
  out: /devices/b1f5r56_b1f5r56_42_n1_fancoil/controls/onoff
  and: /devices/b1f5r56_b1f5r56_30_all1_water-leak/controls/l
- in: /devices/Climate_Living_room/controls/Setpoint Temperature
  out: /devices/b1f5r56_b1f5r56_42_n1_fancoil/controls/setpoint_temperature
  and: /devices/b1f5r56_b1f5r56_30_all1_water-leak/controls/l
- in: /devices/ensystec_virtual/controls/GET_LEAK1
  out: /devices/b1f5r56_b1f5r56_30_n1_water-leak/controls/leak
  or: /devices/b1f5r56_b1f5r56_30_all1_water-leak/controls/leak
- in: /devices/ensystec_virtual/controls/GET_LEAK2
  out: /devices/b1f5r56_b1f5r56_30_n2_water-leak/controls/leak
  or: /devices/b1f5r56_b1f5r56_30_all1_water-leak/controls/leak
```

3.3.6 Пример топика /payload

```
топик in /devices/Climate_Living_room/controls/OnOff 1
топик out /devices/b1f5r56_b1f5r56_42_n1_fancoil/controls/onoff 1
```

3.3.7 Пример обработки управления

```
топик in /devices/Climate_Living_room/controls/OnOff
топик out /devices/b1f5r56_b1f5r56_42_n1_fancoil/controls/onoff
```

```
топик управления получаем /devices/b1f5r56_b1f5r56_42_n1_fancoil/controls/onof/on 2
ремapping топика управления публикуем /devices/Climate_Living_room/controls/OnOff/on2
```

3.4 Алгоритм 2

3.4.1 Описание алгоритма

Данный алгоритм работает по следующей логике:

1. **Подписка на исходные топики:** сервис читает конфигурационный файл и подписывается на все исходные топики (`topic`), указанные в нем. В текущей реализации Algorithm2 не создаёт служебных подписок вида `out_topic + "/on"`; подписка выполняется только на `topics[].topic` из корневого файла Algorithm2.
2. **Загрузка файлов конфигурации отдельных топиков:** сервис читает конфигурационные файлы(`topic_file`) и формирует правила преобразования/формирования исходящих топиков.
3. **Обработка сообщений:** при получении сообщения из `topic` сервис извлекает его содержимое в JSON формате. Содержимое сообщения по правилам из файла конфигурации перенаправляется в целевые топики (`out_topic`).
4. **Автопубликация данных:** задает интервал в секундах для каждого входного топика, через который сервис будет проверять состояние топика и автоматически публиковать его значения, если изменения не были обнаружены. Это полезно для поддержания актуальности данных в системах, где топики могут не обновляться длительное время.

3.4.2 Пример корневого файла конфигурации Алгоритма 2

```
topics:
- topic: "um_smart/d1"
  topic_file: "configs_features/um_smart_selector.yaml"
- topic: "um_smart/d2"
  topic_file: "configs_features/um_smart_selector_1.yaml"
- topic: "device/bytes"
  topic_file: "configs_features/device_bytes.yaml"
- topic: "device/status"
  topic_file: "configs_features/device_status_array.yaml"
```

3.4.3 Настройка Алгоритма 2

YAML-ключ	Обязательное/функциональное поле	Описание	Примеры
<code>topics.topic</code>	Обязательное поле	Исходный топик, на который подписывается сервис.	<code>/devices/multisensor/controls/Ua</code>
<code>topics.topic_file</code>	Обязательное поле	Файл конфигурации для определения правил формирования целевых(исходящих) топиков	<code>devices.multisensor.controls.Ua.yaml</code>

3.4.4 Примеры конфигурации `topics.topic_file`

Пример 1. Предположим, имеется следующий JSON payload:

```
{
```

```
"name": "UM SMART",
"serial": "2",
"fw": "1",
"measures": [
  {
    "measure": "mDIn",
    "devices": [
      {
        "model": "WBMR6",
        "serial": "1",
        "vals": [
          {
            "diff": 0,
            "tags": [
              { "tag": "Chn11", "val": false },
              { "tag": "Chn12", "val": false },
              { "tag": "Chn13", "val": false },
              { "tag": "Chn14", "val": true },
              { "tag": "Chn15", "val": false },
              { "tag": "Chn16", "val": true },
              { "tag": "Chn17", "val": false },
              { "tag": "Chn18", "val": false },
              { "tag": "Chn19", "val": false },
              { "tag": "Chn110", "val": false },
              { "tag": "Chn111", "val": false },
              { "tag": "Chn112", "val": false },
              { "tag": "Chn113", "val": false }
            ],
            "ts": "2025-09-10T18:25:03+03:00"
          }
        ]
      },
      {
        "id": 1,
        "type": 193
      }
    ]
  }
]
```

Предположим, для данного payload необходимо для каждого элемента массива "measures.devices.vals" извлечь поле "val". При том, что:

- measure == mDIn;
- model == WBMR6.

Тогда, конфигурация примет вид:

```
root_topic_mqtt: TEST_TOPIC
```

```
refresh_time: 20
```

```
mapping:
```

```
  parameters:
```

- selector: measures[measure=mDIn].devices[model=WBMR6].vals[0].tags[tag=Chnl1].val
out_topic: "/devices/DI_24V_X8/controls/valsChnl1"
- selector: measures[measure=mDIn].devices[model=WBMR6].vals[0].tags[tag=Chnl2].val
out_topic: "/devices/DI_24V_X8/controls/valsChnl2"
- selector: measures[measure=mDIn].devices[model=WBMR6].vals[0].tags[tag=Chnl3].val
out_topic: "/devices/DI_24V_X8/controls/valsChnl3"
- selector: measures[measure=mDIn].devices[model=WBMR6].vals[0].tags[tag=Chnl4].val
out_topic: "/devices/DI_24V_X8/controls/valsChnl4"
- selector: measures[measure=mDIn].devices[model=WBMR6].vals[0].tags[tag=Chnl5].val
out_topic: "/devices/DI_24V_X8/controls/valsChnl5"
- selector: measures[measure=mDIn].devices[model=WBMR6].vals[0].tags[tag=Chnl6].val
out_topic: "/devices/DI_24V_X8/controls/valsChnl6"
- selector: measures[measure=mDIn].devices[model=WBMR6].vals[0].tags[tag=Chnl7].val
out_topic: "/devices/DI_24V_X8/controls/valsChnl7"
- selector: measures[measure=mDIn].devices[model=WBMR6].vals[0].tags[tag=Chnl8].val
out_topic: "/devices/DI_24V_X8/controls/valsChnl8"

Пример 2. Предположим, имеется следующий JSON payload:

```
{
  "device": {
    "data": {
      "bytes": [10, 20, 30, 40, 3, 60]
    }
  }
}
```

Предположим, в данном payload каждый элемент массива "device.data.bytes" отвечает за параметры:

- device.data.bytes[0] == param_A;
- device.data.bytes[1] == param_B;
- device.data.bytes[2] == param_C;
- device.data.bytes[3] == param_D;
- device.data.bytes[5] == param_E.

Пропуская 4-й индекс (5-й элемент массива). Тогда, конфигурация примет вид:

```
root_topic_mqtt: root
refresh_time: 15

mapping:
```

```
parameters:
- selector: device.data.bytes
  index_tag:
    - index: 0
      out_topic: "/devices/controls/Param_A"
    - index: 1
      out_topic: "/devices/controls/Param_B"
    - index: 2
      out_topic: "/devices/controls/Param_C"
    - index: 3
      out_topic: "/devices/controls/Param_D"
    - index: 5
      out_topic: "/devices/controls/Param_E"
```

3.4.5 Настройка маппинга Алгоритма 2 (topic.topic_file)

YAML-ключ	Обязательное/ функциональное поле	Описание	Примеры
<code>root_topic_mqtt</code>	Обязательное поле	Корневой топик MQTT	Строка. Например: "/devices", "root", "a/b/c"
<code>refresh_time</code>	Обязательное поле	Время гарантированной публикации данных. 0 - для отключения	Целое беззнаковое число. Например: 0, 10, 100
<code>mapping</code>	Обязательное поле	Представляет собой массив из selector (см. п. 3.3.11) с дополнительными полями для маппинга.	Смотреть далее

3.4.6 Настройка mapping

YAML-ключ	Обязательное/функциональное поле	Описание	Примеры
<code>selector</code>	Обязательное поле	Представляет собой путь (со всеми возможными фильтрами) до целевого параметра (DSL) в JSON структуре. Подробнее см. п.3.3.11 Альтернативно формировать путь можно с помощью сайта: https://jsonpathfinder.com/ Необходимо убрать "х." из значения поля "Path"	Строка. Например: Пример 1: <code>selector: type.data.value</code> Пример 2: <code>selector: tag1</code> Пример 3: <code>selector: mods[name="DI 24V X8"].val[name="81"].val</code> Пример 4: <code>selector: measures[measure=mDIn].devices[model=WBM6].vals[0].tags[tag=Chnl1].val</code> Альтернативно, пример 5: <code>selector: measures[0].devices[0].vals[0].tags[0].val</code> ВАЖНО! Поскольку, порядок и содержимое JSON формируется на стороне отправителя использовать фильтр полей, а не индексы., то фильтры полей являются более надёжными (применимо).

index_tag	Функциональное поле	Представляет собой массив из index (см п.3.3.8) используемых для маппинга числовых массивов JSON.	См. п.3.3.8
out_topic	Обязательное поле, если отсутствует функциональное поле index_tag	Выходной топик	Строка. Например: "/devices/DI_24V_X8/controls/vals81"; "/devices/lb1_elevator/controls/negative_floors"; "/topic/data/controls/Value1";

3.4.7 Пример конфигурации index_tag

Предположим, имеется следующий JSON payload:

```
{  
  "t": "2022-02-20T11:14:12Z",  
  "tag1": [0,10,5,4,255,2,31,2,0,15,7,0,0,0,0]  
}
```

В данном payload поле "tag1" представляет собой массив из целых чисел. Индексация начинается от 0.

Предположим:

- Индекс 0 - 3 являются показателями;
- Каждый бит индекса 4 отвечает за определенное состояние.

Необходимо значение в каждом из индексов маппить в соответствующие выходные топики, а для индекса 4 необходимо маппить биты в соответствующие выходные топики, тогда конфигурация будет иметь вид:

```
root_topic_mqtt: root  
refresh_time: 30  
  
mapping:  
  parameters:  
    - selector: tag1  
      index_tag:  
        - index: 0  
          out_topic: "/devices/lb1_elevator/controls/negative_floors"  
        - index: 1  
          out_topic: "/devices/lb1_elevator/controls/floor"  
        - index: 2  
          out_topic: "/devices/lb1_door/controls/state"  
        - index: 3  
          out_topic: "/devices/lb1_door/controls/presure"  
        - index: 4  
          out_topic: "/devices/lb1_door/controls/door_state"  
          bit_mapping:  
            "0":  
              description: "открываются"  
              out_topic: "/devices/lb1_door/controls/door_opening"  
              payload:  
                if_set: 1  
                if_not_set: 0  
            "1":  
              description: "открыты"  
              out_topic: "/devices/lb1_door/controls/door_open"  
              payload:
```

```
    if_set: 1
    if_not_set: 0
    # и так далее
- index: 5
  out_topic: "/devices/lb1_door/controls/index5"
# и так далее
```

В результате, для данного payload будет формироваться топики со значениями, соответствующим значениям из массива JSON.



3.4.8 Настройка index_tag

YAML-ключ	Обязательное/ функциональное поле	Описание	Примеры
<code>index</code>	Обязательное поле	Индекс элемента массива внутри JSON. Нумерация идет с нуля.	Целое число. Например: 0, 1, 2, 3, ...
<code>out_topic</code>	Обязательное поле, в случае наличия bit_mapping является функциональным	Выходной топик	Строка. Например: "/devices/DI_24V_X8/controls/vals81"; "/devices/lb1_elevator/controls/negative_floors"; "/topic/data/controls/Value1";
<code>bit_mapping</code>	Функциональное поле	Представляет собой маппу, которая описывает маппинг n бита. Нумерция битов начинается с 0. Начиная с версии 2.3.0.0 нумерация с 0. До 2.3.0.0 нумерация с 1.	1st bit: <pre>bit_mapping: "0": description: "открываются" # необязательное поле out_topic: "/devices/lb1_door/controls/door_opening" # обязательное поле payload: # необязательное поле if_set: 1 # обязательное поле if_not_set: 0 # обязательное поле</pre> 3rd bit: <pre>bit_mapping: "2": out_topic: "/devices/lb1_door/controls/door_opening"</pre> Подробнее см. п.3.3.9 «Настройка bit_mapping»

3.4.9 Настройка bit_mapping

YAML-ключ	Обязательное/ функциональное поле	Описание	Примеры
description	Функциональное поле	Описание, нигде не используется на данный момент.	Строка. Например: "Дверь открыта", "Дверь открывается"
out_topic	Обязательное поле	Выходной топик	Строка. Например: "/devices/DI_24V_X8/controls/vals81"; "/devices/lb1_elevator/controls/negative_floors"; "/topic/data/controls/Value1";
payload	Функциональное поле	Настройка значений payload выходных топиков (см. п.3.3.10)	Example 1: <pre>payload: if_set: 1 # обязательное поле if_not_set: 0 # обязательное поле</pre> Example 2: <pre>payload: if_set: "on" # обязательное поле if_not_set: "off" # обязательное поле</pre>

3.4.10 Настройка payload

YAML-ключ	Обязательное/функциональное поле	Описание	Примеры
<code>if_set</code>	Обязательное поле	Если данный бит замкнут в true(1), то для данного выходного топика будет использован указанный в этом поле payload.	0, 1, "on", "off", 100.1, ...
<code>if_not_set</code>	Обязательное поле	Если данный бит замкнут в false(0), то для данного выходного топика будет использован указанный в этом поле payload.	1, 0, "off", "on", 100.1, ...

3.4.11 Selector (селектор)

Селектор описывает путь к данным в JSON: ключи через точку, индексы массивов в квадратных скобках, фильтры в массивах, кавычки для ключей с пробелами.

Например:

Пример 1:

```
selector: measures[measure=mDIn].devices[model=WBM6].vals[0].tags[tag=Chnl4].val
```

Пример 2:

```
selector: measures[0].devices[0].vals[0].tags[0].val
```

Пример 3:

```
selector: mods."DI 24V X1".val[11].val
```

Пример 4:

```
selector: device.data
```

Пример 5:

```
selector: tag1
```

3.4.12 Правила и ограничения по пользованию Selector

Раздел	Правило / ограничения	Пример (ОК)	Пример (ошибка)
Путь/сегменты	Нет ведущей/двойной/закрывающей точки	a.b.c	.a, a..b, a.
Ключ (без кавычек)	<code>^[A-Za-z0-9_\-]+</code>	measures	di 24

Ключ (в кавычках)	Любые символы, парные " и экранирование	"DI 24V X1"	"DI 24V X1
Пробелы	Вне кавычек запрещены	arr[field=1]	arr[field = 1]
Индекс массива	Целое ≥ 0	vals[0]	vals[-1]
Фильтр — поле	Без пробелов или в кавычках	arr["model name"]=X]	arr[model name=X]
Фильтр — значение	number/boolean/"string"	id=10, flag=true, tag="Chnl1"	id=01e
Сравнение в фильтре	Строгое, без приведения типов	flag=true	flag="true"
Вложенные []	Допускается последовательность	items[2][kind="A"][0]	items[][0]

3.4.13 Синтаксис Selector'a

Элемент DSL	Описание	EBNF (фрагмент)	Пример
Путь	Сегменты через '.'	Path = Segment { "." Segment }	a.b.c
Сегмент	[Ключ]{Массив}	Segment = [Key\QuotedKey] Postfix	measures[0]
Ключ	Без пробелов/точки	Key = KeyStart {KeyChar}	devices
Ключ в кавычках	Любые символы (экранирование)	QuotedKey = " {QChar} "	"DI 24V X1"
Индекс	0..N	Index = DIGITS	vals[0]
Фильтр	field=value	Filter = (Key\QuotedKey) "=" FilterValue	devices[model=WBM6]
Число	int/float/exp	Number = [-]Int[.Frac][Exp]	1, -2, 3.14, 1e-3
Булево	true/false	Boolean = "true" \ "false"	flag=true
Строка	С экранированием	QuotedValue = " {QChar} "	tag="Chnl1"

3.5 Алгоритм 3

3.5.1 Описание алгоритма

Топик публикуемый согласно конфигурации на WB

```
/devices/split_bB101f-1r135_bB101f-1r135_15/controls/bB101f-1r135_bB101f-1r135_33_n3_pump_privet:alarm_km 99
```

Топик захваченный и отформатированный сервисом

```
/devices/bB101f-1r135_bB101f-1r135_33_n3_pump_privet/controls/alarm_km 99
```

Топик Управления из АС Андромеда

```
/devices/bB101f-1r135_bB101f-1r135_33_n3_pump_privet/controls/alarm_km/on 1000
```

Топик захваченный и отформатированный сервисом для успешного управления на WB

```
/devices/split_bB101f-1r135_bB101f-1r135_15/controls/bB101f-1r135_bB101f-1r135_33_n3_pump_privet:alarm_km/on 1000
```

3.5.2 Общая структура топиков Изначальная структура топиков:

/devices/split_(A)/controls/(B):(C)

После преобразования формируется новая структура:

/devices/(B)/controls/(C)

Где:

- **A** — идентификатор канала (расположение устройства и зона).
- **B** — уникальный идентификатор параметра (включает расположение устройства и наименование).
- **C** — имя параметра.

3.5.3 Пример. Сложная структура с полным идентификатором устройства

Входящий топик:

«/devices/split_bB101f-1r135_bB101f-1r135_15/controls/bB101f-1r135_bB101f 1r135_33_n3_pump:alarm_km».

Разбиение на части:

- **A:** bB101f-1r135_bB101f-1r135_15 (идентификатор канала).
- **B:** bB101f-1r135_bB101f-1r135_33_n3_pump (уникальный идентификатор параметра).
- **C:** alarm_km (имя параметра).

Результат преобразования:

«/devices/bB101f-1r135_bB101f-1r135_33_n3_pump/controls/alarm_km».

3.5.4 Пример. Структура с сокращенным идентификатором параметра

Входящий топик:

«/devices/split_b1f-1r31_b1f-1r31_15:2/controls/n4_pump:oper_time».

Разбиение на части:

- **A:** b1f-1r31_b1f-1r31_15:2 (идентификатор канала).
- **B:** n4_pump (идентификатор параметра).
- **C:** oper_time (имя параметра).

Результат преобразования:

«/devices/b1f-1r31_b1f-1r31_15_n4_pump/controls/oper_time».

3.5.5 Дополнительный идентификатор

Входящий топик:

/devices/split_test1/controls/n101_mode:set

Результат:

/devices/test1_n101_mode/controls/set

3.5.6 Пример описания YAML

```
refresh_time: 5

patterns:
- "/devices/#"
- "$SYS/#"
```

3.5.7 Настройка Алгоритма 3

YAML-ключ	Обязательное/функциональное поле	Описание	Примеры
refresh_time	Обязательное поле	Интервал в секундах, через который сервис будет проверять состояние топиков и автоматически публиковать их значения, если изменения не были обнаружены. Это полезно для поддержания актуальности данных в системах, где топики могут не обновляться длительное время. Для отключения логики автопубликации - выставить 0. Обязательное поле.	30, 0
patterns	Обязательное поле (минимум 1 шаблон)	Шаблоны (wildcard) топиков, на которые сервис подпишется в рамках 3-го алгоритма.	"/devices/#", "#", "\$SYS/#"

4. Демо-режим

Демоверсия Topic Mapper предназначена для ознакомления с возможностями ПО. При запуске демоверсии в логах указывается режим сборки DEMO, а для конфигурации действуют следующие ограничения:

Ограничение	Значение
Максимальное количество уникальных выходных MQTT-топиков	30
Автопубликация (refresh_time)	Недоступна. Во всех включённых алгоритмах refresh_time должен быть равен 0
mqtt.client_id	Нельзя задавать в YAML-конфигурации. Используется значение по умолчанию smartgate.topic_mapper

5. Аргументы запуска

Флаг	Флаг (короткая версия)	Тип принимаемого значения	Стандартное значение	Описание
<code>--cfgpath</code>	-c	String (строка)	configs/mapper/topic_mapper.yaml	Путь к корневому файлу конфигурации
<code>--path_log</code>	-p	String (строка)	/mnt/data/etc/andromeda_embedded/logs/mapper/topic_mapper_2_0.log	Путь до файла логов
<code>--level_log</code>	-l	Int (число)	1	Уровень логирования сервиса (1-3)
<code>--version</code>	-	-	-	Вывод версии и завершение программы.

Лист внесения изменений

Версия документа	Дата публикации	Авторы изменений	Описание изменений
rev_02	27.10.2025	Тимонин А. А.	Скорректировано оформление
rev_03	19.11.2025	Тимонин А. А.	Документ обновлен полностью. Добавлен Алгоритм 3. Расширено описание Алгоритмов 1 и 2
rev_04	05.02.2026	Тимонин А.А.	Документ обновлен в соответствии с новым релизом Topic Mapper
rev_05	23.07.2026	Тимонин А.А.	Добавлено: 1. Раздел "Валидация конфигурации" 2. Раздел "Демо-режим" 3. Доп информация по алгоритмам 1 и 2 : Алгоритм 1 скрыто подписывается на служебный топик <code>abs(topics.out) + "/on"</code> . Алгоритм 2 не создает служебных подписок вида <code>out_topic + "/on"</code> и подписывается только на <code>topics[].topic</code> 4. Раздел "Приоритеты настройки логгера"