

# API

- Изменение статуса мультилида
- Изменение статуса заказа
- Общая информация
- Передача бонусных баллов

# Изменение статуса мультилида

Адрес: **[https://track-api.leadhit.io/client/update\\_multilead/](https://track-api.leadhit.io/client/update_multilead/)**

## Описание

Изменение статуса мультилида и всех привязанных к нему лидов.

Может пригодиться, если нужно ограничить отправку писем определенным лидам, таким лидам можно выставить отличный от дефолтного статус, а в условиях рассылок запретить/разрешить отправку писем лидам с данным статусом.

## Параметры

Метод - POST

В заголовках нужно передать "api-key" и "leadhit-site-id".

Параметры - json с полями "email", "status", "site\_id".

Статус должен быть одним из списка: **"new", "contacted", "customer", "manager", "regular", "wholesale"**.

По умолчанию у мультилида установлен статус "new".

Api-key и site-id предоставляем мы.

## Пример использования

```
POST https://track-api.leadhit.io/client/update_multilead/
```

### Headers (application/json):

```
api-key: <api-key>
leadhit-site-id: <your_site_id>
```

### Body (application/json):

```
{
  "email": "example@mail.ru",
```

```
"status": "contacted",
"site_id": "<your_site_id>"
}
```

#### **Response 200 (application/json):**

```
{
  "status": "success",
  "message": "Multilead with email example@mail.ru was updated. New status - contacted."
}
```

## Возможные ошибки

Если покупатель с таким email не найден:

#### **Response 404 (application/json):**

```
{
  "detail": "Multilead not found"
}
```

## Пример запроса с помощью CURL:

```
curl --request POST \
  --url https://track-api.leadhit.io/client/update_multilead/ \
  --header 'Content-Type: application/json' \
  --header 'api-key: <api-key>' \
  --header 'leadhit-site-id: <your_site_id>' \
  --data '{
    "email": "example@mail.ru",
    "status": "contacted",
    "site_id": "<your_site_id>"
  }'
```

# Изменение статуса заказа

Адрес: **https://service.leadhit.ru/api/v1/orders**

Метод: **set\_order\_status**

## Описание

Изменение статусов заказов можно использовать для настройки триггерных рассылок.

## Параметры

Метод: **POST**

Заголовки: **api-key**

Параметры:

- **JSON** с полями **"order\_id"**, **"status"**;
- Версия JSON-RPC: **2.0**;
- ID запроса: **0**.

Статус должен быть одним из списка: **"submitted"**, **"paid"**, **"delivered"**, **"cancelled"**.

После получения нами заказа, ему выставляется статус "submitted" - оформлен.

API-key запросите у нас.

## Пример использования

```
POST https://service.leadhit.ru/api/v1/orders
```

### Headers (application/json):

```
api-key: <api-key>
```

### Body (application/json):

```
{
  "method": "set_order_status",
  "params": {
    "order_id": "f168148717",
    "status": "paid"
  },
  "jsonrpc": "2.0",
```

```
"id": 0
}
```

### Response 200 (application/json):

```
{
  "jsonrpc": "2.0",
  "result": {
    "status": "ok",
    "message": "Order status changed"
  },
  "id": 0
}
```

## Возможные ошибки

- Если номер заказа не найден:

### Response 200 (application/json):

```
{
  'message': "Order with this id doesn't exist"
}
```

- Если неправильно указан статус заказа:

### Response 200 (application/json):

```
{
  'message': 'Possible statuses: cancelled, delivered, paid, submitted'
}
```

## Пример запроса с помощью CURL:

```
Curl:
```sh
$ curl -i \
  -H "Accept: application/json" \
  -H "Content-Type: application/json" \
  -H "API-KEY: 4f8425fc77796b0266000009: TKW0i5FP5vkIrgMlnpoZbf0EagA" \
  -X POST -d
'{"method": "set_order_status", "params": {"order_id": "f168148717", "status": "paid"}, "jsonrpc": "2.0", "id": 0}' \
```

```
https://service.leadhit.ru/api/v1/orders/
```

```
>>> {"jsonrpc": "2.0", "result": {"status": "ok", "order_status": "paid"}, "id": 0}
```

# Общая информация

## Терминология

- **Узел** - конечный URL, к которому отправляются запросы. Например - `https://service.leadhit.ru/api/v1/orders/`.
- **Методы** - методы, которые принимают узлы, являются обычными питоновскими функциями.

## API

API располагается по адресу <https://service.leadhit.ru/api/v1/>

- Основной протокол - JSON-RPC 2.0
- Content-type - application/json

## Карта методов

У каждого узла API есть специальная страница - карта методов. На этой странице можно увидеть все существующие методы данного узла, а также дополнительную информацию о них.

Ее можно найти, если выполнить GET-запрос (или зайти через браузер), добавив к интересующему вас URL узла в конце `/map`.

Пример карты методов:

```
https://service.leadhit.ru/api/v1/orders/map
```

## Авторизация

Чтобы пользоваться API, сначала необходимо получить API key. Обратитесь к своему менеджеру за ним.

После того, как вы получили ключ, вам необходимо передать его в заголовке POST-запроса к API. Он должен иметь название API-KEY (или api-key).

В случае, если API не получило авторизационный ключ в заголовке, вы получите ответ 403.

# Базовый запрос

Общение с API происходит по протоколу JSON-RPC 2.0, через POST-запросы.

Content-type должен быть `application/json`.

Не забывайте об авторизации (заголовок API-KEY).

Помимо этого, каждый запрос к API должен содержать в себе три обязательных параметра - метод, версия JSON-RPC, id запроса. JSON-RPC всегда равен 2.0, а id может быть любым, такой же id вернется в ответе на запрос:

```
"method": "example_method"
"jsonrpc": "2.0"
"id": 0
```

Помимо обязательных параметров, методы принимают аргументы, которые передаются через "params".

Могут быть либо списком, либо словарем.

В качестве примера - метод **set\_order\_status**.

Запрос:

```
{
  "method": "set_order_status",
  "params": {
    "order_id": "f168148717",
    "status": "paid"
  },
  "jsonrpc": "2.0"
  "id": 0
}
```

Ответ:

```
{
  "jsonrpc": "2.0",
  "result": {
    "status": "ok",
    "message": "Order status changed"
  },
  "id": 0
}
```



```
}
```

# Ошибки

В случае, если вы не авторизованы, сервер вернет статус 403.

Если вы ошиблись в параметре, или произошла какая-то другая ошибка, сервер вернет статус 200, а в теле ответа будет содержаться ключ **error**, в котором описана причина ошибки.

```
{
  "jsonrpc": "2.0",
  "id": 0,
  "error": {
    "message": "Server error",
    "code": -32000,
    "data": {
      "message": "Possible statuses: paid, submitted",
      "args": [
        "Possible statuses: paid, submitted"
      ],
      "type": "ValidationError"
    }
  }
}
```

# Передача бонусных баллов

## Передача данных

Данные о количестве бонусов покупателя передаются клиентом через API по адресу [https://track-api.leadhit.io/client/multilead\\_bonus](https://track-api.leadhit.io/client/multilead_bonus)

Метод - POST

В **заголовках** нужно передать "api-key" и "leadhit-site-id" со значениями API ключа и id сайта. Значение для **api-key** запросите у своего аккаунт-менеджера.

**Параметры** - json с полями "email", "bonus". В поле `bonus` передается количество бонусов покупателя, в формате `float`.

### Заголовок запроса:

```
api-key: <api-key>
leadhit-site-id: <your_site_id>
```

### Тело запроса:

```
[{
  "email": "exists_lead@mail.ru",
  "bonus": 123.00
},
{
  "email": "not_exists_lead@mail.ru",
  "bonus": 10.00
}]
```

## Примеры ответов на запрос

### Response 200 (application/json)

При успешной передаче списка будет выдаваться статус 200 с ответом в виде списка со статусами по каждому обновлению бонусов у лида. Все обновления, которые не были сделаны будут в конце списка со статусом "error":

```
[{
  "status": "success",
  "message": "Bonuses for multilead with email exists_lead@mail.ru updated."
},
{
  "status": "error",
  "message": "multilead with email not_exists_lead@mail.ru not found"
}]
```

### Response 400 (application/json):

```
{
  "detail": "Body is empty"
}
```

При передачи списка с любыми невалидными данными будет выдаваться ошибка 422 с описанием полей, где данные некорректны

### Response 422 (application/json)

```
{
  "detail": [
    {
      "loc": ["string"],
      "msg": "string",
      "type": "string"
    }
  ]
}
```

### Response 500 (application/json)

При получении ошибки 500 необходимо проверить данные на корректность и повторить запрос через несколько минут. Если ошибка повторяется, то уведомить с примерами данных, с которыми возникает ошибка для последующего анализа

```
{
  "detail": "Internal error"
}
```