API manual

Версия 1.09

API предназначен для приёма и отправки сообщений, получения и редактирования информации о клиентах через сервис Nova.Chats.

Термины

Клиент – лицо, которое общается с вашей компанией с помощью сервиса Nova.Chats, используя мессенджеры или SMS.

Канал – аккаунт (телефонный номер или id) в мессенджерах WhatsApp, Viber, Telegram, ВКонтакте, Facebook или Онлайн-чат, номер для SMS-сообщений или какой-либо *внешний канал*, используемые вашей компанией для переписки со своими клиентами посредством сервиса Nova.Chats.

Внешний канал – это канал, напрямую не подключённый к системе Nova.Chats, например, ваш Онлайн-чат, ваш собственный мессенджер, ваш бот, e-mail и т.п. Такой внешний канал подключается к системе через API.

Транспорт – мессенджер или SMS, посредством которого отправляются или принимаются сообщения по данному каналу.

Общие положения

1. Для тестирования API рекомендуем использовать приложение *Postman* для Google Chrome. Вот ссылка на коллекцию готовых API-команд, уже настроенных на Nova.Chats от имени тестового аккаунта (импортировать в Postman):
   https://www.getpostman.com/collections/c598e71f85fd72404ff6
2. Используется архитектура RESTful, методы GET, PUT и POST.
3. Результаты команд возвращаются в формате JSON.
4. API работает через протокол HTTPS.
5. Авторизация осуществляется посредством `<token>`, доступного в настройках API. Параметры для всех команд:

```
Authorization: <token>
Host: api.chat24.io
```

6. Команды, возвращающие большие списки, поддерживают постраничный вывод с использованием параметров в URL:

   a. `limit` – количество возвращаемых записей (по умолчанию – 20).

   b. `offset` – смещение относительно 1-й записи списка (по умолчанию – 0).

   Например (список клиентов): `/v1/clients?offset=50&limit=30`

7. Вообще говоря, система спроектирована таким образом, что вы можете отправить сообщения только в ответ клиентам, которые хотя бы раз сами написали вам. Однако, с помощью clients (POST) вы можете создать нового клиента и затем написать ему.

8. Поддерживается 2 способа получения сообщений от клиентов:

   - Выгрузка накопившихся сообщений с помощью messages (GET).
   - Обработка события о новом сообщении с помощью web_hook (POST). Этот способ более предпочтителен.

Прежде чем вы начнёте

1. Получите ваш `<token>` на сайте Nova.Chats в разделе *Настройки* > *API*. Там же убедитесь, что ваш уровень доступа к API как минимум *demo*. Эта информация доступна и с помощью api_info (GET).

2. Существует 2 способа работы системы:

   a. Ваши клиенты используют мессенджеры для коммуникации напрямую с аккаунтами чат-центра. В таком случае для сбора сообщений используйте

   messages (GET) или web_hook (POST). Для отправки - clients (POST) и messages (POST).

   b. Ваши клиенты используют для коммуникаций *внешний канал* (см. Термины). В таком случае для передачи сообщений от клиента в систему используйте messages/inbox (POST).

Для передачи сообщений от системы к клиенту используйте web_hook (POST), на который будут передаваться соответствующие сообщения.

При этом поле `<transport>` примет значение *external*.

Список API-команд

- messages (GET)

  Возвращает накопившийся список сообщений (как от клиентов, так и к клиентам).
- web_hook (POST)

  Задаёт URL для web hook о поступлении сообщения от клиента, от системы (при использовании *внешнего канала*), а также при установке и удалении тегов клиенту.
- chat_close_web_hook (POST)

  Задаёт URL для web hook о закрытии чата.
- messages (POST)

  Отправляет сообщение клиенту.
- messages/inbox (POST)

  Отправляет сообщение в систему от клиента, используя *внешний канал*.
- messages/<id>/transfer (GET)

  Назначает сообщение и диалог на указанного оператора.
- api_info (GET)

  Возвращает текущую версию API, ваш уровень доступа к API, ваш web_hook (POST), количество API-запросов в месяц и др.
- clients (GET)

  Возвращает список ваших клиентов с информацией по каждому клиенту.
- clients (POST)

  Создаёт нового клиента чтобы отправить ему сообщение.
- clients (PUT)

  Назначает клиенту имя.
- operators (GET)

  Возвращает список ваших операторов в системе.
- operators_groups (GET)

Возвращает список групп операторов в системе.

- transfer_to_group (GET)
  Переводит (назначает) диалог на группу операторов.
- dialogs (GET)
  Возвращает список ваших диалогов в системе.
- dialogs (PUT)
  Изменяет состояние диалога и задаёт ведущего его оператора.
- tags (GET)
  Возвращает ваш список тегов.
- tags (POST)
  Создает новый тег.
- tag_groups (POST)
  Создает новую группу тегов.
- tags/assign_to (POST)
  Присваивает теги клиенту.
- tags (DELETE)
  Удаляет тег клиента.
- templates (GET)
  Возвращает список ваших шаблонов для быстрых ответов.
- qr-decode (POST)
  Декодирует QR-код.
- roles (GET)
  Возвращает имеющиеся роли в системе.
- dialog_states (GET)
  Возвращает имеющиеся в системе состояния диалогов.
- channels (GET)
  Возвращает список ваших каналов.
- regions (GET)
  Возвращает список имеющихся в системе регионов.
- countries (GET)
  Возвращает список имеющихся в системе стран.
- transports (GET)
  Возвращает список транспортов в системе и поддерживаемых типов вложений.
- messages/read (GET)
  Помечает сообщение как прочитанное оператором.

- api_modes (GET)
  Возвращает имеющиеся уровни доступа к API.

Дополнительные команды

- channels/<id>/clients (GET)
  Возвращает список клиентов по указанному каналу.
- questions/<id>/send (GET)
  Отправляет клиенту указанный пункт меню при использовании скриптов (см. *Справка>Описание скриптов*).
- clients/<id>/questions (GET)
  Запрос:

  ```
  GET /v1/clients/<id>/questions?start_date=10-09-2011&finish_date=10-09-2018
  ```

  Возвращает список пунктов меню за указанный диапазон дат (опционально), которые были ранее отправлены клиенту при использовании скриптов (см. *Справка> Описание скриптов*).
- delete_outbox (GET)

  ```
  GET /gateway/delete_outbox
  ```

  Запрос:

  Удаляет все сообщения на отправку, которые еще не были отправлены системой (например, по техническим причинам).

messages (GET)

Возвращает накопившийся список сообщений (как от клиентов, так и к клиентам). Для транспорта Viber business и public доступен статус отправленного клиенту сообщения.

Внимание! Мы не рекомендуем использовать эту команду для получения вх. сообщений, т.к. это ведет к большому количеству ненужных запросов и, соответственно, завышенным затратам. Используйте `web_hook (POST)`.

Запрос:

```
GET /v1/messages
или
GET /v1/messages/<id>
```

Параметры:

- `<id>` – id сообщения.

  Без указания `<id>` команда вернёт входящие и исходящие сообщения списком. При указании `<id>` команда вернёт *расширенную информацию* по одному указанному сообщению. В этом случае дополнительно будет возвращены поля `dialog_id`, `operator_id` и `channel_id`.

При запросе списком (без указания `<id>`) поддерживается фильтрация списка по полям.

- `transport`
- `channel_id`
- `client_id`
- `type` (*`to_client` или `from_client`*)
- `dialog_id`
- `read` (прочитано или нет сообщение оператором)

Пример запроса списком с фильтром:

```
GET
/v1/messages?transport=telegram&type=from_client&read=false
%client_id=100
```

Типовой ответ при запросе списком (без указания <id>):

```
        {

          "id": 614,
          "text": "Добрый день. Не
    работает роутер - см. фото",
          "photo":
    "http://chat24.io/images/users/message/_19-
    45-03.jpg",
          "coordinates": null,
          "transport": "whatsapp",
          "type": "from_client",
          "read": true,
          "created": "2016-01-
    11T19:45:05 UTC",
          "client_id": 13,
          "vb_bus_id":
    "597110513645559990912348615203629047737",
          "recipient_status":
    "delivered",
        }
      ],
      "meta": {
        "total": 617,
        "limit": 20,
        "offset": 610
      },
      "status": "success"
    }
```

Типовой ответ *с расширенной информацией о сообщении* при запросе одного сообщения с указанием <id>:

```
        {

          "id": 614,
          "text": "Добрый день. Не
    работает роутер - см. фото",
          "photo":
    "http://chat24.io/images/users/message/_19-
    45-03.jpg",
          "coordinates": null,
          "transport": "whatsapp",
          "type": "from_client",
          "read": true,
          "created": "2016-01-
    11T19:45:05 UTC",
          "client_id": 13,
          "dialog_id": 11,
          "operator_id": 1,
          "channel_id": 1
        },
      "status": "success"
    }
```


Некоторые поля:

- `coordinates` – координаты при передаче карты (местоположения).
- `type`:
  - *from_client* – от клиента.
  - *to_client* – клиенту.
  - *system* – системное сообщение (автоответ).
- `read` – пометка прочитано сообщение оператором или нет. Сообщение клиенту считается всегда прочитанным, т.к. нет информации о реальном статусе. Исключение - транспорт *Viber Business*.
- `created` – дата создания сообщения (UTC).
- `recipient_status` – информация о статусе доставки сообщения клиенту (сейчас относится только к транспорту *Viber Business*).

web_hook (POST)

При установке web_hook, указанные ниже сообщения будут передаваться на заданный URL. Web_hook используется для 3 целей:

1. Для отслеживания поступления нового сообщения от вашего клиента в систему при использовании стандартных мессенджеров (обычный *канал*).
2. Для отслеживания поступления нового сообщения от системы (чат-центра) к клиенту при использовании *внешнего канала*.
3. Для отслеживания установки и удаления тегов по клиенту.

Запрос на установку web_hook:

```
POST /v1/companies/web_hook
```

Тело запроса:

```
{
  "url": «https://yoursite.com/"
}
```

Допускается любой URL, но рекомендуется https. Если передаётся пустое значение (null), то web hook удаляется. Текущий web hook можно узнать с помощью api_info (GET).

- В случае передачи информации о входящем сообщении от клиента, ответ соответствует формату ответа на команду messages (GET), а параметр `type` равен *from_client*.
- В случае передачи информации об исходящем сообщении от системы (чат-центра) к клиенту, ответ будет содержать данные о сообщении, `id` клиента, которому предназначено сообщение, имени оператора и др. – см. messages/inbox (POST), а параметр `type` равен *from_system*.

chat_close_web_hook (POST)

Задаёт URL для web hook о закрытии чата.

Запрос:

```
POST /v1/companies/chat_close_web_hook
```

Тело запроса:

```
{
  "url": "https://yoursite.com/"
}
```

Допускается любой URL, но рекомендуется https. Если передаётся пустое значение (null), то web hook удаляется.

---

messages (POST)

Отправляет сообщение клиенту. Ссылаться на клиента можно только с помощью его id. Поэтому клиент должен содержаться в списке ваших клиентов. Если клиента нет в списке, используйте clients (POST).

Запрос:

```
POST /v1/messages
```

Тело запроса:

```
{
  "client_id": 7,
  "text": "Hi there!",
  "attachment": "http://chat24.io/images/tg_app_en.jpg",
  "type": "to_client",
  "transport": "viber",
  "channel_id": 1,
  "operator_id": 3
}
```

Параметры:

- `client_id` – id клиента – см. clients (GET).
- `text` – текст сообщения (обязательно если нет attachment или pdf).
- `attachment` – URL на фото. Поддерживаются только прямые ссылки.

- `pdf` – URL на PDF-файл. Поддерживаются только прямые ссылки.

  Отправка клиенту координат не поддерживается.

- `type:`
  - *to_client* – клиенту (по умолчанию).
  - *autoreply* – автоматический ответ.
  - *system* – системное сообщение (клиенту не отправляется).
- `transport:`
  - *whatsapp*
  - *viber*
  - *vkontakte*
  - *facebook*
  - *telegram*
  - *sms* и др.

  Если не указан, то используется транспорт последнего сообщения от клиента.

- `channel_id` – см. channels (GET). Если не указан, от используется канал последнего сообщения от клиента.
- `operator_id` – см. operators (GET). Если не указан, сообщение отправляется от текущего оператора диалога. Если указан оператор отличный от текущего, диалог будет переназначен на нового оператора. Если указан `type` – *autoreply*, то если у диалога не было оператора и он не указан в команде, оператор автоматически не назначается. В противном случае – назначается.

Типовой ответ содержит информацию по реально использованному каналу, оператору и др.:

```
{

  "data": {
    "channel_id": 1,
    "operator_id": 3,
    "transport": "whatsapp",
    "type": "to_client",
    "client_id": 7,
    "dialog_id": 10,
    "request_id": 12
  },
  "status": "success"
}
```

messages/inbox (POST)

Отправляет входящее сообщение от клиента в систему (чат-центр), используя *внешний канал*.

Запрос:

```
POST /v1/messages/inbox
```

Тело запроса:

```
{
    "channel_id": "1",
    "body": "This is message text",
    "image": <url>,
    "video": <url>,
    "location": "lat lng"
    "from_client": {
        "id": 123abc, или "phone": 79110001122,
        "nickname": "API_boy"
    }
}
```

Некоторые параметры:

- `id` –id, которым вы идентифицируете клиента (любые символы), *кроме телефона*. Если вы передаёте телефон, используйте следующий параметр.

- `phone` – телефон клиента, если вы передаёте именно его. В номере телефона допускаются только цифры.

Принимается один из двух параметров: или `id` или `phone`. Если использованы оба или ни одного, то будет возвращена ошибка.

- `nickname` – (необязательно) имя клиента.

Для работы этой команды у вас должен быть подключен транспорт с типом "external". Обратитесь к администрации.

---

messages/<id>/transfer (GET)

Назначает сообщение и соответствующий диалог на указанного оператора. При этом неважно есть уже у диалога оператор или нет.

Запрос:

```
GET /v1/messages/<id>/transfer?operator_id=<id>
```

Параметры:

- `<id>` – id сообщения. См. messages (GET).
- `operator_id` – id оператора, на которого будет назначено сообщение и последующий диалог. См. operators (GET).

---

api_info (GET)

Возвращает:

- Текущая версия API, используемая сервисом Nova.Chats.
- Ваш уровень доступа к API – см. api_modes (GET).
- Количество API-запросов по вашей компании за текущий месяц.
- Количество доступных каналов.
- Установленный web_hook (POST) и др. параметры.

Запрос:

```
GET /v1/companies/api_info
```

```
Host: api.chat24.io
Authorization: <token>
```

Параметры:

- `<token>` – здесь и далее см. описание авторизации.

Типовой ответ:

```
{
  "data": {
    "mode": "demo",
    "hook": null,
    "current_version": 1,
    "requests_this_month": 98,
    "channels": 2,
"company_name": "Test company"
  },
  "status": "success"
}
```

clients (GET)

Возвращает список ваших клиентов с информацией по каждому клиенту.

Запрос:

```
GET /v1/clients
или
GET /v1/clients/<id>
или
GET /v1/clients/<id>/transport
или
GET /v1/clients?phone=79310001122
или
GET /v1/clients/<id>/last_question
или
GET /v1/clients/<id>/dialogs
или
GET /v1/clients/?tags=1,2,5
```

Параметры:

- `<id>` – id клиента.

- Если указан ключ `transport`, будет возвращён список доступных транспортов по указанному клиенту.
- `phone` – фильтр по телефону клиента.
- Если указан ключ `last_question`, будет возвращён последний пункт меню, отправленный клиенту.
- Если указан ключ `dialogs`, будет возвращён список диалогов по клиенту.
- `tags` – фильтр по тегам клиента (id тегов через запятую). Будут возвращены клиенты, где присутствует ***любой*** из указанных тегов.

Типовой ответ без указания в запросе `id` клиента (список всех ваших клиентов):


```
{

  "data": [
…
    {
      "id": 1607,
      "comment": "This is a
comment",
      "assigned_name": "Real VIP
client",
      "phone": "19001112233",
      "name": "Thats my name!",
      "avatar":
"http://chat24.io/images/users/client/1112233
_1.jpg",
      "region_id": null,
      "country_id": 1
    },
…
  ],
  "meta": {
    "total": 3911,
    "limit": 20,
    "offset": 1600
  },
  "status": "success"
}
```


Типовой ответ при указании в запросе `id` клиента (один клиент с расширенной информацией по нему):

```
{

  "data": {
      "id": 1607,
      "comment": "This is a
comment",
      "assigned_name": "Real VIP
client",
      "phone": "19001112233",
      "name": "Thats my name!",
      "avatar":
"http://chat24.io/images/users/client/111122
33_1.jpg",
      "region_id": null,
      "country_id": 1
      "first_client_message": "2015-12-
27T18:36:41 UTC",
      "last_client_message": "2016-01-
28T20:29:25 UTC"
  },
  "status": "success"
}
```


Некоторые поля:

- `assigned_name` – имя, назначенное клиенту – см. clients (PUT).
- `comment` – комментарий к клиенту, оставленный оператором на сайте.
- `phone` – телефон клиента, если он есть.
- `name` – ник клиента (для мессенджеров)
- `tags` – список тегов клиента.

См. ниже типовой ответ при указании в запросе `id` клиента и ключа `transport`. Фактически, это те транспорты (с указанием каналов), которые указанный клиент использовал при общении с вашим чат-центром и которые можно использовать для ответа этому клиенту.

```
{

  "data": [
    {
      "channel_id": 1,
      "transports": [
        "sms",
        "whatsapp"
      ]
    },
    {
      "channel_id": 2,
      "transports": [
        "viber"
      ]
    }
  ],
  "status": "success"
}
```

Ниже приведен типовой ответ при указании в запросе `id` клиента и ключа `last_question`. Этот запрос возвращает последний пункт меню, который был отправлен данному клиенту.

```
{

  "data": {
    "id": 4322,
    "text": "This is menu item
#20",
    "image": null
  },
  "status": "success"
}
```

clients (POST)

Создаёт нового клиента чтобы появилась возможность отправить ему сообщение. Это полезно, например, при рассылке уведомлений своим клиентам, давшим согласие на получение WhatsApp/Viber-сообщений от вас, но ни разу ранее не писавшим вам.

Используйте с осторожностью, т.к. клиент, которому вы напишите первым, скорее всего не имеет вашего номера в своей адресной книге и может нажать кнопку «Это спам!», что в итоге может привести к блокировке вашего WhatsApp- или Viber-аккаунта.

Запрос:

```
POST
/v1/clients?phone=79310001122&transport=whatsapp&channel_
id=1
```

Параметры:

- `phone` – номер телефона клиента с кодом страны, без символов и пробелов.
- `transport` – имя транспорта (мессенджера), через который планируется вести дальнейший диалог с этим клиентом. Создание клиента возможно только с транспортами *WhatsApp*, *Viber* и *SMS*. Для получения списка возможных транспортов в системе используйте transports (GET).
- `channel_id` – (необязательно) id канала, в котором необходимо добавить клиента. Если пропущено, используется ваш первый. Выбранный транспорт должен поддерживаться указанным каналом. Для получения списка ваших каналов используйте channels (GET).

После успешного выполнения, команда вернёт информацию о созданном клиенте, как показано ниже. Регион и страна будут определены в соответствии с указанным номером телефона.


```
{

  "data": {
    "id": 75,
    "assigned_name": null,
    "phone": "11111111113",
    "name": null,
    "avatar": null,
    "region_id": null,
    "country_id": 1,
    "first_client_message": "2016-07-07T05:34:28
UTC",
    "last_client_message": "2016-07-07T05:34:28
UTC"
  },
  "status": "success"
}
```


Из-за технологических особенностей, после создания клиента с транспортом *WhatsApp* первое сообщение ему отправится не ранее чем через 7 минут. Используя полученный `id` клиента, можно отправить ему сообщение с помощью команды messages (POST). Если по указанному

телефону в вашей базе уже имеется клиент, то новый клиент не создаётся, но в список транспортов по данному клиенту добавляется транспорт, указанный в команде.

Возможность создавать клиента (чтобы в дальнейшем написать ему) по умолчанию выключена. В этом случае команда вернёт ошибку. Обратитесь к администрации для включения для вас этой возможности.

clients (PUT)

Назначает клиенту имя (assigned name), комментарий и 3 дополнительных поля.

Запрос:

```
PUT /v1/clients/<id>
```

Тело запроса:

```
{
  "nickname": "Peter",
  "comment": "This is comment",
  "extra_comment_1": "…",
  "extra_comment_2": "…",
  "extra_comment_3": "…"
}
```

Параметры:

- `<id>` – id клиента.
- `nickname` – имя клиента (assigned name). Если задать `null`, имя будет удалено.
- `comment` – комментарий к клиенту.
- `extra_comment_1…3` – доп. комментарий 1…3.

operators (GET)

Возвращает список ваших операторов в системе.

Запрос:

```
GET /v1/operators
```

Поддерживается фильтрация списка по полям:

- phone
- email
- online

Типовой ответ:


```
{
  "data": [
    {
      "id": 1,
      "email": "sss@gmail.com",
      "phone": null,
      "role": "admin",
      "online": false,
      "offline_type": null,
      "first_name": null,
      "last_name": null,
      "last_visit": "2016-03-11T19:45:22 UTC"
    },
    {
      "id": 2,
      "email": "111@11.com",
      "phone": "15550001122",
      "role": "supervisor",
      "online": false,
      "first_name": "John",
      "last_name": "Smith",
      "last_visit": "2016-02-25T17:25:43 UTC"
    }
  ],
  "meta": {
    "total": 2,
    "limit": 20,
    "offset": 0
  },
  "status": "success"
}
```


operators_groups (GET)

Возвращает список групп операторов в системе.

Запрос:

```
GET /v1/operators_groups
```

transfer_to_group (GET)

Переводит (назначает) диалог на группу операторов.

Запрос:

```
GET /v1/messages/<message_id>/transfer_to_group?group_id=<group_id>
```

В ответ команда вернет информацию о том, на какого конкретно оператора из указанной группы был назначен чат.

dialogs (GET)

Возвращает список ваших диалогов в системе. Как только оператор начинает общение с клиентом, сообщения начинают принадлежать диалогу.

Также команда может вернуть список диалогов, где есть неуверенные сообщения от клиента свыше заданного кол-ва секунд.

Запрос:

```
GET /v1/dialogs

или

GET /v1/dialogs/<id>

или

GET /v1/dialogs/unanswered?limit=600 – получение списка просроченных диалогов
```

Параметры:

- `<id>` – id диалога (необязательно). Если задано, будет возвращена информация только по указанному диалогу.
- `600` – время в секундах после последнего сообщения клиента, после которого диалог считается просроченным.

Поддерживается фильтрация списка по полям:

- `operator_id`
- `state`

Пример запроса по диалогам, принадлежащим оператору с `operator_id=2`:

```
GET /v1/dialogs?operator_id=2
```

Типовой ответ для запроса без указания `id` диалога:

```
{
  "data": [
    {
      "id": 1,
      "state": "closed",
      "begin": "2015-07-22T20:52:39 UTC",
      "end": "2015-07-26T20:05:41 UTC",
      "operator_id": 1
    },
    {
      "id": 2,
      "state": "open",
      "begin": "2015-07-22T20:34:24 UTC",
      "end": null,
      "operator_id": 1
    },
    {
      "id": 3,
      "state": "open",
      "begin": "2015-07-24T11:30:32 UTC",
      "end": null,
      "operator_id": 1
    },
    …
  ],
  "meta": {
    "total": 429,
    "limit": 20,
    "offset": 0
  },
  "status": "success"
}
```

Некоторые поля:

- `state` – состояние диалога. Диалог может быть открытым или закрытым. Если клиент напишет в закрытый диалог, то клиент, по умолчанию, получит автоответ, а такой диалог может быть продолжен любым оператором. Пока диалог не закрыт, его ведёт оператор, начавший его.
  Диалог может быть закрыт по следующим причинам:

- Истек период времени с последнего сообщения в диалоге. Период времени задаётся администратором в настройках сервиса.
- Диалог закрыт оператором вручную на сайте, скриптом или с помощью dialogs (PUT).

dialogs (PUT)

Изменяет состояние диалога и задаёт ведущего его оператора с отправкой соответствующего уведомления оператору. Также см. messages/<id>/transfer

Запрос:

```
PUT /v1/dialogs/<id>
```

Параметры:

- `<id>` – id диалога (обязательно). См. dialogs (GET).

Тело запроса:

```
{
  "operator_id": 1,
  "state": "open" или "closed",
}
```

tags (GET)

Возвращает список доступных тегов.

Запрос:

```
GET /v1/tags/<id>
```

Параметры:

- `<id>` – id тега (необязательно), информацию о котором следует вернуть. Если пропущено, команда вернет полный список ваших тегов.

tags (POST)

Создает новый тег.

Запрос:

```
POST /v1/tags
```

Тело запроса:

```
{
  "tag_group_id": 1,
  "tag_label": "ABC1",
  "tag_description": "My tag description",
  "tag_bg_color": "5b9ee6",
  "tag_text_color": "ffffff",
  "order_show": 5
}
```

Параметры:

- `tag_group_id` – id группы тегов, внутри которой будет создан новый тег. Каждый тег должен принадлежать какой-либо группе.
- `tag_label` – короткая этикетка (тикет) тега (8 символов макс.).
- `tag_description` – описание тега.
- `tag_bg_color` – номер цвета фона тега.
- `tag_text_color` – номер цвета текста тега.
- `order_show` – используется для сортировки тегов в интерфейсе.

---

tag_groups (POST)

Создает группу тегов.

Запрос:

```
POST /v1/tag_groups
```

Тело запроса:

```
{
  "tag_group_name": "Name of the group»
  "order_show": 5,
  "display": 0 или 1
}
```

Параметры:

- `tag_group_name` – имя группы тегов.
- `order_show` – используется для сортировки группы в интерфейсе.
- `display` – выводить (1) или нет (0) эту группу тегов в интерфейсе при ручном редактировании клиента или обращения.

---

tags/assign_to (POST)

Присваивает клиенту один или несколько тегов.

Запрос:

```
POST /v1/tags/assign_to
```

Тело запроса:

```
{
  "tag_ids": [1,2],
  "assignee_type": "client",
  "assignee_id": "1020"
}
```

Параметры:

- `tag_ids` – список id тегов, которые требуется присвоить.
- `assignee_type` – кому требуется присвоить теги: клиенту или обращению. Пока поддерживается присваивание только клиенту — `client`.
- `assignee_id` – id клиента или обращения, которому требуется присвоить указанные теги. Пока поддерживается присваивание только клиенту.

tags (DELETE)

Удаляет тег у клиента.

Запрос:

```
DELETE /v1/tags/<id>/delete_from
```

Параметры:

- `<id>` – id тега, который требуется удалить.

Тело запроса:

```
{
  "client_id": 1020
}
```

Параметры:

- `<client_id>` – id клиента, у которого требуется удалить указанный тег.

templates (GET)

Возвращает список ваших шаблонов для быстрых ответов.

Запрос:

```
GET /v1/templates/<id>
```

Параметры:

- `<id>` – id шаблона, который требуется получить (необязательно). Если пропущено, команда вернет полный список ваших шаблонов.

qr-decode (POST)

Декодирует QR-код.

Запрос:

```
POST /v1/qr-decode
```

Тело запроса:

```
{
  "image_path":
«http://storage.staging.chat24.io/clients/inbox/device_10
755/2017-10/13-client463204-10-49-32-59e09a3cd6d74.jpg"
}
```

Параметры:

- `image_path` – путь к картинке с QR-кодом.

roles (GET)

Возвращает имеющиеся роли в системе.

Запрос:

```
GET /v1/help/roles
```

Типовой ответ:

```
{
  "data": [
    "unconfirmed",
    "operator",
    "supervisor",
    "admin",
    "disabled",
    "deleted"
  ],
  "status": "success"
}
```

Некоторые возможные значения:

- `unconfirmed` – пользователь не подтвердил свой e-mail.
- `supervisor` – оператор с расширенными правами (супервайзер).
- `disabled` – заблокированный оператор.
- `deleted` – удалённый оператор.

dialog_states (GET)

Возвращает имеющиеся в системе состояния диалогов.

Запрос:

```
GET /v1/help/dialog_states
```

Типовой ответ:

```
{
  "data": [
    "open",
    "closed"
  ],
  "status": "success"
}
```

channels (GET)

Возвращает список ваших каналов с их названиями, поддерживаемыми транспортами и номерами телефона.

Запрос:

```
GET /v1/channels
```

Поддерживается фильтрация списка по полям:

- `phone`

Пример запроса с фильтром:

```
GET /v1/channels?phone=172502619555
```

Типовой ответ:

```
{
  "data": [
    {
      "id": 1,
      "name": "Демо-номер",
      "phone": "79217639603",
      "transports": [
        "whatsapp",
        "telegram",
        "viber"
      ]
    },
    {
      "id": 2,
      "name": "Viber only",
      "phone": "79006323538",
      "transports": [
        "viber"
      ]
    }
  ],
  "meta": {
    "total": 2,
    "limit": 20,
```

```
      "offset": 0
    },
    "status": "success"
}
```

regions (GET)

Возвращает список имеющихся в системе регионов. Список используется при автоматическом определении региона по телефонному номеру клиента.

Запрос:

```
GET /v1/regions
```

Типовой ответ:

```
{
  "data": [
    {
      "id": 1,
      "country": "Россия",
      "region": "Краснодар"
    },
    {
      "id": 2,
      "country": "Россия",
      "region": "Тверь"
    },
    {
      "id": 3,
      "country": "Россия",
      "region": "Челябинск"
    },
…
```

Регионы внутри страны пока определяются только для России.

countries (GET)

Возвращает список имеющихся в системе стран.

Запрос:

```
GET /v1/countries
```

Типовой ответ:

```
{
  "data": [
    {
      "id": 1,
```

```
        "name": "USA, Canada"
      },
      {
        "id": 2,
        "name": "Russia"
      },
      {
        "id": 3,
        "name": "Egypt"
      },
      {
        "id": 4,
        "name": "South Africa"
      },
…
```

transports (GET)

Возвращает список имеющихся транспортов в системе и поддерживаемых типов вложений на отправку и приём.

Запрос:

```
GET /v1/help/transports
```

Типовой ответ:

```
{
  "data": {
    "whatsapp": {
      "from_client": {
        "text": "yes",
        "image": "yes",
        "audio": "yes",
        "pdf": "no",
        "video": "no",
        "location": "yes"
      },
      "to_client": {
        "text": "yes",
        "image": "yes",
        "audio": "no",
        "pdf": "yes",
        "video": "no",
        "location": "no"
      }
    },
   …
    }
  },
  "status": "success"
}
```

messages/read (GET)

Помечает сообщение как *прочитанное* или *непрочитанное* оператором.

Запрос:

```
GET /v1/messages/<id>/read
или
GET /v1/messages/<id>/unread
```

Параметры:

- `<id>` – id сообщения (обязательно). См. messages (GET).

Команда помечает указанное сообщение как прочитанные или непрочитанное *оператором*. Имейте в виду, что при ответе на сообщение, это сообщение и весь диалог автоматически помечаются как прочитанные.

api_modes (GET)

Возвращает имеющиеся уровни доступа к API. Ваш текущий уровень доступа – см. api_info (GET).

Запрос:

```
GET /v1/help/api_modes
```

Типовой ответ:


```
{
  "data": [
    {
      "name": "disabled",
      "description": "API disabled"
    },
    {
      "name": "demo",
      "description": "1000 requests per month"
    },
    {
      "name": "normal",
      "description": "50000 requests per month"
    },
    {
      "name": "unlimited",
      "description": "unlimited requests per month"
```

```
        }
      ],
      "status": "success"
    }
```