# Povikvane Public API — Ръководство за интеграция за клиенти

**Версия 1.4 · Септември 2026 · SMS и Viber съобщения**

Публичното API на Povikvane позволява на вашето приложение да изпраща SMS и Viber съобщения програмно и да следи статуса на доставка в реално време. То спазва проста REST конвенция с JSON тяло на заявките и стандартни HTTP кодове на отговор.

---

## Съдържание



---

## 1. Общ преглед

| Свойство | Стойност |
|---|---|
| **Базов URL** | `https://app.povikvane.com` |
| **Протокол** | HTTPS (задължителен за всички API заявки) |
| **Content-Type** | `application/json` — задължителен при всички POST заявки |
| **Кодиране** | UTF-8 |
| **Удостоверяване** | API ключ (Bearer токен) + Service ID |

> **Откъде да вземете данните си за достъп** Вашият **API ключ** и **Service ID** се генерират от таблото за управление на акаунта ви в **Настройки → API**. Пазете API ключа в тайна — третирайте го като парола и никога не го включвайте в код, който се изпълнява в браузъра или от страна на клиента.

API-то предоставя две крайни точки:

| Метод | Път | Предназначение |
|---|---|---|
| `POST` | `/public-api/v1/sms` | Изпращане на SMS или Viber съобщение |
| `GET` | `/public-api/v1/sms/{id}` | Проверка на статуса на доставка на съобщение |

---

## 2. Първи стъпки

Изпратете първото си съобщение за под минута с минималния cURL пример по-долу. Заменете примерните стойности с реалните си данни за достъп:

```
curl -X POST https://app.povikvane.com/public-api/v1/sms \
  -H "Content-Type: application/json" \
  -H "Authorization: Bearer YOUR_API_KEY" \
  -d '{
    "service-id": "YOUR_SERVICE_ID",
    "message": {
      "to": "+359888123456",
      "text": "Здравейте! Вашата поръчка беше изпратена.",
      "channel": "sms"
    }
  }'
```

Успешният отговор изглежда така:

```
{
  "data": {
    "type": "sms",
    "id": "f47ac10b-58cc-4372-a567-0e02b2c3d479",
    "attributes": {
      "send-at": "2026-06-03 14:00:00",
      "status": "queued_on_smsc",
      "segments": 1,
      "credits-charged": 1,
      "status_detail": "queued",
      "created_at": "2026-06-03 14:00:00",
      "submitted_at": "2026-06-03 14:00:00",
      "delivered_at": null
    },
    "links": {
      "self": "https://app.povikvane.com/public-api/v1/sms/f47ac10b-58cc-4372-a567-
0e02b2c3d479"
    }
  }
}
```


Запазете стойността `data.id`. Тя ще ви е необходима, за да проверявате статуса на доставка (вижте Раздел 8), или може да настроите **уебхук**, за да получавате актуализации на статуса автоматично (вижте Раздел 10).

---

## 3. Удостоверяване

Всяка заявка трябва да съдържа два елемента за удостоверяване:

| Елемент | Как се подава | Описание |
|---|---|---|
| **API ключ** | HTTP хедър: `Authorization: Bearer <api-key>` | Уникален таен ключ, генериран от таблото ви. Никога не го споделяйте и не го вграждайте в код от страна на клиента. |
| **Service ID** | **POST**: в JSON тялото като `"service-id"` · **GET**: като параметър в URL `?service-id=<id>` или хедър `x-service-id: <id>` | Уникален идентификатор на вашата услуга, сдвоен с API ключа. Двата трябва да съвпадат; несъответстващи двойки се отхвърлят. |

**И двата елемента са задължителни при всяка заявка** Липсващ или невалиден API ключ връща **HTTP 401 Unauthorized**. Липсващ Service ID връща **HTTP 400 Bad Request**. API ключ, който е деактивиран от таблото, също връща **HTTP 401**.

Всеки отговор включва и хедър `X-Request-Id` — уникален идентификатор на конкретната заявка. Записвайте го при вас; посочването му към поддръжката прави проследяването на всеки проблем много по-бързо. По желание може да изпратите собствен хедър `X-Request-Id` и той ще бъде върнат в отговора.

---

## 4. Изпращане на съобщение

```
POST /public-api/v1/sms
```

Приема съобщение и го препраща към оператора веднага. Чрез API-то няма планирано или отложено изпращане.

### Минимална заявка (SMS)

```
POST /public-api/v1/sms HTTP/1.1
Host: app.povikvane.com
Content-Type: application/json
Authorization: Bearer YOUR_API_KEY

{
  "service-id": "a1b2c3d4-e5f6-7890-abcd-ef1234567890",
  "message": {
    "to": "+359888123456",
    "text": "Вашият код за потвърждение е 482910."
  }
}
```

### Viber съобщение

```
POST /public-api/v1/sms HTTP/1.1
Host: app.povikvane.com
Content-Type: application/json
Authorization: Bearer YOUR_API_KEY

{
  "service-id": "a1b2c3d4-e5f6-7890-abcd-ef1234567890",
  "message": {
    "to": "+359888123456",
    "text": "Здравей, Мария! Часът ти е потвърден за утре в 10:00.",
    "channel": "viber"
  }
}
```

**Viber** Задайте `"channel": "viber"`, за да доставите чрез Viber. Всички специфични за Viber параметри за маршрутизация (код на платформата, TTL на съобщението) се конфигурират автоматично от платформата — не е нужно да ги задавате.

## 5. Параметри на заявката

| Параметър | Тип | Задължителен | Описание |
|---|---|---|---|
| `service-id` | string | Да | Вашият Service ID от таблото. Подава се в JSON тялото. |
| `message.to` | string | Да | Телефонен номер на получателя в международен формат, включително кода на държавата. Български пример: `+359888123456`. Номерът се валидира и нормализира автоматично. Неподдържани формати се отхвърлят с HTTP 400. |
| `message.text` | string | Да | Текст на съобщението. Не може да е празен (празна или липсваща стойност връща HTTP 400). Препоръчителна максимална дължина: 1 600 знака (вижте Правила за валидация). |
| `message.channel` | string | Не | Канал за доставка. Допустими стойности: `"sms"` (по подразбиране) или `"viber"`. Всяка друга стойност се третира като `"sms"`. |
| `message.template_id` | string | Не | ID на ваш Viber шаблон в Povikvane. Използва се само с `"channel": "viber"` за изрично заявяване на шаблон. Вътрешно одобрен шаблон, който все още не е одобрен за транзакционна доставка, се изпраща като маркетингов текст. |
| `message.params` | object | Не | Стойности за променливите в `template_id`, с ключове като `firstName` и `customField1`. Задължително е, когато избраният шаблон има променливи. |

**Формат на телефонния номер** Винаги включвайте пълния международен код за избиране. За български номера използвайте `+359`, последван от 9-цифрения абонатен номер без водеща нула. Валидни примери: `+359888123456` · `+359898765432` · `+359877000111`. Водещо `00` също се приема и нормализира (`00359888123456` → `+359888123456`).

## Транзакционни Viber шаблони

Когато изпращането чрез Viber шаблони е включено за платформата, Viber съобщение, чийто текст съвпада точно с ваш Viber шаблон, одобрен за транзакционна доставка, се изпраща автоматично като транзакционно шаблонно съобщение. Не е нужна промяна по интеграцията. Съвпадението е точно: пунктуацията, интервалите и целият фиксиран текст трябва да съвпадат.

За изричен избор на шаблон добавете неговото Povikvane ID и стойностите на променливите му. `text` остава задължителен и служи като защитен текст, ако доставчикът отхвърли шаблонното изпращане.

```
{
  "service-id": "YOUR_SERVICE_ID",
  "message": {
    "to": "+359888123456",
    "text": "Здравейте Мария, часът ви е утре.",
    "channel": "viber",
    "template_id": "YOUR_POVIKVANE_TEMPLATE_UUID",
    "params": { "firstName": "Мария" }
  }
}
```

Изричният шаблон трябва да принадлежи на вашата компания, да е Viber шаблон и да получи точно определените за него ключове. Всяка стойност е до 125 знака и не може да съдържа URL. Вътрешното одобрение остава задължителната защитна проверка: вътрешно неодобрен шаблон връща HTTP 400 и никога не се изпраща.

Ако шаблонът е вътрешно одобрен, но прегледът му за транзакционна доставка е със статус `pending`, `declined` или `not_submitted`, заявката се приема и се изпраща като маркетингов Viber текст, използвайки `message.text` като съдържание. Самият шаблон не се изпраща в този случай. Всички други невалидни изрични заявки продължават да връщат HTTP 400. Ако функцията е изключена, тези полета се приемат и игнорират, а отговорът съдържа `"template": "disabled"`.

Успешните Viber отговори включват допълнителен `template` детайл: `"transactional"` (и `template_id`), `"promotional"` или `"disabled"`. Когато изричен шаблон бъде изпратен като маркетингов текст, защото не е одобрен за транзакционна доставка, отговорът включва и допълнителния атрибут `template_status`: `"pending"`, `"declined"` или `"not_submitted"`. При обикновени маркетингови съобщения и резервно изпращане след отказ от доставчика `template_status` не се връща. Краен отговор с `"transactional"` отчита числовата стойност `0.8` и в `segments`, и в `credits-charged`. Маркетинговите съобщения остават `1`. Отхвърляне на шаблон и повторно изпращане като маркетингов текст запазват първоначалната цена от 1 кредит; по-късен статус за доставка не променя отчетената цена.

## 6. Идемпотентност (безопасни повторни опити)

Мрежови проблеми могат да ви оставят в неяснота дали една заявка е била обработена. Незадължителният хедър **`Idempotency-Key`** предотвратява случайно изпращане на едно и също съобщение два пъти при повторен опит.

### Как работи

- Добавете уникален низ като хедър `Idempotency-Key` към POST заявка (UUID v4 е идеален избор).
- Ако повторите със **същия ключ и същото тяло** в рамките на **24 часа**, платформата връща кеширания отговор от първоначалния опит — без да изпраща ново съобщение и без да удържа нов кредит. Повторението отговор носи допълнителен хедър **`Idempotent-Replayed: true`**.
- Ако използвате същия ключ с **различно тяло**, заявката се отхвърля с **HTTP 422** (несъответствие на отпечатъка).
- Ако повторен опит пристигне, **докато първата заявка все още се обработва**, получавате **HTTP 409** с `Retry-After: 1` — изчакайте една секунда и опитайте отново.
- Заявки **без** хедъра не са засегнати и винаги се обработват нормално.

**Формат на ключа:** до 255 знака, само от букви, цифри и знаците `_ - : .`. Ключовете са в обхвата на вашия акаунт. Невалиден ключ връща **HTTP 400**.

### Пример с Idempotency-Key

```
POST /public-api/v1/sms HTTP/1.1
Host: app.povikvane.com
Content-Type: application/json
Authorization: Bearer YOUR_API_KEY
Idempotency-Key: 550e8400-e29b-41d4-a716-446655440000

{
  "service-id": "a1b2c3d4-e5f6-7890-abcd-ef1234567890",
  "message": {
    "to": "+359888123456",
    "text": "Вашата поръчка #12345 беше изпратена.",
    "channel": "sms"
  }
}
```

> **Добра практика** Генерирайте UUID преди заявката и го запазете заедно със записа за поръчката или операцията. Използвайте същия UUID при всеки повторен опит за тази операция и нов UUID за всяко ново, отделно съобщение.

## 7. Успешен отговор

Успешното изпращане връща **HTTP 200 OK**:


```
{
  "data": {
    "type": "sms",
    "id": "f47ac10b-58cc-4372-a567-0e02b2c3d479",
    "attributes": {
      "send-at": "2026-06-03 14:00:00",
      "status": "queued_on_smsc",
      "segments": 1,
      "credits-charged": 1,
      "status_detail": "queued",
      "created_at": "2026-06-03 14:00:00",
      "submitted_at": "2026-06-03 14:00:00",
      "delivered_at": null
    },
    "links": {
      "self": "https://app.povikvane.com/public-api/v1/sms/f47ac10b-58cc-4372-a567-
0e02b2c3d479"
    }
  }
}
```


 

| Поле | Тип | Описание |
|---|---|---|
| `data.type` | string | Винаги `"sms"`, дори когато каналът е Viber. |
| `data.id` | string (UUID) | Уникален идентификатор на съобщението. Използвайте го, за да проверявате статуса или да съпоставяте входящи уебхук събития. |
| `data.attributes.send-at` | string | Време на изпращане във формат `YYYY-MM-DD HH:MM:SS` (UTC). |
| `data.attributes.status` | string | Винаги `queued_on_smsc` непосредствено след успешно изпращане. Вижте Раздел 9 за всички възможни стойности. |
| `data.attributes.segments` | number | Брой SMS сегменти, на които е разделено съобщението (винаги `1` за Viber). Вижте Кредитна система. |
| `data.attributes["credits-charged"]` | number | Удържани кредити за това съобщение. Равно на `segments`. |
| `data.attributes.status_detail` | string | По-детайлен статус. В отговора при изпращане винаги е `"queued"`. Вижте Раздел 9. |
| `data.attributes.created_at` | string | Кога е създаден записът на съобщението, във формат `YYYY-MM-DD HH:MM:SS` (UTC). |
| `data.attributes.submitted_at` | string | Кога платформата е приела и подала съобщението, във формат `YYYY-MM-DD HH:MM:SS` (UTC). |
| `data.attributes.delivered_at` | string \| null | Винаги `null` в отговора при изпращане. Потвърждението за доставка идва по-късно през крайната точка за статус или чрез уебхук. |
| `data.links.self` | string (URL) | Пълен URL за извличане на текущия статус на съобщението чрез GET (вижте Раздел 8). |
| `warnings` | array \| липсва | Присъства когато платформата е открила име на поле, наподобяващо честа грешка (вижте по-долу). При отговор **200** съобщението е изпратено нормално; при отговор **422** масивът `warnings` се появява заедно с масива `errors`, за да улесни установяването на вероятната причина. |

**Обратна съвместимост** Полетата `status_detail`, `created_at`, `submitted_at` и `delivered_at` са допълнителни. Първоначалните полета (`status` и `send-at`) са непроменени. Интеграции, които игнорират непознати полета, продължават да работят без промени.

### Полето `warnings`

Когато тялото на заявката съдържа ime на поле, съответстващо на позната честа грешка — например `recipient` вместо `message.to`, или `body` вместо `message.text` — отговорът включва масив `warnings` на най-горно ниво.

- При отговор **200 OK** съобщението е изпратено нормално; предупреждението е само информационно.
- При отговор **422 Unprocessable Entity** масивът `warnings` се появява заедно с масива `errors`. Това се случва, когато изпратеният псевдоним е разпознат, но правилното задължително поле все още липсва — например изпратили сте `message.recipient`, но сте пропуснали `message.to`. Стойността `hint` на предупреждението показва точно кое поле трябва да коригирате.

```
{
  "data": { "..." : "..." },
  "warnings": [
    {
      "field": "recipient",
      "hint": "Did you mean \"message.to\"? The field \"recipient\" is not a recognised top-level parameter and was ignored."
    }
  ]
}
```

| Поле в предупреждението | Тип | Описание |
|---|---|---|
| `field` | string | Името на съмнителното поле така, както се е появило в заявката ви (напр. `"recipient"` или `"message.body"`). |
| `hint` | string | Съвет на човешки език, назоваващ правилното поле. |

**Таблица с псевдоними** — полета, предизвикващи предупреждение:

| Изпратено поле | Правилно поле | Бележки |
|---|---|---|
| `to` (на най-горно ниво) | `message.to` | Получателят трябва да е вътре в обекта `message`. |
| `recipient` (на най-горно ниво) | `message.to` | Получателят трябва да е вътре в обекта `message`. |
| `phone` (на най-горно ниво) | `message.to` | Получателят трябва да е вътре в обекта `message`. |
| `body` (на най-горно ниво) | `message.text` | Текстът на съобщението трябва да е вътре в обекта `message`. |
| `content` (на най-горно ниво) | `message.text` | Текстът на съобщението трябва да е вътре в обекта `message`. |
| `message.recipient` | `message.to` | Грешно име на поле вътре в `message`. |
| `message.phone` | `message.to` | Грешно име на поле вътре в `message`. |
| `message.body` | `message.text` | Грешно име на поле вътре в `message`. |
| `message.content` | `message.text` | Грешно име на поле вътре в `message`. |

> **Проверявайте за `warnings` в интеграцията си** Ако видите масив `warnings` в отговора, разгледайте стойностите на `hint` и коригирайте тялото на заявката. Изпратеното от вас поле е игнорирано мълчаливо — предвидената стойност (телефонен номер или текст на съобщението) **не** е използвана.

---

## 8. Проверка на статуса на съобщение

```
GET /public-api/v1/sms/{id}
```

Извлича текущия статус на доставка на вече изпратено съобщение. Заменете `{id}` с UUID-а, върнат при изпращането на съобщението.

### Вариант А — Service ID като параметър в URL (препоръчително)

```
GET /public-api/v1/sms/f47ac10b-58cc-4372-a567-0e02b2c3d479?service-id=a1b2c3d4-e5f6-7890-abcd-ef1234567890 HTTP/1.1
Host: app.povikvane.com
Authorization: Bearer YOUR_API_KEY
```

### Вариант Б — Service ID като хедър

```
GET /public-api/v1/sms/f47ac10b-58cc-4372-a567-0e02b2c3d479 HTTP/1.1
Host: app.povikvane.com
Authorization: Bearer YOUR_API_KEY
x-service-id: a1b2c3d4-e5f6-7890-abcd-ef1234567890
```

### Примерен отговор (доставено съобщение)

```
{
  "data": {
    "type": "sms",
    "id": "f47ac10b-58cc-4372-a567-0e02b2c3d479",
    "attributes": {
      "send-at": "2026-06-03 14:00:00",
      "status": "delivered_to_handset",
      "status_detail": "delivered"
    },
    "links": {
      "self": "https://app.povikvane.com/public-api/v1/sms/f47ac10b-58cc-4372-a567-
0e02b2c3d479"
    }
  }
}
```

**Бележка за полетата в отговора** Крайната точка за статус връща `status` и `send-at`, плюс `status_detail`, когато има съответствие (`queued` или `delivered`). Полетата `created_at`, `submitted_at` и `delivered_at` се връщат в отговора при **изпращане**, а не в отговора за статус.

**Използвайте уебхукове вместо периодична проверка, където е възможно** Уебхуковете (вижте Раздел 10) изпращат крайния статус на доставка към вашата крайна точка в момента, в който той пристигне от оператора. Те са по-ефективни от проверка в цикъл и намаляват излишните API заявки.

---

 

# 9. Статуси и жизнен цикъл на съобщенията

## Статуси, връщани от API-то (`status`)

| Статус | Значение |
|---|---|
| `queued_on_smsc` | Съобщението е прието и препратено към оператора. Потвърждението за доставка предстои. Това е винаги статусът непосредствено след успешно изпращане. |
| `delivered_to_handset` | Операторът потвърди, че съобщението е доставено до устройството на получателя (SMS телефон или Viber приложение). |
| `not_delivered_to_handset` | Съобщението не беше доставено. Възможни причини: операторът го отхвърли, номерът е невалиден или изключен, Viber не е инсталиран или TTL на съобщението е изтекъл. |

## Жизнен цикъл на статуса

```
queued_on_smsc  →  delivered_to_handset       (успешен път)
queued_on_smsc  →  not_delivered_to_handset   (път при неуспех)
```

## Полето `status_detail`

`status_detail` е допълнителен, по-детайлен етикет. API-то връща само следните две стойности и пропуска полето изцяло, когато няма приложимо съответствие:

| `status_detail` | Появява се, когато `status` е | Значение |
|---|---|---|
| `queued` | `queued_on_smsc` | Прието и препратено към оператора; очаква се разписка за доставка (DLR). |
| `delivered` | `delivered_to_handset` | Получена DLR: съобщението е достигнало устройството. |
| *(пропуснато)* | `not_delivered_to_handset` | При недоставяне не се връща `status_detail`. Разчитайте на `status` (и полето `error` в уебхука), за да установите неуспехите. |

## Съответствие между имената при уебхук и GET

Крайната точка GET използва описателни имена за етапа от процеса, а уебхуковете използват по-кратки имена в стил „събитие“ за двата крайни резултата. Съпоставете ги както следва:

| Резултат | `status` при GET | `data.status` при уебхук |
|---|---|---|
| Доставено | `delivered_to_handset` | `delivered` |
| Недоставено | `not_delivered_to_handset` | `failed` |
| Все още в движение | `queued_on_smsc` | *(уебхук още не е изпратен)* |

---

## 10. Уебхукове (push известия)

Конфигурирайте **URL за уебхук** в таблото си на **Настройки → API → Webhook URL**, за да получавате автоматични HTTP POST известия, когато съобщение достигне краен статус на доставка.

### Кога се изпраща уебхук?

Известията се задействат само когато съобщението достигне едно от двата крайни състояния:

- `delivered` — съобщението е доставено успешно до получателя.
- `failed` — съобщението не можа да бъде доставено.

### Тяло на уебхука

```
{
  "event": "message.status_updated",
  "data": {
    "id": "f47ac10b-58cc-4372-a567-0e02b2c3d479",
    "status": "delivered",
    "channel": "sms",
    "recipient": "+359888123456",
    "timestamp": "2026-06-03T14:01:23.000Z",
    "error": null
  }
}
```

| Поле | Тип | Описание |
|---|---|---|
| `event` | string | Винаги `"message.status_updated"`. |
| `data.id` | string (UUID) | Идентификатор на съобщението. Съвпада с `data.id` от отговора при изпращане. |
| `data.status` | string | `"delivered"` или `"failed"`. |
| `data.channel` | string | `"sms"` или `"viber"`. |
| `data.recipient` | string | Телефонният номер на получателя в международен формат. |
| `data.timestamp` | string (ISO 8601) | Кога е настъпило събитието за статус. |
| `data.error` | string \| null | Четима причина за неуспеха, когато `status` е `"failed"`; `null` при успех. |

## Изисквания към вашата крайна точка за уебхук

- Трябва да е достъпна по `http://` или `https://` от публичния интернет (не може да сочи към `localhost`).
- Трябва да отговори с HTTP 2xx статус в рамките на **5 секунди**. Ако крайната ви точка надхвърли времето или върне грешка, известието **не** се повтаря автоматично.
- Конфигурира се през таблото — не през самото API.

**Отговаряйте веднага, обработвайте асинхронно** Върнете HTTP 200 веднага щом получите заявката, след което обработете данните във фонова задача. Така времето ви за отговор остава далеч под 5 секунди и се избягват пропуснати известия.

**Направете обработчика си идемпотентен** При редки условия едно и също известие може да пристигне повече от веднъж. Проверявайте дали вече сте обработили `data.id`, преди да действате по него. Уебхуковете не са криптографски подписани, затова също потвърждавайте, че `data.id` отговаря на съобщение, което действително сте изпратили, преди да се доверите на събитието.

**Периодичната проверка е достоверният резервен вариант** Уебхуковете се изпращат еднократно без автоматично повторение, така че известие може да бъде пропуснато, ако крайната ви точка е временно недостъпна. За всяко съобщение, чийто краен статус е важен, третирайте **крайната точка GET за статус** (вижте Раздел 8) като източник на истина и периодично сверявайте.

# 11. Лимити на заявките

API-то налага два независими лимита върху **крайната точка POST за изпращане**, с плъзгащ прозорец от 60 секунди:

| Ниво на лимит | Обхват | Лимит | Прозорец |
|---|---|---|---|
| **На фирма** | Вашият фирмен акаунт | **400 съобщения** | 60 секунди |
| **За цялата платформа** | Всички клиенти заедно | **4 000 съобщения** | 60 секунди |

При надхвърляне на лимит API-то връща **HTTP 429 Too Many Requests** с хедър `Retry-After`, указващ колко секунди да изчакате.

### Примерен отговор 429

```
HTTP/1.1 429 Too Many Requests
Retry-After: 15
Content-Type: application/json

{
  "errors": [
    {
      "status": "429",
      "title": "Rate limit exceeded",
      "detail": "You have exceeded the rate limit of 400 messages per minute. Please wait 15 seconds before sending more messages."
    }
  ]
}
```

## Лимит за GET (проверка на статус)

За крайната точка GET за статус поддържайте проверките в рамките на ориентир от **600 заявки на 60 секунди на Service ID**. Твърдото налагане на лимита по-горе се отнася за крайната точка POST за изпращане; третирайте стойността за GET като препоръчителен таван за разумна периодична проверка. Все пак препоръчваме уебхуковете като основен механизъм за известяване за доставка, така че рядко да се налага да проверявате изобщо.

## Добри практики

- **Спазвайте** `Retry-After` — изчакайте поне толкова секунди, преди да опитате отново. Незабавните повторни опити се отхвърлят и забавят нулирането на прозореца.

- **Използвайте `Idempotency-Key` при повторни опити** (вижте Раздел 6), за да не може повторен опит след 429 да удържи кредит два пъти.
- **Прилагайте експоненциално нарастващо изчакване** след повтарящи се неуспехи.
- **Разпределяйте масовите изпращания** във времето, вместо да ги пускате едновременно.
- **Използвайте уебхукове вместо периодична проверка**, за да не изчерпвате квотата си за GET.

---

## 12. Кредитна система

**SMS съобщенията се таксуват на сегмент.** SMS, което се побира в един сегмент, струва 1 кредит; по-дългите съобщения струват по 1 кредит на сегмент. Размерът на сегмента зависи от използваните знаци:

| Кодиране | Кога се прилага | Единично съобщение | Всеки сегмент от многосегментно съобщение |
|---|---|---|---|
| GSM-7 | Латински букви, цифри, обичайна пунктуация | 160 знака | 153 знака |
| UCS-2 | Всеки друг знак (напр. кирилица, емоджи) | 70 знака | 67 знака |

Забележка: разширените GSM-7 знаци `| ^ € { } [ ] ~ \` се броят по **2** знака. Един-единствен знак извън GSM-7 (напр. една буква на кирилица) превключва цялото съобщение към UCS-2.

**Viber съобщенията винаги струват 1 кредит**, независимо от дължината.

Успешният отговор при изпращане съдържа цената в `data.attributes["credits-charged"]` (и еквивалентното `data.attributes.segments`), за да следите разхода за всяко съобщение.

Кредитите се проверяват и удържат атомарно в момента на заявката — преди съобщението да бъде препратено към оператора.

| Сценарий | Ефект върху кредитите | Бележки |
|---|---|---|
| Съобщението е прието и изпратено | **−N кредита** (N = брой сегменти; 1 за Viber) | Удържат се веднага при приемане на заявката. |
| Недостатъчно кредити (HTTP 402) | 0 кредита | Заявката се отхвърля, преди да бъде удържан кредит — проверката използва пълната сегментна цена на съобщението. Заредете баланса си от таблото. |
| Операторът изрично отхвърля съобщението (HTTP 502) | **+N кредита (върнати)** | Точно удържаната сума се връща автоматично, когато операторът отговори с ясен отказ по време на изпращането. |
| Връзката с оператора изтече (HTTP 504) | −N кредита (не се връщат) | Резултатът от доставката е неизвестен; кредитите не се връщат автоматично, а отговорът 504 не съдържа идентификатор на съобщение. Свържете се с поддръжката, ако това се повтаря. |

**При HTTP 504 (Gateway Timeout)** Връзката с оператора е прекъсната, преди да бъде получен категоричен отговор. Съобщението може да е било препратено или не. Ако често виждате грешки 504, свържете се с поддръжката за проучване.

**Следете баланса си от кредити** Оставащите ви кредити се показват в горната лента на таблото и в **Настройки → Плащане**. Зареждайте предварително, за да избегнете прекъсвания.

---

## 13. Справочник на грешките

Всички отговори с грешка споделят една и съща JSON структура:

```
{
  "errors": [
    {
      "status": "400",
      "title": "Bad Request",
      "detail": "Missing message.to or message.text"
    }
  ]
}
```

Някои грешки включват и машинно четимо поле `code` (например грешките за лимит на заявките).

| HTTP код | Заглавие | Кога възниква |
|---|---|---|
| **400** | Bad Request | Липсващи или невалидни параметри: `service-id` липсва в тялото, липсва `message.to` / `message.text`, невалиден телефонен номер, неправилен JSON или невалиден формат на `Idempotency-Key`. |
| **401** | Unauthorized | Липсващ, грешен или деактивиран API ключ; или Service ID не съвпада с API ключа. |
| **402** | Payment Required | Недостатъчно кредити за пълната сегментна цена на съобщението. Не е удържан кредит. Заредете баланса си. |
| **404** | Not Found | Няма съобщение с подадения ID при `GET` (връща се и когато съобщението принадлежи на друга фирма, от съображения за сигурност). |
| **409** | Conflict | Заявка със същия `Idempotency-Key` все още се обработва. Спазете `Retry-After: 1` и опитайте отново. |
| **422** | Unprocessable Entity | `Idempotency-Key` вече е бил използван с различно тяло на заявката (използвайте нов ключ за ново съобщение) или задължително поле не е преминало стриктна валидация. Когато е открит разпознат псевдоним на поле, в тялото на отговора 422 се включва и масив `warnings` заедно с `errors`, за да помогне при идентифицирането на правилното поле. |
| **429** | Too Many Requests | Надхвърлен лимит на заявките. Прочетете `Retry-After`. Вижте Раздел 11. |
| **500** | Internal Server Error | Неочаквана грешка в платформата. Изчакайте кратко и опитайте отново; ако продължава, свържете се с поддръжката. |
| **502** | Bad Gateway | Операторът изрично отхвърли съобщението. Кредитите ви (пълната удържана сума) се връщат. Проверете номера на получателя или опитайте друг канал. |
| **504** | Gateway Timeout | Връзката с оператора изтече. Резултатът от доставката е неизвестен; кредитите не се връщат и не се връща идентификатор на съобщение. |

# 14. Правила за валидация

| Поле | Правило |
|---|---|
| `service-id` | Задължително. Трябва да съвпада с активен API ключ в платформата. Чувствително към главни/малки букви. |
| `message.to` | Задължително. Трябва да е валиден телефонен номер в международен формат (валидиран чрез `libphonenumber-js`). Нормализира се автоматично — напр. `00359888123456` става `+359888123456`. Невалидни номера връщат HTTP 400. |
| `message.text` | Задължително. Не може да е празно (празно или липсващо връща HTTP 400). Вижте препоръчителните лимити за дължина по-долу. |
| `message.channel` | Незадължително. `"sms"` (по подразбиране) или `"viber"`. Всяка друга стойност се третира като `"sms"`. |
| `Idempotency-Key` (хедър) | Незадължително. До 255 знака от `A–Z a–z 0–9 _ - : .`. В обхвата на вашия акаунт. Прозорец за дедупликация: 24 часа. |

**Допълнителните полета се игнорират мълчаливо** API-то приема и игнорира всякакви допълнителни полета от най-горно ниво или вътре в `message.*`, които не са изброени по-горе. Интеграции, включващи допълнителни метаданни (напр. `status`, `tracking_id` или персонализирани свойства) в тялото на заявката, ще продължат да работят без промени. Само документираните полета по-горе се четат и обработват.

## Препоръчителна дължина на текста на съобщението

За надеждна доставка поддържайте текста на съобщението в рамките на тези лимити:

| Канал | Препоръчителен максимум | Бележки |
|---|---|---|
| SMS | 1 600 знака | Дългите съобщения се разделят на сегменти (160 знака GSM-7 / 70 UCS-2 за единично съобщение; 153 / 67 на сегмент при съединяване). **Всеки сегмент струва 1 кредит** — вижте Кредитна система. |
| Viber | 1 000 знака | Поддържа пълен Unicode, включително емоджи. Струва 1 кредит. |

# 15. Примери с код

## cURL — Изпращане на съобщение

```
curl -X POST https://app.povikvane.com/public-api/v1/sms \
  -H "Content-Type: application/json" \
  -H "Authorization: Bearer YOUR_API_KEY" \
  -H "Idempotency-Key: 550e8400-e29b-41d4-a716-446655440000" \
  -d '{
    "service-id": "YOUR_SERVICE_ID",
    "message": {
      "to": "+359888123456",
      "text": "Здравейте! Вашата поръчка #12345 беше изпратена.",
      "channel": "sms"
    }
  }'
```

## cURL — Проверка на статуса

```
curl "https://app.povikvane.com/public-api/v1/sms/MESSAGE_ID?service-id=YOUR_SERVICE_ID" \
  -H "Authorization: Bearer YOUR_API_KEY"
```

## PHP — Изпращане на съобщение

```php
<?php
function sendPovikvaneMessage(
    string $apiKey,
    string $serviceId,
    string $to,
    string $text,
    string $channel = 'sms',
    ?string $idempotencyKey = null
): array {
    $url = 'https://app.povikvane.com/public-api/v1/sms';
    $payload = json_encode([
        'service-id' => $serviceId,
        'message'    => [
            'to'      => $to,
            'text'    => $text,
            'channel' => $channel,
        ],
    ]);
    $headers = [
        'Content-Type: application/json',
        'Authorization: Bearer ' . $apiKey,
    ];
    if ($idempotencyKey !== null) {
        $headers[] = 'Idempotency-Key: ' . $idempotencyKey;
    }

    $ch = curl_init($url);
    curl_setopt_array($ch, [
        CURLOPT_POST           => true,
        CURLOPT_RETURNTRANSFER => true,
        CURLOPT_HTTPHEADER     => $headers,
        CURLOPT_POSTFIELDS     => $payload,
        CURLOPT_TIMEOUT        => 30,
    ]);
    $body     = curl_exec($ch);
    $httpCode = curl_getinfo($ch, CURLINFO_HTTP_CODE);
    curl_close($ch);

    $data = json_decode($body, true);
    if ($httpCode === 200) {
        return [
            'success'    => true,
```

```
            'message_id' => $data['data']['id'] ?? null,
            'status'     => $data['data']['attributes']['status'] ?? null,
        ];
    }
    return [
        'success' => false,
        'code'    => $httpCode,
        'error'   => $data['errors'][0]['detail'] ?? 'Unknown error',
    ];
}
```

## Node.js — Изпращане на съобщение

```
async function sendMessage({ apiKey, serviceId, to, text, channel = "sms", idempotencyKey })
{
  const headers = {
    "Content-Type": "application/json",
    Authorization: `Bearer ${apiKey}`,
  };
  if (idempotencyKey) headers["Idempotency-Key"] = idempotencyKey;

  const res = await fetch("https://app.povikvane.com/public-api/v1/sms", {
    method: "POST",
    headers,
    body: JSON.stringify({
      "service-id": serviceId,
      message: { to, text, channel },
    }),
  });

  const data = await res.json();
  if (res.ok) {
    return { ok: true, id: data.data.id, status: data.data.attributes.status };
  }
  if (res.status === 429) {
    return { ok: false, retryAfter: Number(res.headers.get("retry-after")) || 60 };
  }
  return { ok: false, code: res.status, error: data.errors?.[0]?.detail };
}
```

### Python — Изпращане на съобщение

```
import requests

def send_message(api_key, service_id, to, text, channel="sms", idempotency_key=None):
    headers = {
        "Content-Type": "application/json",
        "Authorization": f"Bearer {api_key}",
    }
    if idempotency_key:
        headers["Idempotency-Key"] = idempotency_key

    res = requests.post(
        "https://app.povikvane.com/public-api/v1/sms",
        headers=headers,
        json={
            "service-id": service_id,
            "message": {"to": to, "text": text, "channel": channel},
        },
        timeout=30,
    )
    data = res.json()
    if res.status_code == 200:
        return {"ok": True, "id": data["data"]["id"],
                "status": data["data"]["attributes"]["status"]}
    if res.status_code == 429:
        return {"ok": False, "retry_after": int(res.headers.get("Retry-After", 60))}
    return {"ok": False, "code": res.status_code,
            "error": data.get("errors", [{}])[0].get("detail")}
```

---

## 16. Поддръжка

Ако имате въпроси относно интеграцията с публичното API на Povikvane, нужда от изясняване на статус на доставка или срещате постоянни грешки `500` / `504`, свържете се с нас на **contact@povikvane.com**. Когато съобщавате за проблем, посочете `data.id` на съобщението и хедъра `X-Request-Id` от отговора — те ни позволяват да проследим заявката ви бързо.

---

*Povikvane Public API — Ръководство за интеграция за клиенти · v1.3*