



GUIDES · API V1

Idempotency

Networks drop responses. An Idempotency-Key lets you retry any POST without sending — or paying for — the same SMS twice.

Updated 2026-10-10 · generated 2026-10-10 · smsray.com/docs/idempotency

How it works

Add an Idempotency-Key header to any POST — /sms/send, /sms/otp/send or /sms/otp/verify. The key is 1–200 printable ASCII characters and is remembered for **24 hours**, scoped to your API client.

We fingerprint the request (method, path and body). When the first request finishes successfully, its response is stored. Any retry with the same key and the same body gets that stored response back — nothing is sent or billed again.

Possible outcomes

SITUATION	RESULT
First request with this key	Processed normally.
Same key, same body, first request finished	The original status and body, with header Idempotent-Replayed: true.
Same key, same body, first request still running	409 idempotency_conflict with Retry-After: 1.
Same key, different body	409 idempotency_conflict.
First request returned 4xx or 5xx	Not stored. Fix the problem and retry with the same key.

Example

cURL

```
# Safe to run twice – the second call replays the first response
curl https://api.smsray.com/api/sms/v1/sms/send \
  -H "x-api-key: $SMSRAY_API_KEY" \
  -H "content-type: application/json" \
  -H "Idempotency-Key: order-1042-shipped" \
  -d '{"to": "+9779801234567", "text": "Your order #1042 has shipped."}' -i
```

Node.js

```
async function sendOnce(key, body, tries = 4) {
  for (let i = 0; i < tries; i++) {
    try {
      const res = await fetch("https://api.smsray.com/api/sms/v1/sms/send", {
        method: "POST",
        headers: {
          "x-api-key": process.env.SMSRAY_API_KEY,
          "content-type": "application/json",
          "Idempotency-Key": key, // same key on every retry
        },
        body: JSON.stringify(body),
      });
      if (res.status < 500 && res.status !== 409 && res.status !== 429) return res.json();
      const wait = Number(res.headers.get("retry-after") ?? 2 ** i);
      await new Promise((r) => setTimeout(r, wait * 1000));
    } catch {
      await new Promise((r) => setTimeout(r, 2 ** i * 1000)); // network error: retry
    }
  }
  throw new Error("SMS send failed after retries");
}
```

Python

```
import os, time, requests

def send_once(key, body, tries=4):
    for i in range(tries):
        try:
            r = requests.post("https://api.smsray.com/api/sms/v1/sms/send", json=body, timeout=10,
» headers={
                "x-api-key": os.environ["SMSRAY_API_KEY"],
                "Idempotency-Key": key, # same key on every retry
            })
            if r.status_code < 500 and r.status_code not in (409, 429):
                return r.json()
            time.sleep(float(r.headers.get("Retry-After", 2 ** i)))
        except requests.RequestException:
            time.sleep(2 ** i)
    raise RuntimeError("SMS send failed after retries")
```

PHP

```
function send_once(string $key, array $body, int $tries = 4): array {
    for ($i = 0; $i < $tries; $i++) {
        $retryAfter = null;
        $ch = curl_init("https://api.smsray.com/api/sms/v1/sms/send");
        curl_setopt_array($ch, [
            CURLOPT_POST => true,
            CURLOPT_RETURNTRANSFER => true,
            CURLOPT_TIMEOUT => 10,
            CURLOPT_HTTPHEADER => [
                "x-api-key: " . getenv("SMSRAY_API_KEY"),
                "content-type: application/json",
                "Idempotency-Key: " . $key, // same key on every retry
            ],
            CURLOPT_HEADERFUNCTION => function ($ch, $h) use (&$retryAfter) {
```

```
        if (strpos($h, "retry-after:") === 0) $retryAfter = (int) trim(substr($h, 12));
        return strlen($h);
    },
    CURLOPT_POSTFIELDS => json_encode($body),
]);
$out = curl_exec($ch);
$status = curl_getinfo($ch, CURLINFO_HTTP_CODE);
if ($out !== false && $status < 500 && $status !== 409 && $status !== 429) return
» json_decode($out, true);
    sleep($retryAfter ?? 2 ** $i);
}
throw new RuntimeException("SMS send failed after retries");
}
```

Go

```
func SendOnce(key string, body []byte) (*http.Response, error) {
    for i := 0; i < 4; i++ {
        req, _ := http.NewRequest("POST", "https://api.smsray.com/api/sms/v1/sms/send",
» bytes.NewReader(body))
        req.Header.Set("x-api-key", os.Getenv("SMSRAY_API_KEY"))
        req.Header.Set("content-type", "application/json")
        req.Header.Set("Idempotency-Key", key) // same key on every retry

        wait := time.Duration(1<<i) * time.Second
        res, err := http.DefaultClient.Do(req)
        if err == nil {
            if res.StatusCode < 500 && res.StatusCode != 409 && res.StatusCode != 429 {
                return res, nil
            }
            if s, e := strconv.Atoi(res.Header.Get("Retry-After")); e == nil {
                wait = time.Duration(s) * time.Second
            }
            res.Body.Close()
        }
        time.Sleep(wait)
    }
    return nil, errors.New("SMS send failed after retries")
}
```

Choosing keys

- Derive the key from the business event, not from the attempt: order-1042-shipped, invoice-88-reminder-1.
- A random UUID also works if you generate it once and reuse it for every retry of the same message.
- Never reuse a key for a different message — you will get 409 or, worse, the old message's response.
- Keys expire after 24 hours, so don't rely on them to de-duplicate across days.

The web page is always the current version: smsray.com/docs/idempotency

Full API reference PDF: smsray.com/docs/pdf · Your keys and live examples: app.smsray.com/developers