interweb

Generated on 13/11/2026 17:57

KNOWLEDGE BASE · PUBLIC DOCUMENT

A reusable send-SMS function in PHP, Python or Node.js, with timeouts and error handling

WhatsApp & APIs

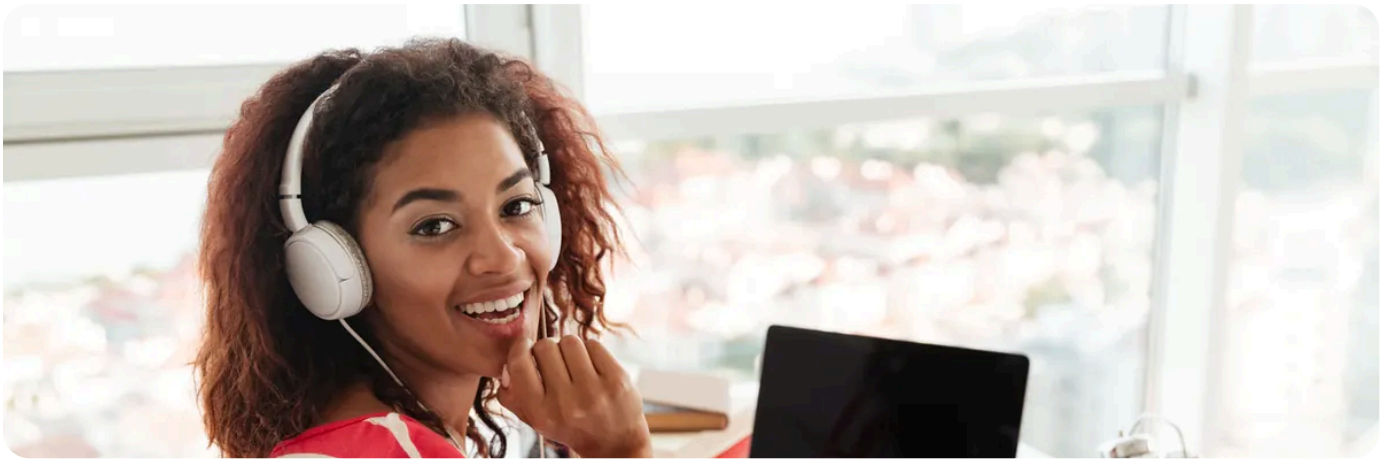
CODE	VERSION	COUNTRY · LANGUAGE
RLX-AO-KB-2602	v2026.11.13	Angola · English

Interweb Angola
RLX Angola, Lda. · Group RLX Holdings
Tax ID (NIF) 500 260 5724 · Commercial registration no. 26464-25/250724

https://www.meu.interweb.ao/knowledgebase/2602

A reusable send-SMS function in PHP, Python or Node.js, with timeouts and error handling

WhatsApp & APIs Created on **November 13, 2026** 3 min read



The short answer: write **one function** that sends the SMS and use it everywhere in your system. It takes the number, the text and an event key, reads the API key from an environment variable, has a **timeout**, always sends an `Idempotency-Key` and returns the HTTP code and the answer. The retry decision lives in one place: retry with a pause on a `429` or `5xx`, never on a `422`.

In PHP

```
function send_sms($number, $text, $event_key) {
$ch = curl_init('https://api.interweb.ao/v1/sms/messages');
curl_setopt_array($ch, [
CURLOPT_RETURNTRANSFER => true,
CURLOPT_POST => true,
CURLOPT_TIMEOUT => 30,
CURLOPT_HTTPHEADER => [
'Authorization: Bearer ' . getenv('SMS_API_KEY'),
'Content-Type: application/json',
'Idempotency-Key: ' . $event_key,
],
CURLOPT_POSTFIELDS => json_encode(['number' => $number, 'text' => $text]),
]);
$body = curl_exec($ch);
$http = curl_getinfo($ch, CURLINFO_HTTP_CODE);
curl_close($ch);
return [$http, json_decode((string) $body, true)];
}
```

In Python

```
import os, requests

def send_sms(number, text, event_key):
r = requests.post(
"https://api.interweb.ao/v1/sms/messages",
headers={"Authorization": "Bearer " + os.environ["SMS_API_KEY"],
"Idempotency-Key": event_key},
json={"number": number, "text": text},
timeout=30,
)
return r.status_code, r.json()
```

In Node.js

```
async function sendSms(number, text, eventKey) {
const r = await fetch("https://api.interweb.ao/v1/sms/messages", {
method: "POST",
headers: { "Authorization": "Bearer " + process.env.SMS_API_KEY,
"Content-Type": "application/json",
"Idempotency-Key": eventKey },
body: JSON.stringify({ number: number, text: text }),
signal: AbortSignal.timeout(30000),
});
return [r.status, await r.json()];
}
```

The `event_key` is what makes retries safe: something like `payment-3381` or `order-8812-shipped`. With the same value and the same body, the API returns the stored answer instead of sending again.

What to do with the answer

Code	What to do
201	Store the `id` and `status` with the event. Done.
422	Do not retry. Read `details.reason` (for example `invalid_number` or `insufficient_credits`) and fix it or alert someone on your team.
401 and 403	Do not retry. The key is missing, wrong or expired, lacks the `sms:write` permission, the IP is not on its list, or the account has no active SMS service (`sms_reseller_required`).

Code	What to do
409	The same event key with a different text, or the first request still running. Wait and retry unchanged, or use a new key if it really is another send.
429, 500, 502, 503	Retry later with the same event key. On a 429, wait at least as long as the `Retry-After` header says.
No answer (timeout)	Retry with the same event key: if the first request got through, the API does not send a second SMS.

Do not send the SMS inside the customer's request.
If your payment page waits for the SMS API, any delay in the SMS delays the payment. Put the send in a queue or a background job, and let the function retry there.

Keep the `correlation_id` of any error
(it is in the answer and in the `X-Correlation-ID` header). If you need help, support will ask for it, with the time and the operation. Your API key, on the other hand, is never sent to anyone.

RECOMMENDED
Ready to connect your system? Create a Sandbox key and start with Transactional SMS.
See Transactional SMS →

SEE ALSO
- Sending SMS through the API: key, permissions, Sandbox and your first request →
- SMS API errors and message statuses: what each one means and what to do →
- Timeouts, retries and idempotency: how not to charge a customer twice →
- Test cases for transactional SMS in the Sandbox: wrong numbers, long texts, no credits, repeats →
- Tracking each transactional SMS: the message id, masked numbers, statuses and charged →
- An API key or token has leaked: what to do, in the right order →

interweb

Interweb Angola

RLX Angola, Lda. · Group RLX Holdings

TAX ID (NIF)	500 260 5724
COMMERCIAL REGISTRATION NO.	26464-25/250724
REGISTERED OFFICE	Rua Direita do Patriota, Bairro Dangereux Belas, Luanda, Angola
CONTACT	info@interweb.ao · +244 942 011 191 · www.interweb.ao
OPENING HOURS	Monday to Friday - 08 AM to 04 PM Saturday & Sunday - Closed

Online version of this document

RLX-AO-KB-2602 · v2026.11.13 · Generated on 13/11/2026 17:57