

KNOWLEDGE BASE · PUBLIC DOCUMENT

Timeouts, retries and idempotency: how not to charge a customer twice

PHP & Development

CODE

RLX-MZ-KB-2168

VERSION

v2026.10.16

COUNTRY · LANGUAGE

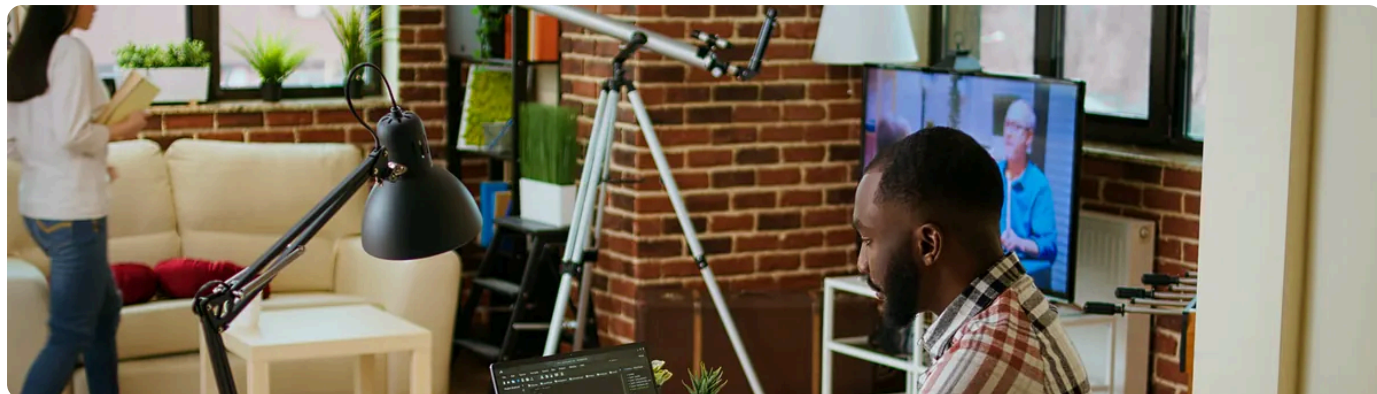
Mozambique · English

Interweb Mozambique

RLX Holdings, S.A. · Group RLX Holdings

Tax ID (NUIT) 401 191 607 · Legal entity no. (NUEL) 105 045 974 · Economic activity code (CAE) 62022

<https://www.meu.interweb.co.mz/knowledgebase/2168>



The short answer: when a request to an API times out, you do not know whether it got through. It may have failed before leaving, or been carried out on the other side with the reply lost on the way back. Retrying blindly is how a customer gets charged twice or the same SMS goes out twice. The defence has a name, **idempotency**: making a repeated request have the same effect as doing it once. This is for developers connecting a site to payments, SMS or any API that creates things.

When to retry, and when not to

What happened	Retry?	How
The name does not resolve or the connection was refused (cURL error 6 or 7)	Yes	The request never left. See cURL errors 28, 7 and 6 .
Timed out (error 28) or the connection dropped halfway	Only with care	You do not know whether it was done. Check the status first, or retry with the same idempotency key.
429 Too Many Requests	Yes, later	Wait for whatever the Retry-After header says. See error 429 .
500, 502, 503 or 504	Usually yes	With longer and longer waits and, if the request creates or charges something, with an idempotency key.
400, 401, 403, 404 or 422	No	The request is wrong or not allowed. Retrying gives the same error: fix it first.
409 Conflict	It depends	In many APIs it means “already exists” or “already in progress”. Check before doing anything else.
200 or 201	No	It is done. Keep the response.

Making a charge safe to retry

1

Give every operation an identifier of your own, created before the first attempt and stored with the order: the order reference, for example, or a unique identifier generated on the spot.

2

If the API accepts an idempotency key, send it, and send the same one on every retry. When the API recognises the key, it returns the result of the first attempt instead of charging again. The field's name varies between services (an `Idempotency-Key` header in several APIs, an "external reference" in others):

```
curl -i -X POST
https://api.example.com/v1/charges \
-H "Authorization: Bearer YOUR_KEY" \
-H "Idempotency-Key: order-1042-charge" \
-H "Content-Type: application/json" \
-d '{"order": "1042", "amount": "..."}'
```

3

If the API does not accept one, check before retrying: ask for the status using your reference and only retry when you are sure it was not done.

4

Retry with growing waits and a maximum number of attempts, never in a tight loop. A little randomness in each wait stops many requests from coming back all at once. Set explicit timeouts in cURL (`CURLOPT_CONNECTTIMEOUT` and `CURLOPT_TIMEOUT`) rather than living with the defaults.

5

On your side, make the database refuse duplicates. A unique constraint on the transaction or event identifier column rejects the second row, even when two requests arrive at the same moment. Take a backup before altering a table: `ALTER TABLE payments ADD UNIQUE (transaction_id);`

6

The same goes for the webhooks you receive: the operator may send the same event more than once, especially if your site was slow to answer. Store the event identifier and ignore repeats, as in **receiving a webhook in PHP**.

The “Pay” button repeats too. The customer clicks twice, or goes back and resubmits the form. Disable the button after the first click, but do not rely on that alone: on the server, use the same order reference so the second request runs into the first instead of creating another charge.

Log every attempt with its reference, time, status code and cURL error. When a customer says “I was charged twice”, that log shows whether there were two charges or one retry the API ignored. See also **calling a web API from PHP**.

Requests from your site to an API keep timing out? Tell us the API’s address and the time of one case, and we will check whether the hosting’s outgoing connection plays a part.

[Open a support ticket](#)

SEE ALSO

[cURL error 28, 7 and 6 in PHP: timeout, connection refused and could not resolve host](#)

[Error 429 Too Many Requests: who is asking too much, and what to do](#)

[Calling a web API from PHP with cURL and reading the JSON reply](#)

[Receiving a webhook in PHP and checking its signature](#)

[Callback URL or return URL: why a customer coming back is not proof of payment](#)

[Synchronous or asynchronous: why a payment or SMS confirmation arrives later](#)



Online version of this document

Interweb is a leading internet services company, with a mission to help businesses to be born and prosper through technology.

2020

FOUNDED

37,000+

CLIENTS

130+

SIGN-UPS A DAY

Interweb Mozambique

RLX Holdings, S.A. • Group RLX Holdings

TAX ID (NUIT)	401 191 607
LEGAL ENTITY NO. (NUEL)	105 045 974
ECONOMIC ACTIVITY CODE (CAE)	62022
REGISTERED OFFICE	Bairro 7 de Setembro Vilankulo 1304 • CEP 0411-02, Inhambane, Mozambique
CONTACT	info@interweb.co.mz • +258 84 635 1111 • +258 86 635 1111 • www.interweb.co.mz
OPENING HOURS	Monday to Friday - 08 AM to 04 PM Saturday & Sunday - Closed