Mozout

Gerado a 13/11/2026 19:46

BASE DE CONHECIMENTO · DOCUMENTO PÚBLICO

Pôr a verificação por SMS no seu site: exemplos em PHP, Node.js e Python

WhatsApp e APIs

CÓDIGO	VERSÃO	PAÍS · IDIOMA
RLX-MZ-KB-2661	v2026.11.13	Moçambique · Português

MozOut Moçambique
RLX Holdings, S.A. · Grupo RLX Holdings
NUIT 401 191 607 · NUEL 105 045 974 · CAE 62022

https://www.meu.mozout.co.mz/knowledgebase/2661

Pôr a verificação por SMS no seu site: exemplos em PHP, Node.js e Python

WhatsApp e APIs Criado em **13 de novembro de 2026** 3 min de leitura



A resposta curta: são dois pedidos `POST` com JSON e a chave no cabeçalho `Authorization: Bearer`: um para `/v1/sms/otp` com o número, outro para `/v1/sms/otp/{id}/verify` com o código que o cliente escreveu. Em qualquer linguagem, a lógica é a mesma: guarde o `id` na sessão do servidor, confira, e decida pelo campo `valid`. Os exemplos abaixo seguem os da página do SMS de Verificação, com a chave lida do ambiente e uma `Idempotency-Key` no envio.

Antes de copiar

1. **Uma chave Sandbox** com `sms:write`, criada na página API do Meu MozOut, guardada numa variável de ambiente (`API_KEY`), nunca no código nem no repositório.

2. **Os pedidos saem do seu servidor.** Nunca chame a API a partir do navegador ou de uma aplicação móvel: a chave ficaria à vista de todos.

3. **O `id` fica na sessão do servidor**, ligado ao utilizador e ao que ele está a fazer.

PHP

```
<?php
// Keep the key on the server, e.g. in an environment variable
function api_call(string $path, array $body, ?string $idem = null): array {
$h = ["Authorization: Bearer " . getenv("API_KEY"), "Content-Type: application/json"];
if ($idem) { $h[] = "Idempotency-Key: " . $idem; }
$ch = curl_init("https://api.mozout.co.mz" . $path);
curl_setopt_array($ch, [CURLOPT_RETURNTRANSFER => true, CURLOPT_POST => true,
CURLOPT_HTTPHEADER => $h, CURLOPT_POSTFIELDS => json_encode($body), CURLOPT_TIMEOUT =>
30]);
```

```
$r = json_decode(curl_exec($ch), true);
return $r["data"] ?? ["error" => $r["error"] ?? null];
}
// 1. Send the code
$otp = api_call("/v1/sms/otp", ["number" => "841234567"], bin2hex(random_bytes(16)));
$_SESSION["otp_id"] = $otp["id"];
// 2. Check the code the user typed
$check = api_call("/v1/sms/otp/" . $_SESSION["otp_id"] . "/verify", ["code" => $_POST["code"]]);
if (!empty($check["valid"])) { /* number confirmed */ }
```

Node.js

Com o `fetch` e o `crypto.randomUUID()` que já vêm no Node.js 20 ou mais recente, dentro de uma função `async`:

```
const API = "https://api.mozout.co.mz/v1/sms/otp";
const headers = { "Authorization": "Bearer " + process.env.API_KEY,
"Content-Type": "application/json" };
// 1. Send the code
const sent = await (await fetch(API, { method: "POST",
headers: { ...headers, "Idempotency-Key": crypto.randomUUID() },
body: JSON.stringify({ number: "841234567" }) })).json();
// 2. Check the code the user typed
const check = await (await fetch(`${API}/${sent.data.id}/verify`, {
method: "POST", headers, body: JSON.stringify({ code: userCode }) })).json();
if (check.data && check.data.valid) { /* number confirmed */ }
```

Python

Com a biblioteca `requests`:

```
import os, uuid, requests
API = "https://api.mozout.co.mz/v1/sms/otp"
H = {"Authorization": "Bearer " + os.environ["API_KEY"]}
# 1. Send the code
otp = requests.post(API, headers={**H, "Idempotency-Key": str(uuid.uuid4())},
json={"number": "841234567"}, timeout=30).json()["data"]
# 2. Check the code the user typed
check = requests.post(f"{API}/{otp['id']}/verify", headers=H,
json={"code": user_code}, timeout=30).json().get("data", {})
if check.get("valid"):
pass # number confirmed
```

O que os exemplos deixam para si

Falta	Porquê
Tratar os erros HTTP	Um `422` (número inválido) ou um `429` (limite) não trazem `data`. Veja **ler as respostas da API de verificação**.
Mensagens para cada estado	`valid: false` pode querer dizer engano, expirado, falhado ou substituído: leia `status`.
Travões no formulário	Captcha e limites por IP antes de chamar a API. Veja **a fraude pelo formulário do código**.
Reenviar	`POST /v1/sms/otp/{id}/resend`, com o mesmo `id`.

Se a conferência falhar por rede, não a repita às cegas.
Cada conferência com o código errado gasta uma tentativa. Veja primeiro o estado com `GET /v1/sms/otp/{id}` (precisa de `sms:read`) e só depois decida.

Teste tudo na Sandbox:
A resposta do envio traz o `code`, e o seu teste automático pode conferi-lo sem telemóvel. Em Produção, troque só a chave. Veja **testar a verificação na Sandbox**.

RECOMENDADO

Quer ver os exemplos completos e criar a sua chave? Veja a página da API.

Ver a página da API →

VEJA TAMBÉM

- API de verificação por SMS: geramos, enviamos e conferimos o código, e o seu sistema guarda só o id →
- Testar a verificação por SMS na Sandbox: o código vem na resposta →
- Ler as respostas da API de verificação: valid false, 409, 422 e 429 →
- Estados de um código de verificação: pending, verified, expired, failed e cancelled →
- Enviar SMS pela API: chave, permissões, Sandbox e o primeiro pedido →

Uma chave de API ou um token ficou exposto: o que fazer, pela ordem certa

MozOut Moçambique

RLX Holdings, S.A. · Grupo RLX Holdings

NUIT	401 191 607
NUEL	105 045 974
CAE	62022
SEDE	Bairro 7 de Setembro Vilankulo 1304 · CEP 0411-02, Inhambane, Moçambique
CONTACTOS	info@mozout.co.mz · +258 84 635 1111 · +258 86 635 1111 · www.mozout.co.mz
HORÁRIO	2ª. à 6ª. Feira - 08 à 16 Horas Sábado & Domingo - Fechado

Versão online deste documento

RLX-MZ-KB-2661 · v2026.11.13 · Gerado a 13/11/2026 19:46