



API REFERENCE · API V1

OTP

Let SMSRay generate, send and check verification codes. You never store the code, and abuse limits are enforced for you.

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

1 Send an OTP

POST /sms/otp/send <https://api.smsray.com/api/sms/v1/sms/otp/send>

Generate a 6-digit code, send it as a branded one-segment SMS, and store only its SHA-256 hash. The code is valid for 5 minutes.

HEADERS

x-api-key string **required**

Your secret key, ls_live_ followed by 48 hex characters.

content-type string **required**

application/json

Idempotency-Key string **optional**

1–200 printable ASCII characters. Safe retries for 24 h. See [Idempotency](#).

BODY (JSON)

to string **required**

Recipient. **E.164 is recommended** (+9779801234567); the national format is accepted for your workspace's country (for a Nepal workspace 98XXXXXXX, 97... or 96..., with or without 977). Must match `^\+?\d{7,15}$`. See [Numbers & countries](#).

purpose string **optional**

Up to 50 characters. Defaults to default. Codes are scoped to (your client, to, purpose), so login and reset-password never collide.

ip string **optional**

Your end user's IP address, up to 64 characters. Enables the per-IP limit (10 OTPs per 10 minutes).

senderId string **optional**

Optional; same meaning as on /sms/send.

EXAMPLE REQUESTS

cURL

```
curl https://api.smsray.com/api/sms/v1/sms/otp/send \
-H "x-api-key: $SMSRAY_API_KEY" \
-H "content-type: application/json" \
-d '{ "to": "+9779801234567", "purpose": "login", "ip": "203.0.113.7" }'
```

Node.js

```
const res = await fetch("https://api.smsray.com/api/sms/v1/sms/otp/send", {
  method: "POST",
  headers: {
    "x-api-key": process.env.SMSRAY_API_KEY,
    "content-type": "application/json",
  },
  // pass the end user's IP so per-IP limits protect you from abuse
  body: JSON.stringify({ to: "+9779801234567", purpose: "login", ip: req.ip }),
});

if (res.status === 429) {
  const wait = Number(res.headers.get("retry-after") ?? 60);
  // show "try again in {wait}s" in your UI
}

const { otpId, messageId, expiresInSeconds } = await res.json();
```

Python

```
import os, requests

r = requests.post(
    "https://api.smsray.com/api/sms/v1/sms/otp/send",
    headers={"x-api-key": os.environ["SMSRAY_API_KEY"]},
    json={"to": "+9779801234567", "purpose": "login", "ip": "203.0.113.7"},
    timeout=10,
)
if r.status_code == 429:
    print("Retry in", r.headers.get("Retry-After"), "seconds")
else:
    print(r.json()["otpId"])
```

PHP

```
$ch = curl_init("https://api.smsray.com/api/sms/v1/sms/otp/send");
curl_setopt_array($ch, [
    CURLOPT_POST => true,
    CURLOPT_RETURNTRANSFER => true,
    CURLOPT_HTTPHEADER => [
        "x-api-key: " . getenv("SMSRAY_API_KEY"),
        "content-type: application/json",
    ],
    CURLOPT_POSTFIELDS => json_encode(["to" => "+9779801234567", "purpose" => "login", "ip" =>
» "203.0.113.7"]),
]);
$res = json_decode(curl_exec($ch), true);
echo $res["otpId"] ?? $res["error"];
```

Go

```
// imports: bytes, encoding/json, fmt, log, net/http, os
body, _ := json.Marshal(map[string]string{"to": "+9779801234567", "purpose": "login", "ip":
» "203.0.113.7"})
req, _ := http.NewRequest("POST", "https://api.smsray.com/api/sms/v1/sms/otp/send",
» bytes.NewReader(body))
req.Header.Set("x-api-key", os.Getenv("SMSRAY_API_KEY"))
req.Header.Set("content-type", "application/json")
```

```
res, err := http.DefaultClient.Do(req)
if err != nil {
    log.Fatal(err)
}
defer res.Body.Close()

var out struct {
    OtpID      string `json:"otpId"`
    ExpiresInSeconds int    `json:"expiresInSeconds"`
}
json.NewDecoder(res.Body).Decode(&out)
fmt.Println(res.StatusCode, out.OtpID, out.ExpiresInSeconds)
```

RESPONSES

200

```
{
  "success": true,
  "otpId": "9d5c0b7e-3a1f-4e62-8b1d-6a4f2c9e0d17",
  "messageId": "4f8a2c61-0b9e-4d3a-a7c5-1e6f9b2d8c40",
  "expiresInSeconds": 300
}
```

429 · Resend too soon

```
{
  "error": "Please wait before requesting another code",
  "code": "rate_limited"
}
```

RESPONSE FIELDS

otpId string (uuid) **required**

Id of the code record. You never need it to verify.

messageId string (uuid) **required**

The SMS that carries the code; track it like any message.

expiresInSeconds integer **required**

Always 300 (5 minutes).

ERRORS

HTTP	CODE	WHEN
400	invalid_request	Validation failed (see details).
400	invalid_number	The number can't receive SMS — for example a landline, or a number that fails its country's mobile rules.
402	insufficient_balance	Not enough balance for one OTP-rate segment.
401	unauthorized	The x-api-key header is missing ("Missing x-api-key") or the key is unknown ("Invalid API key").
403	client_disabled	The API client that owns this key has been disabled in the dashboard.

HTTP	CODE	WHEN
403	forbidden	OTP messages are not allowed on your plan or workspace.
403	workspace_pending	Your workspace has not been activated yet, so it cannot send.
403	workspace_suspended	Your workspace is suspended.
409	idempotency_conflict	The same Idempotency-Key was used with a different body, or the first request is still running (Retry-After: 1).
422	destination_not_supported	A valid number in a country your workspace can't send to yet. details is { country, allowed }; show "We don't deliver to <country> yet".
429	rate_limited	Resend cooldown (60 s per number and purpose), 3 OTPs per number per 10 minutes, 10 per end-user IP per 10 minutes, 60 requests per minute per calling IP, or your per-client rate. Retry-After is set whenever it is known.
503	unavailable	Delivery is temporarily unavailable. Nothing was charged; retry with backoff.
500	server_error	Something went wrong on our side. The body never contains a stack trace.

NOTES

- The SMS reads: <App>: 482913 is your verification code. Valid for 5 minutes. Do not share it. <App> is your API client's brand name (GSM-7, up to 30 characters, set in the dashboard).
- The message uses SMSRay's built-in OTP template, so no template approval is needed. It is billed at your OTP rate and always fits in one GSM-7 segment.
- The code is never stored or shown in plain text. In the dashboard and in GET /sms/messages/:id the text appears as (OTP hidden).
- The per-number limit counts across every SMSRay customer, so one phone number cannot be flooded with codes from many apps.
- OTPs are sent immediately and are never deferred.

2 Verify an OTP

POST /sms/otp/verify <https://api.smsray.com/api/sms/v1/sms/otp/verify>

Check the code your user typed against the latest active code for the same number and purpose. Codes are single use.

HEADERS

x-api-key string **required**

Your secret key, ls_live_ followed by 48 hex characters.

content-type string **required**

application/json

Idempotency-Key string **optional**

1–200 printable ASCII characters. Safe retries for 24 h. See [Idempotency](#).

BODY (JSON)

to string **required**

The same number you sent the code to.

code string **required**

The 6 digits your user entered.

purpose string **optional**

Must match the purpose used on send. Defaults to default.

EXAMPLE REQUESTS**cURL**

```
curl https://api.smsray.com/api/sms/v1/sms/otp/verify \
-H "x-api-key: $SMSRAY_API_KEY" \
-H "content-type: application/json" \
-d '{ "to": "+9779801234567", "purpose": "login", "code": "482913" }'
```

Node.js

```
const res = await fetch("https://api.smsray.com/api/sms/v1/sms/otp/verify", {
  method: "POST",
  headers: {
    "x-api-key": process.env.SMSRAY_API_KEY,
    "content-type": "application/json",
  },
  body: JSON.stringify({ to: "+9779801234567", purpose: "login", code: req.body.code }),
});
const { verified, reason } = await res.json();

// A wrong code is a normal 200 answer, not an HTTP error
if (!verified) {
  // reason: "invalid_code" | "too_many_attempts" | "no_active_otp"
}
```

Python

```
import os, requests

r = requests.post(
    "https://api.smsray.com/api/sms/v1/sms/otp/verify",
    headers={"x-api-key": os.environ["SMSRAY_API_KEY"]},
    json={"to": "+9779801234567", "purpose": "login", "code": "482913"},
    timeout=10,
).json()

if r["verified"]:
    print("Signed in")
else:
    print("Rejected:", r["reason"])
```

PHP

```
$ch = curl_init("https://api.smsray.com/api/sms/v1/sms/otp/verify");
curl_setopt_array($ch, [
    CURLOPT_POST => true,
    CURLOPT_RETURNTRANSFER => true,
    CURLOPT_HTTPHEADER => [
```

```
"x-api-key: " . getenv("SMSRAY_API_KEY"),
"content-type: application/json",
],
CURLOPT_POSTFIELDS => json_encode(["to" => "+9779801234567", "purpose" => "login", "code" =>
» "482913"]),
]);
$res = json_decode(curl_exec($ch), true);
// A wrong code is a normal 200 answer, not an HTTP error
echo $res["verified"] ? "Signed in" : "Rejected: " . $res["reason"];
```

Go

```
// imports: bytes, encoding/json, fmt, log, net/http, os
body, _ := json.Marshal(map[string]string{"to": "+9779801234567", "purpose": "login", "code":
» "482913"})
req, _ := http.NewRequest("POST", "https://api.smsray.com/api/sms/v1/sms/otp/verify",
» bytes.NewReader(body))
req.Header.Set("x-api-key", os.Getenv("SMSRAY_API_KEY"))
req.Header.Set("content-type", "application/json")

res, err := http.DefaultClient.Do(req)
if err != nil {
    log.Fatal(err)
}
defer res.Body.Close()

var out struct {
    Verified bool `json:"verified"`
    Reason    string `json:"reason"`
}
json.NewDecoder(res.Body).Decode(&out)
// A wrong code is a normal 200 answer, not an HTTP error
if !out.Verified {
    fmt.Println("Rejected:", out.Reason) // invalid_code | too_many_attempts | no_active_otp
}
```

RESPONSES

200 · Correct code

```
{ "success": true, "verified": true }
```

200 · Wrong code

```
{ "success": true, "verified": false, "reason": "invalid_code" }
```

RESPONSE FIELDS

verified boolean **required**

true once, for the correct code. The code is then consumed.

reason string **optional**

Present when verified is false: `invalid_code`, `too_many_attempts` (5 wrong tries) or `no_active_otp` (none sent, expired or already used).

ERRORS

HTTP	CODE	WHEN
400	invalid_request	A required field is missing or malformed.
401	unauthorized	The x-api-key header is missing ("Missing x-api-key") or the key is unknown ("Invalid API key").
403	client_disabled	The API client that owns this key has been disabled in the dashboard.
403	workspace_pending	Your workspace has not been activated yet, so it cannot send.
403	workspace_suspended	Your workspace is suspended.
409	idempotency_conflict	The same Idempotency-Key was used with a different body, or the first request is still running (Retry-After: 1).
429	rate_limited	You exceeded your per-client request rate (default 20 requests per second). Honour Retry-After.
500	server_error	Something went wrong on our side. The body never contains a stack trace.

NOTES

- A wrong code is not an HTTP error. Always read `verified`; never treat a 200 as success on its own.
- Each wrong code counts as an attempt. After 5 wrong attempts the code is locked and every further check returns `too_many_attempts` — send a new code.
- Codes expire 5 minutes after they are sent and are deleted automatically afterwards.

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

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