



SMSRay

Developer documentation · A product of Lacspace

API REFERENCE · API V1

SMSRay API Reference

A small, predictable REST API for OTP and transactional SMS. JSON in, JSON out, one header to authenticate, signed webhooks for every status change. Nepal is live today; more countries are on the way.

BASE URL

`https://api.smsray.com/api/sms/v1`

AUTHENTICATION

`x-api-key: ls_live_...` (48 hex characters)

NUMBERS

E.164 recommended, e.g. `+9779801234567`

ENDPOINTS

POST	/sms/send
POST	/sms/otp/send
POST	/sms/otp/verify
GET	/sms/messages/:id
GET	/sms/balance
GET	/sms/coverage

Updated 2026-10-10, generated 2026-10-10 from smsray.com/docs. The web documentation is always the current version.

<https://smsray.com/docs>

CONTENTS

SMSRay API Reference

1. Quickstart	3
2. Authentication	5
3. Test mode	7
4. API reference	8
5. Send SMS	9
5.1 POST /sms/send	9
6. OTP	12
6.1 POST /sms/otp/send	12
6.2 POST /sms/otp/verify	14
7. Message status	16
7.1 GET /sms/messages/:id	16
8. Balance	18
8.1 GET /sms/balance	18
9. Coverage	20
9.1 GET /sms/coverage	20
10. Webhooks	21
11. Numbers & countries	25
12. Errors and codes	26
13. Rate limits	29
14. Idempotency	30
15. Segments & encoding	31
16. Changelog	33
17. Support	34

GET STARTED

1. Quickstart

From API key to your first verified OTP in about five minutes. OTP uses SMSRay's built-in template, so it works on day one.

1 Sign in and open your workspace

[Talk to sales](#) and pick a plan (see [pricing](#)); the SMSRay team creates your workspace. Then sign in at app.smsray.com with your phone number or email — we send you a one-time code, no password needed.

ACTIVATION

Until your workspace is activated it shows as pending. You can create keys and call the GET endpoints straight away; POST requests return 403 workspace_pending until activation.

2 Create an API key

In the dashboard, open **Clients** and create an API client — one per app or environment works well. Give it a brand name: it appears at the start of every OTP message. You get two secrets, shown once. The portal's [developer page](#) shows your keys and live examples pre-filled for them.

- ls_live_... — your API key, sent as x-api-key.
- whsec_... — your webhook signing secret.

Store them in your secret manager or environment, never in client-side code:

```
shell
```

```
export SMSRAY_API_KEY="ls_live_..."
```

TEST MODE

Integrating in CI or staging? Put the key in [test mode](#): messages are recorded but never sent, and they're free. For this quickstart leave it off, so the code really reaches your phone.

3 Send your first OTP

Send a code to your own mobile number. Use E.164 (+9779801234567); the national format of your workspace's country works too. OTP uses SMSRay's built-in template, so it works on day one — no template approval needed.

```
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" }'
```

Also in Node.js, Python, PHP, Go: smsray.com/docs/quickstart

The reply comes back in milliseconds and the SMS arrives a moment later:

200 application/json

```
{ "success": true, "otpId": "9d5c0b7e-...", "messageId": "4f8a2c61-...", "expiresInSeconds": 300 }
```

4 Verify the code

Post the 6 digits from the SMS with the same to and purpose. A wrong code is a normal 200 answer with `verified: false` — always read `verified`.

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" }'
```

Also in Node.js, Python, PHP, Go: smsray.com/docs/quickstart

200 application/json

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

5 Send an SMS and track it

APPROVED TEMPLATES

Every custom message uses a template approved by SMSRay, which protects deliverability and keeps spam off the networks. Create one in the dashboard (or start from the template library); we aim to review it within one business day.

cURL

```
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." }'
```

Also in Node.js, Python, PHP, Go: smsray.com/docs/quickstart

You get a `messageId` back immediately. The status moves from `queued` to `sent` to `delivered`; look it up any time:

cURL

```
curl https://api.smsray.com/api/sms/v1/sms/messages/<messageId> \
-H "x-api-key: $SMSRAY_API_KEY"
```

Also in Node.js, Python, PHP, Go: smsray.com/docs/quickstart

6 Add a webhook

Polling is fine for testing. In production, set a webhook URL (`https://`) on your API client and SMSRay POSTs a signed `message.status` event on every change, plus `message.inbound` when someone replies. [Set up webhooks](#).

Next steps

- **OTP reference** — Cooldowns, attempt caps and per-number limits. smsray.com/docs/otp
- **Retry safely** — Never double-send on a network error. smsray.com/docs/idempotency
- **Numbers & countries** — E.164, national format and coverage. smsray.com/docs/numbers
- **Handle errors** — Every code and what to do about it. smsray.com/docs/errors

GET STARTED

2. Authentication

Every request carries your secret key in the `x-api-key` header. Keys belong to an API client inside your workspace.

The x-api-key header

Send your key in the `x-api-key` header on every request, over HTTPS. There are no sessions or OAuth flows on the product API.

Authenticated request

```
curl https://api.smsray.com/api/sms/v1/sms/balance \
-H "x-api-key: ls_live_3f9c..."
```

API keys and clients

- A key looks like `ls_live_` followed by 48 hex characters. The dashboard shows only its prefix (for example `ls_live_3f9c`) afterwards.
- We store only a SHA-256 hash of the key. Nobody — including us — can show it to you again, so copy it when it is created.
- Each key belongs to one **API client** in your workspace. The client holds its own brand name, webhook URL, webhook secret, request rate and counters.
- Balance, rates and allowed message types belong to the workspace and are shared by all its clients.
- A key can only see its own messages. Looking up a message sent by another client returns 404, even inside the same workspace.
- A key can be put in [test mode](#): requests are accepted and recorded, nothing is sent and nothing is charged.

Rotating a key

Workspace owners can rotate a client's key from the dashboard. The new key works at once and **the old key stops working immediately**, so deploy the new key first-thing or rotate during a quiet window. To stop a client entirely, disable it instead: its requests return 403 `client_disabled` and its message history stays intact.

Workspace status

STATUS	GET REQUESTS	POST REQUESTS
pending	Work	403 workspace_pending
active	Work	Work
suspended	May be refused	403 workspace_suspended

GET /sms/balance returns workspaceStatus, so your app can check it at start-up.

Auth errors

HTTP	CODE	WHEN
401	unauthorized	"Missing x-api-key" — the header was not sent.
401	unauthorized	"Invalid API key" — the key is unknown or was rotated.
403	client_disabled	The client that owns this key is disabled.
403	workspace_pending	POST before your workspace is activated.
403	workspace_suspended	POST while your workspace is suspended.

Good practice

- Call the API from your server only. A key in a mobile app or browser bundle can be extracted and used to spend your balance.
- Use a separate client per app and environment, so you can rotate or disable one without touching the others.
- Keep keys in environment variables or a secret manager, and out of logs and source control.
- Every change in the portal — keys created, rotated or disabled — is recorded in your workspace audit log.

GET STARTED

3. Test mode

Build and test against the real API without sending a single SMS or spending balance.

What test mode does

An account owner can switch any API key (API client) into test mode from the dashboard. While it is on:

- Messages are **accepted and recorded**, so your code gets real replies and real `messageIds`.
- Messages are **never sent** to the phone.
- Messages are **free** — nothing is reserved from your balance.
- Each message ends with status `test`, visible in the dashboard and with `GET /sms/messages/:id`.

When to use it

- Automated tests and CI pipelines that exercise your send path.
- A staging environment, on its own API client, so staging can never message real customers.
- Load-testing your own queueing and retry logic without spending balance.

Turning it on and off

- Open **Clients** in the dashboard, pick the client and switch test mode on or off. It applies to that client's key only; other keys keep sending.
- Turn it off before going live. A message sent while test mode is on is never delivered later.
- For OTP flows in test mode no SMS arrives, so test the "code received" step with a key that has test mode off.

ONE KEY PER ENVIRONMENT

Keep production and test on separate API clients rather than flipping one key back and forth. It keeps audit logs clear and makes a forgotten switch harmless.

API REFERENCE

4. API reference

Six endpoints, two webhook events, one auth header and one error shape.

Basics

Base URL	https://api.smsray.com/api/sms/v1
Authentication	x-api-key: <code>ls_live_...</code> on every request. See Authentication .
Format	JSON over HTTPS. Timestamps are ISO 8601 UTC.
Numbers	E.164 recommended; national format accepted for your workspace's country. See Numbers & countries .
Errors	{ "error", "code", "details"? } with a conventional HTTP status. See Errors .
Safe retries	Idempotency-Key on any POST, kept 24 h. See Idempotency .

Endpoints

METHOD	PATH	WHAT IT DOES
POST	/sms/send	Send an SMS, as free text or from an approved template.
POST	/sms/otp/send	Generate and send a 6-digit code.
POST	/sms/otp/verify	Check the code your user typed.
GET	/sms/messages/:id	Look up a message and its status.
GET	/sms/balance	Workspace balance, rates and your counters.
GET	/sms/coverage	Where this key can send, and where SMSRay delivers.

Webhook events

EVENT	WHEN
message.status	A message reaches sent, delivered, undelivered or failed.
message.inbound	A recipient replies to one of your messages.

Machine-readable specs

- [OpenAPI 3.1](#) — </openapi.json>: every public v1 endpoint, generated from these docs. Import it into your API client, code generator or gateway.
- [Postman collection](#) — </postman.json>: ready-made requests. Set the `apiKey` variable and send.

- [API reference PDF](#): the whole reference as one document.

YOUR KEYS AND LIVE EXAMPLES

Signed in? The portal's [developer page](#) shows your own keys with every example pre-filled for them.

API REFERENCE

5. Send SMS

Send one message to one mobile number. The reply comes back in milliseconds with a `messageId`; delivery is reported afterwards.

5.1 Send an SMS

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

Queue one SMS to one mobile number, as free text or from an approved template. The cost is reserved from your workspace balance at accept time and the message goes straight into delivery.

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](#).

`text` string **optional**

Free text. Send **exactly one** of `text` or `templateId`. At least 1 character, at most 6 segments (GSM-7: 918 characters, UCS-2: 402 code units). When your workspace requires templates (the default) the text must match one of your approved templates of the same `type`.

`templateId` string **optional**

An approved template of your workspace. SMSRay renders it with `variables`. Send exactly one of `text` or `templateId`.

`variables` object **optional**

Values for the template's `{{variables}}`, as strings or numbers. Every variable is required; each value is 1–100 characters with no line break.

`type` string **optional**

`transactional` (default), `otp` or `promotional`. Must be one of your workspace's `allowedTypes`; `promotional` is available on Enterprise only. With `templateId`, the template's type is used when omitted.

`senderId` string **optional**

Stored as `from` on the message. Not validated in v1 — sender-ID approval is not available yet.

EXAMPLE REQUEST

cURL

```
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." }'
```

Also in Node.js, Python, PHP, Go: smsray.com/docs/send-sms

RESPONSES

200

```
{
  "success": true,
  "messageId": "2b1f6c1e-8d4f-4c55-9a51-0d2b0b7f3a11",
  "status": "queued",
  "segments": 1,
  "cost": 0.5,
  "encoding": "GSM7",
  "from": "SMSRay"
}
```

400 · Validation error

```
{
  "error": "Invalid request",
  "code": "invalid_request",
  "details": [
    { "path": ["to"], "message": "Required", "code": "invalid_type" }
  ]
}
```

403 · No matching template

```
{
  "error": "This message doesn't match an approved template. Submit the template for approval in your SMSRay dashboard, or send with an approved templateId.",
  "code": "template_required"
}
```

422 · Country not supported yet

```
{
  "error": "We don't deliver to India yet. This workspace can send to: Nepal.",
  "code": "destination_not_supported",
  "details": { "country": "IN", "allowed": ["NP"] }
}
```

RESPONSE FIELDS

messageId string (uuid) **required**

Use it with GET /sms/messages/:id and to match webhook events.

status string **required**

Always queued in this reply.

segments integer **required**

How many SMS parts the text needs.

cost number **required**

Your rate for type × segments, reserved from the balance.

encoding string **required**

GSM7 or UCS2.

from string **required**

The sender recorded for the message: your senderId if given, otherwise the default sender.

ERRORS

HTTP	CODE	WHEN
400	invalid_request	Validation failed (see details), both or neither of text and templateId were sent, or the text needs more than 6 segments.
400	invalid_number	The number can't receive SMS — for example a landline, or a number that fails its country's mobile rules.
400	invalid_variables	With templateId: a variable is missing, unknown, empty, longer than 100 characters or contains a line break.
402	insufficient_balance	Your workspace balance is lower than the cost of this message.
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	forbidden	This message type is not allowed on your plan or workspace.
403	template_required	The text doesn't match an approved template of the same type, or templateId is not an approved template of this 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	You exceeded your per-client request rate (default 20 requests per second). Honour Retry-After.
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 reply always says `queued`. Follow the message with `GET /sms/messages/:id` or, better, a `message.status` webhook. No webhook is sent for the initial `queued`.
- Every custom message must use a template approved by SMSRay. Approval protects deliverability and keeps spam off the networks; we aim to review new templates within one business day. OTPs sent with `POST /sms/otp/send` use SMSRay's built-in template, so OTP works on day one.
- Messages are delivered over the recipient's mobile network, whichever one they are on, with automatic retries. If delivery finally fails, the message ends as `failed` and the reserved cost is refunded. `undelivered` is not refunded.
- Numbers are stored in E.164, so `to` in lookups and webhooks is always `+<country code><number>`.
- Send an `Idempotency-Key` on every send so network retries never double-send or double-bill.
- On a key in [test mode](#) the message is accepted and recorded with status `test`, but never sent and never charged.

API REFERENCE

6. OTP

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

6.1 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 REQUEST**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" }'
```

Also in Node.js, Python, PHP, Go: smsray.com/docs/otp

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").

HTTP	CODE	WHEN
403	client_disabled	The API client that owns this key has been disabled in the dashboard.
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.

6.2 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 REQUEST

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" }'
```

Also in Node.js, Python, PHP, Go: smsray.com/docs/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).

HTTP	CODE	WHEN
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.

API REFERENCE

7. Message status

Look up any message you sent with this key. Use it for support tools and reconciliation; use webhooks for real-time updates.

7.1 Retrieve a message

GET /sms/messages/:id <https://api.smsray.com/api/sms/v1/sms/messages/:id>

Fetch the current state of a message sent with this API key: status, encoding, segments, cost and timestamps.

HEADERS

x-api-key string **required**

Your secret key, **ls_live_** followed by 48 hex characters.

PATH PARAMETERS

id string (uuid) **required**

The messageId returned by /sms/send or /sms/otp/send.

EXAMPLE REQUEST

cURL

```
curl https://api.smsray.com/api/sms/v1/sms/messages/2b1f6c1e-8d4f-4c55-9a51-0d2b0b7f3a11 \
-H "x-api-key: $SMSRAY_API_KEY"
```

Also in Node.js, Python, PHP, Go: smsray.com/docs/messages

RESPONSES

200

```
{
  "_id": "2b1f6c1e-8d4f-4c55-9a51-0d2b0b7f3a11",
  "to": "+9779801234567",
  "from": "SMSRay",
  "text": "Your order #1042 has shipped.",
}
```

```
{
  "type": "transactional",
  "encoding": "GSM7",
  "segments": 1,
  "cost": 0.5,
  "status": "delivered",
  "error": "",
  "attempts": 1,
  "submittedAt": "2026-10-10T04:15:01.000Z",
  "sentAt": "2026-10-10T04:15:03.000Z",
  "deliveredAt": "2026-10-10T04:15:07.000Z",
  "failedAt": null,
  "createdAt": "2026-10-10T04:15:00.000Z",
  "updatedAt": "2026-10-10T04:15:07.000Z"
}
```

404 · Unknown id

```
{ "error": "Message not found", "code": "not_found" }
```

RESPONSE FIELDS

status string **required**

queued, submitting, submitted, sent, delivered, undelivered, failed or rejected (test for keys in test mode).

to string **required**

The recipient in E.164.

text string **required**

The message body, or (OTP hidden) for messages created by /sms/otp/send.

from string **required**

The sender recorded for the message.

attempts integer **required**

How many delivery attempts were made.

error string **required**

A neutral failure code when one is known (network_unavailable, delivery_timeout, carrier_rejected, invalid_number, delivery_failed), otherwise empty.

dlr object **optional**

Delivery report { status, code, raw, at }, present when the network returned one.

submittedAt · sentAt · deliveredAt · failedAt string | null **required**

ISO 8601 UTC timestamps for each transition.

ERRORS

HTTP	CODE	WHEN
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.
404	not_found	No message with this id was sent by this API key. Messages from other keys — even in the same workspace — are not visible.

HTTP	CODE	WHEN
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

- Polling works, but webhooks are cheaper and faster. If you poll, back off (for example 2 s, 5 s, 15 s, 60 s) and stop at a final status: `delivered`, `undelivered`, `failed` or `rejected`.
- Internal delivery details are never returned.

API REFERENCE

8. Balance

Check how much you can send before you send it, and alert your team well before the balance runs out.

8.1 Retrieve balance

GET /sms/balance <https://api.smsray.com/api/sms/v1/sms/balance>

Read your workspace balance, per-type rates, allowed message types and this API client's counters.

HEADERS

`x-api-key` string **required**

Your secret key, `ls_live_` followed by 48 hex characters.

EXAMPLE REQUEST

cURL

```
curl https://api.smsray.com/api/sms/v1/sms/balance \
-H "x-api-key: $SMSRAY_API_KEY"
```

Also in Node.js, Python, PHP, Go: smsray.com/docs/balance

RESPONSES

200

```
{
  "balance": 5000.0,
  "rate": { "transactional": 0.5, "otp": 0.5, "promotional": 0.5 },
  "allowedTypes": ["transactional", "otp"],
  "counters": { "sent": 120, "delivered": 110, "failed": 4 },
  "workspaceStatus": "active"
}
```

RESPONSE FIELDS

balance number **required**

Credit remaining for the current monthly period, in your workspace's currency (NPR for Nepal workspaces), shared by every API client in the workspace.

rate object **required**

Price per SMS segment for transactional, otp and promotional. $\text{Cost} = \text{rate} \times \text{segments}$. promotional is enabled only on Enterprise; check `allowedTypes`.

allowedTypes string[] **required**

Message types this workspace may send. Others return 403 forbidden.

counters object **required**

This API client's own sent, delivered and failed totals.

workspaceStatus string **required**

active, pending or suspended.

ERRORS

HTTP	CODE	WHEN
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.
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

- GET requests keep working while your workspace is pending, so you can integrate and check your setup before activation.
- Your rate is your plan's per-SMS price. See [pricing](#). Remaining balance expires at the end of each monthly period; top-ups are charged at your plan rate.
- When the balance is lower than a message's cost, the send returns 402 `insufficient_balance`.
- A message's cost is refunded only when delivery finally fails (`failed`). `undelivered` is not refunded.

API REFERENCE

9. Coverage

Ask the API where you can send instead of hard-coding countries. Nepal is live today; more countries are on the way.

9.1 Retrieve coverage

GET /sms/coverage <https://api.smsray.com/api/sms/v1/sms/coverage>

Where this API key can send today: your workspace's country, the destinations allowed for it, and the countries SMSRay delivers to now or soon.

HEADERS

x-api-key string **required**

Your secret key, ls_live_ followed by 48 hex characters.

EXAMPLE REQUEST

cURL

```
curl https://api.smsray.com/api/sms/v1/sms/coverage \
-H "x-api-key: $SMSRAY_API_KEY"
```

Also in Node.js, Python, PHP, Go: smsray.com/docs/coverage

RESPONSES

200

```
{
  "country": "NP",
  "allowed": ["NP"],
  "live": [{ "code": "NP", "name": "Nepal" }],
  "comingSoon": [
    { "code": "BD", "name": "Bangladesh" },
    { "code": "IN", "name": "India" }
  ]
}
```

RESPONSE FIELDS

country string **required**

Your workspace's country (ISO 3166-1 alpha-2). National-format numbers are read as numbers of this country.

allowed string[] **required**

Countries this key can send to right now. A number elsewhere returns 422 destination_not_supported.

live object[] **required**

{ code, name } for every country SMSRay delivers to today.

comingSoon object[] **required**

{ code, name } for countries we are preparing. Not sendable yet.

ERRORS

HTTP	CODE	WHEN
401	unauthorized	The <code>x-api-key</code> 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.
429	rate_limited	You exceeded your per-client request rate (default 20 requests per second). Honour <code>Retry-After</code> .
500	server_error	Something went wrong on our side. The body never contains a stack trace.

NOTES

- Use allowed to validate phone numbers in your sign-up form before you send, and to pre-select the country picker with country.
- A public, unauthenticated variant without the workspace fields is available at `GET https://api.smsray.com/api/sms/coverage`. It returns `{ live, comingSoon }` and is cached for 5 minutes.
- Today Nepal is live; India and Bangladesh are coming soon. The response is always the current list, so don't hard-code it.

API REFERENCE

10. Webhooks

SMSRay POSTs a signed JSON event to your webhook URL whenever a message changes status or a reply arrives.

Setting up

- In the dashboard, set a webhook URL on your API client. It must use `https://`.
- Copy the signing secret (`whsec_...`) shown when the client was created or the secret was rotated.
- Each event is a `POST` with `content-type: application/json` and a single event object as the body.
- Respond with any `2xx` within 10 seconds. Do slow work after you respond.

Request headers

HEADER	VALUE
<code>x-lacsapace-signature</code>	<code>t=<unix seconds>,v1=<hex></code> — HMAC-SHA256 of <code><t>.<raw body></code> . A second <code>v1=</code> appears during secret rotation.
<code>x-lacsapace-event</code>	<code>message.status</code> or <code>message.inbound</code>
<code>x-lacsapace-delivery</code>	Unique id of this delivery. Retries of the same event reuse it — use it to deduplicate.
<code>user-agent</code>	<code>lacsapace-sms-webhooks/1</code>

HEADER	VALUE
content-type	application/json

message.status

Sent when a message you sent reaches sent, delivered, undelivered or failed.

message.status

```
{
  "type": "message.status",
  "id": "2b1f6c1e-8d4f-4c55-9a51-0d2b0b7f3a11",
  "status": "delivered",
  "to": "+9779801234567",
  "at": "2026-10-10T04:15:07.000Z",
  "messageType": "transactional",
  "segments": 1
}
```

- Never sent on accept — the API reply already told you **queued**.
- **queued** is sent only when a message goes back to **queued**, for example when delivery is deferred or retried.
- Never sent twice for the same message and status, even if a late duplicate report arrives.
- **error** is included only when the status is **failed**, **undelivered** or **queued** and a reason is known.
- **at** is the time of the transition (delivered, sent or failed time), in ISO 8601 UTC.
- Events can arrive out of order after retries. Compare **at** before overwriting a later status.

Status mapping

The API has more statuses than the webhook. They map like this:

MESSAGE STATUS (GET /SMS/MESSAGES/:ID)	WEBHOOK STATUS
submitted, sent	sent
delivered	delivered
undelivered	undelivered
failed, rejected	failed
anything else	queued

message.inbound

Sent when a recipient replies to a message you sent and the reply is attributed to your API client. **inReplyTo** links it to the message they answered when known.

message.inbound

```
{
  "type": "message.inbound",
  "id": "6705a1c2e4b0f3a9d8c71b25",
  "from": "+9779801234567",
}
```

```
"text": "Thanks",
"at": "2026-10-10T04:20:00.000Z",
"inReplyTo": "2b1f6c1e-8d4f-4c55-9a51-0d2b0b7f3a11"
}
```

Verifying signatures

Always verify before trusting a webhook. Recompute HMAC-SHA256 over `<t>.<raw body>` with your secret, compare in constant time against each `v1`, and reject timestamps more than 300 seconds from your clock. Each retry is re-signed with a fresh `t`, so a slow retry still verifies.

Node.js

```
import crypto from "node:crypto";

// x-lacsapace-signature: t=<unix>,v1=<hex>[,v1=<hex during secret rotation>]
export function verifySmsrayWebhook(rawBody, header, secret, toleranceSec = 300) {
  const pairs = header.split(",").map((p) => p.split("="));
  const t = Number(pairs.find(([, k]) => k === "t")?.[1]);
  if (!t || Math.abs(Date.now() / 1000 - t) > toleranceSec) return false;
  const expected = Buffer.from(
    crypto.createHmac("sha256", secret).update(`${t}.${rawBody}`).digest("hex"), "hex");
  return pairs
    .filter(([, k]) => k === "v1")
    .some(([, v]) => {
      const sig = Buffer.from(v, "hex");
      return sig.length === expected.length && crypto.timingSafeEqual(sig, expected);
    });
}
```

Also in Python, PHP, Go: smsray.com/docs/webhooks

USE THE RAW BODY

Verify the exact bytes you received. Parsing and re-serialising JSON changes spacing and key order and the signature will not match.

Node.js (Express)

```
import express from "express";
import { verifySmsrayWebhook } from "../verify.js"; // the function above

const app = express();

// Verify against the RAW body – parse JSON only after the check
app.post("/hooks/sms", express.raw({ type: "application/json" }), async (req, res) => {
  const raw = req.body.toString("utf8");
  const ok = verifySmsrayWebhook(raw, req.get("x-lacsapace-signature") ?? "",
  » process.env.SMSRAY_WEBHOOK_SECRET);
  if (!ok) return res.status(400).send("bad signature");

  res.sendStatus(200); // acknowledge fast, work after

  const event = JSON.parse(raw);
  const deliveryId = req.get("x-lacsapace-delivery"); // dedupe on this
  if (event.type === "message.status") await markStatus(event.id, event.status, event.at);
  if (event.type === "message.inbound") await saveReply(event.from, event.text, event.inReplyTo);
});
```

```
});
```

Also in Python (Flask), PHP, Go (net/http): smsray.com/docs/webhooks

Secret rotation

Workspace owners can rotate the webhook secret from the dashboard and choose a grace period of 0–168 hours (24 h by default). During the grace period, every delivery carries **two** signatures: one **v1=** made with the new secret and one with the previous secret.

During rotation

```
x-lacspace-signature: t=1760069707,v1=5f2c...(new secret),v1=a91e...(previous secret)
```

- Because the verifiers above accept any matching **v1**, your handler keeps working with the old secret while you deploy the new one.
- Deploy the new secret within the grace period. After it ends, only the new signature is sent.
- Choose a grace period of 0 if the old secret may have leaked — it stops working at once.

Retries and timeouts

A delivery that does not get a 2xx within 10 seconds is retried with increasing delays, up to 8 attempts in total.

ATTEMPT	WHEN	WAIT
1	Right after the event	—
2	10 s after attempt 1	10 s
3	30 s after attempt 2	30 s
4	2 min after attempt 3	2 min
5	10 min after attempt 4	10 min
6	30 min after attempt 5	30 min
7	1 h after attempt 6	1 h
8	3 h after attempt 7	3 h

- The first attempt is made right after the event happens.
- Delays grow along the schedule above and are capped at 6 hours.
- After 8 attempts — or 3 attempts if your endpoint answers with a permanent 4xx (anything 400–499 except 408 and 429) — the delivery is marked dead.
- Delivery history is kept for 7 days.
- The event body never changes between attempts; only the signature timestamp does.

Handler checklist

- Verify the signature on the raw body and reject old timestamps.
- Return 2xx quickly; queue heavy work.
- Deduplicate on x-lacspace-delivery, and treat status updates as idempotent.
- Return 5xx (or time out) for temporary problems so we retry; return 2xx for events you choose to ignore.

GUIDES

11. Numbers & countries

Send numbers in E.164 and let the API tell you where you can deliver.

Number formats

E.164 is recommended: a +, the country code and the subscriber number, with no spaces, for example +9779801234567. The national format of your workspace's country is accepted too, and the country code without + works as well.

YOU SEND	READ AS	NOTES
+9779801234567	+9779801234567	E.164 — recommended.
9779801234567	+9779801234567	Country code without +.
9801234567	+9779801234567	National format of the workspace's country.
014XXXXXX (landline)	—	400 invalid_number: can't receive SMS.

Every number must match `^\+?\d{7,15}$`. SMSRay stores recipients in E.164, so to in message lookups and webhooks is always `+<country code><number>`.

Where you can send

Your workspace has a country, set when it is created. You can send to that country once SMSRay delivers there. Today **Nepal is live**; **India** and **Bangladesh** are coming soon. Ask the API for the current list with [GET /sms/coverage](#) instead of hard-coding it.

Number errors

HTTP	CODE	WHEN
400	invalid_request	to is missing, shorter than 7 digits or not digits (zod details).
400	invalid_number	The number can't receive SMS — for example a landline, or a number that fails its country's mobile rules.
422	destination_not_supported	The number is valid, but in a country your workspace can't send to yet. details is { country, allowed }.

Show 422 differently from 400. The number isn't wrong — we just don't deliver there yet:

422 application/json

```
{
  "error": "We don't deliver to India yet. This workspace can send to: Nepal.",
  "code": "destination_not_supported",
  "details": { "country": "IN", "allowed": ["NP"] }
}
```

Node.js

```
const res = await fetch("https://api.smsray.com/api/sms/v1/sms/send", { method: "POST", headers,
»   body });
if (res.status === 422) {
  const { code, details } = await res.json();
  if (code === "destination_not_supported") {
    const name = new Intl.DisplayNames(["en"], { type: "region" }).of(details.country);
    return showError(`We don't deliver to ${name} yet.`); // details.allowed lists where you can
»   send
  }
}
```

Also in Python, PHP, Go: smsray.com/docs/numbers

GUIDES

12. Errors and codes

Errors use conventional HTTP status codes and one JSON shape, with a stable machine-readable code you can branch on.

Error shape

Every error response has a JSON body with a human-readable **error** and a stable **code**. Some errors add **details**: a list of field problems on validation errors, or an object such as { **country**, **allowed** }.

400 application/json

```
{
  "error": "Invalid request",
  "code": "invalid_request",
  "details": [
    { "path": ["to"], "message": "Required", "code": "invalid_type" }
  ]
}
```

422 application/json

```
{
  "error": "We don't deliver to India yet. This workspace can send to: Nepal.",
  "code": "destination_not_supported",
  "details": { "country": "IN", "allowed": ["NP"] }
}
```

The error text may change. Write your logic against code and the HTTP status.

Error codes

HTTP	CODE	WHEN
400	invalid_request	Validation failed (<code>details</code> lists each problem), bad JSON, both or neither of <code>text</code> and <code>templateId</code> , or text over 6 segments.
400	invalid_number	The number can't receive SMS — for example a landline.
400	invalid_variables	A template variable is missing, unknown, empty, over 100 characters or contains a line break.
401	unauthorized	<code>x-api-key</code> missing or invalid.
402	insufficient_balance	Workspace balance is lower than the message cost. Top up at your plan rate.
403	client_disabled	The API client that owns the key is disabled.
403	workspace_pending	Your workspace has not been activated yet.
403	workspace_suspended	POST while the workspace is suspended.
403	forbidden	The message <code>type</code> is not allowed on your plan or workspace (for example <code>promotional</code> outside Enterprise).
403	template_required	The text doesn't match an approved template of the same type, or the <code>templateId</code> isn't an approved template of this workspace.
404	not_found	Unknown id, or a message that belongs to another API client.
409	idempotency_conflict	Idempotency-Key reused with a different body, or the first request is still running.
422	destination_not_supported	Valid number, but in a country your workspace can't send to yet. <code>details: { country, allowed }</code> . Show "We don't deliver to <country> yet", not "invalid number".
429	rate_limited	A rate or OTP limit was hit. Wait for <code>Retry-After</code> seconds.
500	server_error	Unexpected error on our side. Retry with backoff and the same Idempotency-Key.
503	unavailable	Delivery is temporarily unavailable. Nothing was charged; retry with backoff.
503	sms_disabled	The SMS service is temporarily switched off. Retry later.

Key restriction codes

If an account owner sets restrictions on an API client in the dashboard, requests outside them get one of these:

HTTP	CODE	WHEN
403	type_not_allowed	The message type is outside the types allowed for this key.

HTTP	CODE	WHEN
403	ip_not_allowed	The call came from an IP outside the key's allowlist.
403	prefix_not_allowed	The number's prefix is outside the key's allowed prefixes.
429	daily_cap	The key reached its daily message cap.
429	number_cap	Too many messages to one number in an hour.
429	suspected_pumping	OTP traffic on this key looks like SMS pumping and the guard is set to block.

Answers that are not errors

A wrong OTP is a normal answer from `POST /sms/otp/verify`, returned with HTTP 200:

200 application/json

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

REASON	MEANING
invalid_code	The code does not match. The attempt is counted.
too_many_attempts	5 wrong attempts were made. Send a new code.
no_active_otp	No code was sent for this number and purpose, it expired, or it was already used.

Likewise, a message that ends as `undelivered` or `failed` is reported through its status, with a neutral error code such as `delivery_timeout`, not as an API error.

Handling errors

STATUS	RETRY?	WHAT TO DO
400, 403, 404	No	Fix the request, the key, the template or the workspace setup.
401	No	Check the key; it may have been rotated.
402	After top-up	Add balance, then retry with the same Idempotency-Key.
409	Yes, carefully	If the first request is still running, wait 1 s and retry. Otherwise use a new key for a new message.
422	No	Tell the user you don't deliver to that country yet; <code>details.allowed</code> lists where you can.
429	Yes	Wait <code>Retry-After</code> seconds.
500, 503	Yes	Exponential backoff with jitter, same Idempotency-Key.

GUIDES

13. Rate limits

Limits keep delivery healthy for everyone and stop OTP abuse. Every limit answers 429 `rate_limited`, with `Retry-After` whenever it is known.

Per-client rate

Each API client may make up to **20 requests per second** by default, across all v1 endpoints. The limit is per client, so separate apps on separate clients don't slow each other down. If you need more, talk to us.

SENDING SPEED

The request rate is how fast we accept messages. Delivery is paced by our platform to protect deliverability, so a large burst is accepted at once and delivered over the following minutes.

OTP limits

`POST /sms/otp/send` has extra limits that protect your users and your balance from SMS pumping and brute force:

LIMIT	SCOPE	VALUE
Resend cooldown	Your client + number + purpose	1 code per 60 s
Per number	The phone number, across every SMSRay customer	3 codes per 10 min
Per end user IP	Your client + the <code>ip</code> you pass	10 codes per 10 min
Per calling IP	Your client + your server's IP	60 requests per minute
Verify attempts	Each code	5 wrong tries, then locked
Code lifetime	Each code	5 minutes

The per-IP limit applies only when you pass `ip`. Pass your end user's IP, not your server's.

The 429 response

Every limit answers the same way. `Retry-After` (in seconds) is set whenever the wait is known, including the OTP resend cooldown.

429 response

```
HTTP/1.1 429 Too Many Requests
Retry-After: 42
content-type: application/json
```

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

Retrying well

- Honour Retry-After when present; otherwise back off exponentially with jitter.
- For OTP, show the remaining cooldown in your UI instead of letting users hammer the resend button.
- Reuse the same Idempotency-Key when retrying a send so a retry can never double-send. See [Idempotency](#).
- Smooth large batches on your side to stay under your per-second rate.
- Optional per-key caps (daily cap, per-number hourly cap) answer 429 daily_cap or number_cap. See [Errors](#).

Webhook retries

Limits run the other way too: when your webhook endpoint fails or is slow, SMSRay retries up to 8 times over about 4.5 hours (10 s, 30 s, 2 min, 10 min, 30 min, 1 h, 3 h), and stops after 3 attempts on a permanent 4xx. See [Retries and timeouts](#).

GUIDES

14. Idempotency

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

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
```

Also in Node.js, Python, PHP, Go: smsray.com/docs/idempotency

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.

GUIDES

15. Segments & encoding

You are billed per segment. The encoding your text needs decides how many characters fit in each one — try it below.

Segment counter

Type a message to see how SMSRay will encode and split it. This uses the same rules as the API.

INTERACTIVE COUNTER

The live segment counter is on the web page: smsray.com/docs/encoding.

GSM-7

If every character is in the GSM 03.38 basic or extension table, the message uses GSM-7: **160 characters** fit in one segment. That covers English letters, digits, common punctuation and a few accented letters.

Extension characters cost two septets each, because they are sent as an escape plus the character:

`^ { } \ [ ] ~ | €` form feed

UCS-2 and Nepali

A single character outside GSM-7 switches the whole message to UCS-2, where **70 UTF-16 code units** fit in one segment. Devanagari is never GSM-7, so **Nepali text is always UCS-2**. Emoji and other characters outside

the Basic Multilingual Plane cost two code units each.

LOOK OUT FOR

Curly quotes (“ ” ‘ ’), en and em dashes, and non-breaking spaces pasted from documents silently switch a message to UCS-2 and can double its cost.

Long messages

Longer messages are split into parts that the phone joins back together. Each part loses a few characters to the joining header:

ENCODING	SINGLE SEGMENT	PER SEGMENT WHEN SPLIT	6 SEGMENTS (MAX)
GSM-7	160	153	918 septets
UCS-2	70	67	402 code units

An escape pair (GSM-7) or a surrogate pair (UCS-2) is never split across two parts, so a part can end one unit short.

Limits and billing

- A message may use at most **6 segments**. Longer text returns 400 `invalid_request`.
- Empty text is 0 segments and is rejected.
- Cost is your rate for the message `type` × segments, reserved at accept. Your rate is your plan's per-SMS price; see [pricing](#). The reply tells you `segments`, `encoding` and `cost`.
- OTP messages are always one GSM-7 segment.

Keeping costs down

- Keep transactional messages under 160 GSM-7 characters where you can.
- Replace typographic quotes and dashes with plain ASCII before sending English text.
- For Nepali messages, aim for 70 characters or fewer; every 67 more adds a segment.
- Avoid emoji in transactional and OTP messages.

RESOURCES

16. Changelog

What changed in the API and the docs, and when.

Changes to the SMSRay API and developer docs, newest first. Only shipped changes are listed.

10 October 2026

Templates library

- The dashboard has a templates library with **5 prebuilt templates** to start from. Pick one, adjust the wording and submit it for approval.
- Send from an approved template with `templateId` and variables on [POST /sms/send](#); free text must match an approved template, otherwise 403 `template_required`.

Coverage and country-neutral numbers

- New endpoint [GET /sms/coverage](#) returns { `country`, `allowed`, `live`, `comingSoon` } for the calling key.
- Numbers are handled in **E.164**. National format is still accepted for your workspace's country; to in lookups and webhooks is now always E.164.
- New error **422 `destination\_not\_supported`** with details: { `country`, `allowed` } for valid numbers in countries your workspace can't send to yet, distinct from 400 `invalid_number`.

smsray.com

- The API base is now <https://api.smsray.com/api/sms/v1> and the docs live at smsray.com/docs.

Developer docs

- Go samples alongside cURL, Node.js, Python and PHP.
- [OpenAPI 3.1](#) and a [Postman collection](#), generated from the docs.
- Every docs page has a Print / PDF version.

RESOURCES

17. Support

Real people, by email. Tell us the messageId and the error code and we can usually answer in one reply.

Contacts

FOR	CONTACT
Integration help and API questions	support@lacspace.com
Plans, top-ups and higher limits	sales@lacspace.com or talk to sales
Security reports	security@lacspace.com — see Security
Abuse and spam reports	abuse@lacspace.com
Privacy and data requests	privacy@lacspace.com

What to include

- The messageId (or the time and recipient) of an affected message.
- The HTTP status, the code from the error body and the time of the request, in UTC.
- Your API key's prefix (for example ls_live_3f9c) — **never the full key**.
- For webhooks, the x-lacspace-delivery id.

Self-serve

- [Your keys and live examples](#) in the portal.
- [Errors](#) lists every code and what to do about it.
- [Changelog](#) shows what changed and when.