



GET STARTED · API V1

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.

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

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" }'
```

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)
```

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" }'
```

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
}
```

```
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." }'
```

Node.js

```
const res = await fetch("https://api.smsray.com/api/sms/v1/sms/send", {
  method: "POST",
  headers: {
    "x-api-key": process.env.SMSRAY_API_KEY,
    "content-type": "application/json",
    "Idempotency-Key": "order-1042-shipped",
  },
  body: JSON.stringify({ to: "+9779801234567", text: "Your order #1042 has shipped." }),
});
const { messageId, status, segments } = await res.json();
// status is "queued"; track it with GET /sms/messages/:id or a webhook
```

Python

```
import os, requests

r = requests.post(
    "https://api.smsray.com/api/sms/v1/sms/send",
    headers={
        "x-api-key": os.environ["SMSRAY_API_KEY"],
        "Idempotency-Key": "order-1042-shipped",
    },
    json={"to": "+9779801234567", "text": "Your order #1042 has shipped."},
    timeout=10,
)
print(r.json()["messageId"])
```

PHP

```
$ch = curl_init("https://api.smsray.com/api/sms/v1/sms/send");
curl_setopt_array($ch, [
    CURLOPT_POST => true,
    CURLOPT_RETURNTRANSFER => true,
    CURLOPT_HTTPHEADER => [
        "x-api-key: " . getenv("SMSRAY_API_KEY"),
        "content-type: application/json",
    ],
]);
```

```
"Idempotency-Key: order-1042-shipped",
],
CURLOPT_POSTFIELDS => json_encode([
    "to" => "+9779801234567",
    "text" => "Your order #1042 has shipped.",
]),
]);
$res = json_decode(curl_exec($ch), true);
echo $res["messageId"];
```

Go

```
package main

import (
    "bytes"
    "encoding/json"
    "fmt"
    "net/http"
    "os"
)

func main() {
    body, _ := json.Marshal(map[string]string{
        "to": "+9779801234567",
        "text": "Your order #1042 has shipped.",
    })
    req, _ := http.NewRequest("POST", "https://api.smsray.com/api/sms/v1/sms/send",
    » bytes.NewReader(body))
    req.Header.Set("x-api-key", os.Getenv("SMSRAY_API_KEY"))
    req.Header.Set("content-type", "application/json")
    req.Header.Set("Idempotency-Key", "order-1042-shipped")

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

    var out struct {
        MessageID string `json:"messageId"`
        Status     string `json:"status"`
    }
    json.NewDecoder(res.Body).Decode(&out)
    fmt.Println(out.MessageID, out.Status) // status is "queued"
}
```

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"
```

Node.js

```
const res = await fetch(`https://api.smsray.com/api/sms/v1/sms/messages/${messageId}`, {
  headers: { "x-api-key": process.env.SMSRAY_API_KEY },
});
const { status } = await res.json(); // "queued" -> "sent" -> "delivered"
```

Python

```
r = requests.get(f"https://api.smsray.com/api/sms/v1/sms/messages/{message_id}",
  headers={"x-api-key": os.environ["SMSRAY_API_KEY"]}, timeout=10)
print(r.json()["status"])
```

PHP

```
$ch = curl_init("https://api.smsray.com/api/sms/v1/sms/messages/" . $messageId);
curl_setopt_array($ch, [
  CURLOPT_RETURNTRANSFER => true,
  CURLOPT_HTTPHEADER => ["x-api-key: " . getenv("SMSRAY_API_KEY")],
]);
$res = json_decode(curl_exec($ch), true);
echo $res["status"]; // "queued" -> "sent" -> "delivered"
```

Go

```
// imports: encoding/json, fmt, log, net/http, os
req, _ := http.NewRequest("GET", "https://api.smsray.com/api/sms/v1/sms/messages/" + messageID,
» nil)
req.Header.Set("x-api-key", os.Getenv("SMSRAY_API_KEY"))

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

var out struct {
  Status string `json:"status"`
}
json.NewDecoder(res.Body).Decode(&out)
fmt.Println(out.Status) // "queued" -> "sent" -> "delivered"
```

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

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

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

SMSRAY
DEVELOPER DOCS