

SMS OTP

Integration Guide

For agents and business systems integrating with SetupSMS from their own backend servers. This release provides two endpoints: send an SMS verification code and verify the submitted code.

01 Connection details

Item	Value
API base URL	https://setupsms.store/otp-gateway
Send a code	POST /v1/messages
Verify a code	POST /v1/verifications
Authentication	Authorization: Bearer <API_KEY>
Request format	Content-Type: application/json; UTF-8
Recommended client ID	User-Agent: EngageHub-Client/1.0
Current SMS template	ID "1"; 6 numeric digits; valid for 10 minutes
Client timeout	Allow 45 seconds; follow the retry and idempotency rules below

The platform supplies a separate API key for each agent. The key is bound to that agent, its approved templates and its sending limits. Store the key on your backend server. Browser and mobile applications should call your own backend.

02 Integration flow

Send a request from your backend → save message_id → the user receives and enters the code → your backend submits a verification request → authorize the business action only when verified is strictly true.

Bind message_id to the intended phone number, business user and operation in your own system. Treat message IDs as strings.

03 Release scope

SetupSMS authenticates requests, checks agent permissions and forwards accepted requests to the official EngageLab OTP API. Recipients are encrypted in the gateway records; submitted OTP values are not stored in those records.

This agent API does not expose template creation, message-history queries, delivery callbacks or administrator sessions. Administrator access is supplied separately. Prepaid agent-wallet charging is not enabled; commercial rates are agreed separately.

Send a verification code

POST <https://setupsms.store/otp-gateway/v1/messages>

Authorization: Bearer <API_KEY>

Content-Type: application/json

User-Agent: EngageHub-Client/1.0

Idempotency-Key: <UNIQUE_REQUEST_ID>

Field	Type / required	Description
to	string / yes	International phone number, including + and country code. Do not include spaces or hyphens.
template.id	string / yes	The assigned approved template. Current template ID: "1".
template.language	string / no	Omit to use the default. Supported values: default, zh_CN, zh_HK, en, ja, th, es.
template.params	object / no	Only supply variables required by the template. String values only; at most 32 entries. Keys: 1-64 characters. Values: at most 1,024 characters.
channel	string / no	For the current SMS template, omit this field or use "sms".
end_user_ip	string / no	The real end-user IPv4 or IPv6 address for upstream risk checks. Do not blindly trust an address supplied by the end user.

Example request and successful response

```
{"to":"+6591234567","template":{"id":"1"}}
```

HTTP 200

```
{"message_id":"1725407449772531712","send_channel":"sms"}
```

The phone number and message ID above are format examples. HTTP 200 means the sending request was accepted, not that the recipient received the SMS. This release has no agent-facing final-delivery query.

Idempotency-Key is required

Generate a unique request ID, such as a UUID, for each new sending intent. Allowed characters: letters, digits, underscore, period, colon and hyphen; maximum 128 characters. Repeating the same key and request body for the same agent returns the saved result. A different body with the same key returns HTTP 409.

A concurrent request still in progress can return HTTP 202 with `{"message_id":null,"state":"pending"}`. Wait and replay the original key and identical request body. Do not generate a new key to retry a potentially accepted send.

A timeout, connection loss or HTTP 5xx can occur after upstream acceptance. The gateway does not automatically resend. Retrieve the saved outcome with the original key. If it remains pending or unknown, contact the platform before starting another send.

Verify a submitted code

POST <https://setupsms.store/otp-gateway/v1/verifications>
Authorization: Bearer <API_KEY>
Content-Type: application/json
User-Agent: EngageHub-Client/1.0

Field	Type / required	Description
message_id	string / yes	The message ID returned by a send made by the same agent.
verify_code	string / yes	The code entered by the user. The current template uses 6 digits. Keep any leading zeros.

Example request

```
{
  "message_id": "1725407449772531712",
  "verify_code": "123456"
}
```

Successful verification

HTTP 200
{ "message_id": "1725407449772531712", "verified": true }

Check both the HTTP result and verified === true. HTTP 200 alone does not mean verification succeeded. If verified is false, do not authorize the business action. The gateway does not echo verify_code in its response.

Expiry and single use

Current template "1" expires after 10 minutes. After a successful verification, the same OTP cannot be verified again. Mark your own challenge as completed and prevent repeated business authorization.

Expired, already verified or missing messages can return errors. Start a new OTP flow when appropriate and use a new Idempotency-Key for a genuinely new send.

Ownership and historical records

An agent can verify only messages sent by that agent through this gateway. A missing message or another agent's message returns HTTP 404 / message_not_found. Knowing another message ID does not grant access.

Messages from older applications, and imported provider history without a corresponding agent send, are historical records. They do not grant permission to verify through this API.

Application responsibilities

Limit verification attempts and resend frequency in your own application. Only verified=true should complete sign-in, phone binding or password recovery. Do not log OTP values or API keys.

Errors, limits and troubleshooting

Gateway validation errors generally use {"error":"identifier"}. Upstream errors may use {"code":number,"message":"description"}. Keep the HTTP status, error/code and X-Request-ID for support.

HTTP	Common reason	Recommended handling
400	Invalid recipient, template, code or request; upstream rejection	Inspect error/code. Do not repeatedly send.
401 / 403	Missing or invalid key; disabled agent; IP/country restriction; insufficient permission	Check credentials and agent permissions with the platform.
404	message_not_found or an unsupported route	Check message ownership and the URL.
409	idempotency_conflict or unsupported template strategy	Keep the original request body; contact the platform if needed.
413 / 415	Body too large or incorrect Content-Type	Use JSON and keep the API request body within 16 KiB.
429	IP, agent, recipient or upstream rate limit	Stop rapid retries. Delay or contact the platform.
502 / 503 / other 5xx	Upstream failure, unavailable configuration or uncertain outcome	Keep the original request ID. Follow the send-retry rules.

Current gateway limits

The default integration agent has a daily limit of 100 accepted or potentially accepted sends, counted by UTC day. The global recipient limit is 5 sends per hour. Edge limits are approximately 2 requests/second per source IP and 5 concurrent connections. Agent permissions and upstream policy may impose stricter limits.

Both POST endpoints support standard server clients without browser-integrity checks. User-Agent: EngageHub-Client/1.0 is recommended for identification. Other edge protections still apply. A 403 without X-Request-ID may originate at Cloudflare. Check the response status and Content-Type before parsing JSON. API responses are not cached.

Common gateway error identifiers

idempotency_key_required — missing send request ID.
idempotency_conflict — the original request body changed.
daily_limit_reached — the agent has exhausted its daily allowance.
destination_rate_limited — the recipient is being contacted too frequently.
upstream_unavailable — upstream outcome may be uncertain.

When contacting support

Provide the request time and timezone, endpoint, HTTP status, X-Request-ID, Idempotency-Key, message_id if available, and a redacted error response. Do not include your API key or OTP.

Quick start and acceptance checks

Run these examples on your backend server. Set `API_KEY` and `OTP_PHONE` securely. Use a number you control whose recipient has agreed to receive the test. The examples below are documentation; they do not send automatically.

1. Send: create a fresh `REQUEST_ID` for each new intent

```
curl --max-time 45 -sS -X POST \
  "https://setupsms.store/otp-gateway/v1/messages" \
  -H "Authorization: Bearer ${API_KEY}" \
  -H "User-Agent: EngageHub-Client/1.0" \
  -H "Content-Type: application/json" \
  -H "Idempotency-Key: ${REQUEST_ID}" \
  --data "{\"to\":\"${OTP_PHONE}\",\"template\":{\"id\":\"1\"}}"
```

Save the successful response `message_id` as `MESSAGE_ID`. On an error, keep `REQUEST_ID`; do not silently create a new send request.

2. Verify: submit the code actually received

```
curl --max-time 45 -sS -X POST \
  "https://setupsms.store/otp-gateway/v1/verifications" \
  -H "Authorization: Bearer ${API_KEY}" \
  -H "User-Agent: EngageHub-Client/1.0" \
  -H "Content-Type: application/json" \
  --data "{\"message_id\":\"${MESSAGE_ID}\",\"verify_code\":\"${OTP_CODE}\"}"
```

Before going live

1. Obtain the agent's own API key and confirm template permissions and quota.
2. Send one approved test to your controlled number and confirm receipt.
3. Verify the received code and confirm `verified=true`.
4. Prevent repeated business authorization after success.
5. Handle idempotency conflicts, incorrect codes, timeouts and rate limits.

Credentials are supplied separately. Delivery depends on destination permissions, template approval and carrier requirements. Validate your intended destinations with the platform.

Official upstream references

[EngageLab: Send OTP API](#)

[EngageLab: Verify OTP API](#)

This guide defines the public SetupSMS interface. The platform manages upstream authentication and gateway records. Use the endpoint paths and agent permissions supplied by SetupSMS.