RESELLER DEVELOPER DOCUMENTATION

SMS API

Integration guide and examples

Connect your own systems to the reseller.co.tz SMS API: send single messages from your prepaid SMS wallet, read your send history, and track delivery status, all from a secure backend.

Audience	Reseller partners and their developers or system integrators
Product	SMS API v1 (REST, JSON over HTTPS)
Base URL	https://reseller.co.tz/api/v1/sms
Live reference	https://reseller.co.tz/api-docs
Version	1.0
Last updated	26 October 2026

Scope of this guide

Covers sending single SMS messages, reading your send history and reading delivery reports. Campaigns, contact groups, sender-ID requests and wallet top-ups are managed in the reseller portal. **The SMS API uses its own bearer token, which is separate from your Domains API key.**

Contents

No.	Section	What it covers
1	About this guide	Who it is for and what it covers
2	API at a glance	Capabilities, base URL, integration boundary
3	Before you integrate	SMS wallet, bearer token, sender IDs, IP whitelist
4	Request and response conventions	Headers, envelope, credits, errors, retries
5	Data reference	Send result, log entry, delivery report, status values
6	Integration examples	Quick start, cURL, Laravel, Node.js, plain PHP
7	Endpoint reference	Send, logs and delivery reports with full examples
8	Workflow guidance	Sending safely, tracking, reconciliation
9	Testing and go-live	Test cases and production launch checklist
10	Troubleshooting and support	Common errors and what to send to support
11	Document information	Versioning, maintenance and sharing

Tip: the live reference at reseller.co.tz/api-docs is updated with every API release and is the authority for parameter lists. This guide adds the integration guidance, end-to-end examples and operational rules your team needs to plan, build and launch.

1 About this guide

Use this guide to connect a reseller system to the reseller.co.tz SMS API through a secure backend service. Messages are charged to your prepaid SMS wallet in credits, where one credit equals one billable message segment.

1.1 What you can build

- **Transactional alerts** from your website, ERP, school system or billing platform.
- **Order and delivery notifications** such as payment confirmations and pickup reminders.
- **Customer portals** that send a message when a customer takes an action.
- **Operational alerts** for your own team, for example failed payment or low stock warnings.
- **Reporting dashboards** that read your send history and delivery status.

1.2 What this guide covers

Sending a single SMS, reading your send log, and reading delivery reports. All three endpoints are authenticated with the SMS bearer token and restricted to your own account.

Not covered by the API: bulk campaigns, contact groups, sender-ID requests and wallet top-ups. Those are managed in the reseller portal under **SMS Reseller**.

1.3 How to use it

1. Read sections 2 to 5 to understand access, credits and the data you will store.
2. Send your first test message using section 6.1 and confirm it arrives.
3. Use section 7 as the endpoint reference while you build each feature.
4. Follow section 8 for the reconciliation rules that keep your credits and records accurate.
5. Complete the launch checklist in section 9 before sending to real customers.

2 API at a glance

2.1 Capabilities

Area	What your system can do
Send	Send one SMS to a single recipient through an approved sender ID and deduct credits from your wallet.
Send log	Read your own attempt log with filters by status, source and purpose, with pagination.
Delivery reports	Read submit-level delivery status for your attempts, including filtering by provider message ID and date range.

2.2 Base URL and transport

Item	Value
Base URL	https://reseller.co.tz/api/v1/sms
Protocol	HTTPS only. TLS is required on every request.
Request body	JSON (`Content-Type: application/json`) for POST requests.
Response body	JSON. Always send `Accept: application/json` so errors come back as JSON.
Authentication	`Authorization: Bearer <token>` using your SMS API token, plus a whitelisted source IP address.
Billing	Credits are deducted from your prepaid SMS wallet: one credit per billable segment.
Versioning	The path carries the version (`/api/v1/sms`). Breaking changes will appear under a new version.

2.3 Integration boundary

Call the API from a server, never from the browser

Your SMS token belongs to a server-side backend. Do not call the API directly from browser JavaScript or a mobile app bundle, where the token can be read and abused to spend your credits.

Good fit	Do not do this
A backend service with fixed outbound IP addresses	Browser JavaScript holding the SMS token
A queue that retries safely and checks the log	Resending automatically when a request times out
Server-side storage of the token in a secret manager	Storing the token in source control or chat
General, customer-friendly error messages	Showing raw gateway error text to customers

3 Before you integrate

Most first-time failures come from SMS access not being enabled, no approved sender ID, or an unwhitelisted IP address. Complete these checks first.

3.1 Access checklist

Check	Expected state	Where to manage it
SMS reseller access	The SMS Reseller workspace is available when you sign in.	reseller.co.tz
SMS API token	A bearer token has been generated and the API is enabled.	SMS Reseller dashboard
Sender ID	At least one sender ID with status **approved**.	SMS Reseller → Sender IDs
SMS credit	The wallet holds enough credits for your expected volume.	SMS Reseller → Wallet
IP whitelist	Your server's outbound IP is whitelisted on the API Access page.	API Access page

3.2 The bearer token is separate from the Domains API key

The SMS API and the Domains API use different credentials. The SMS token is shown on the **SMS Reseller dashboard** and only works on /api/v1/sms endpoints. A Domains API key (sent as X-API-KEY) is not accepted here, and the SMS token is not accepted by the Domains API.

- Create the token from the SMS Reseller dashboard, where it is shown next to the SMS base URL.
- **Regenerating the token invalidates the previous one immediately.** Every connected system stops working until it is updated.
- You can **disable** the SMS API at any time from the same page without affecting your Domains API key.
- Store the token in an environment variable or secret manager and never log the Authorization header.

3.3 Sender IDs

Every message is sent from one of your approved sender IDs. An unapproved or unknown sender ID is rejected before any message is sent.

- Request a sender ID from **SMS Reseller → Sender IDs**. A fee may apply and approval is required.
- Sender IDs are 3 to 11 characters and may contain letters and digits only. No spaces or symbols.
- Matching is case-insensitive: Shop, SHOP and shop are the same sender ID.
- A sender ID already approved for another reseller cannot be reused.
- Send only with an approved sender ID; exact casing does not matter, but the ID must be approved.

3.4 Wallet credit and message segments

One credit equals one billable message segment, and the cost is calculated from the message text before it is sent.

Message type	Single segment	Each extra segment
Standard characters (GSM-7)	up to 160 characters	153 characters per segment
Special characters, or any non-Latin script (Unicode)	up to 70 characters	67 characters per segment

- The API returns segments and credits_charged so you can reconcile with the wallet statement.
- If the wallet cannot cover the message, nothing is sent and no credit is deducted.
- Top up from **SMS Reseller** → **Wallet**. Credits are prepaid, so a low balance stops sending.
- One accented or non-Latin character, and some symbols, moves the whole message to Unicode and roughly halves the characters per segment. Keep messages short and plain to avoid unexpected multi-segment charges.

3.5 Recipient numbers

Numbers are normalised before sending. Local and international formats are accepted:

You send	Sent as	Notes
0712345678	255712345678	A leading 0 is treated as a Tanzanian local number.
+255 712 345 678	255712345678	Spaces, dashes and brackets are ignored.
255712345678	255712345678	Already in international format.

The number must contain 10 to 15 digits after normalisation, otherwise the request is rejected. Store the normalised number returned in data.recipient.

4 Request and response conventions

4.1 Authentication headers

Header	Value	Required when
Authorization	Bearer <your SMS token>.	Every request.
Accept	application/json.	Every request.
Content-Type	application/json.	Requests with a JSON body (POST).

4.2 Common JSON response

Success

```
{
  "success": true,
  "message": "SMS accepted successfully.",
  "data": {
    "recipient": "255712345678",
    "sender_id": "SHOP",
    "segments": 1,
    "credits_charged": 1,
    "status": "accepted",
    "provider_message_id": "msg_123"
  }
}
```

Error

```
{
  "success": false,
  "message": "The given data was invalid.",
  "errors": {
    "sender_id": ["Use one of your approved sender IDs."]
  }
}
```

Branch on the HTTP status first, then on success. Show customers a short, friendly explanation and never print raw gateway error text.

4.3 Status codes

Code	Meaning	Recommended client behaviour
200	Request accepted. The message (or page of results) was returned.	Read success, message and data.
401	The bearer token is missing, invalid, or the SMS API is disabled for the account.	Stop retries; fix the token or enable the API.
403	The token is valid but the calling IP is not whitelisted.	Check the exact outbound IP and whitelist settings.
404	SMS sending is not available on the platform right now.	Stop sending; contact support if it continues.
422	Validation failed: unapproved sender ID, invalid number, insufficient credit, or the gateway rejected the message.	Show a safe message; fix the input. Do not loop retries.

Code	Meaning	Recommended client behaviour
429	Too many requests.	Back off and retry later.
500	Unexpected server error.	Retry reads cautiously. Check the log before resending a message.

4.4 Credits, retries and duplicate messages

A timeout does not mean the message failed

If the connection drops or the gateway does not confirm acceptance, the message may still have been sent and the credits may remain charged while the outcome is verified. **Always check before resending.**

Action type	Safe retry rule
Reading logs and delivery reports	Safe to retry with a short delay. Never retry 401 or 403.
Sending an SMS	Do not resend automatically. Check GET /sms/logs for the same recipient and message before sending again.
Ambiguous send result	The response says the credits remain charged while the outcome is verified. Wait and check the log; do not resend immediately.
Definite rejection	A rejected message is refunded automatically. Correct the input before trying again.

Example duplicate guard

```
def send_once(recipient, message, sender_id):
    # Look for the same message to the same number in the last few minutes.
    recent = api.get("/sms/logs", params={"per_page": 25})
    for row in recent["data"]["logs"]:
        if row["recipient"] == recipient and message.startswith(row["message_preview"][:20]):
            return row          # already attempted, do not resend
    return api.post("/sms/send", {...})
```

4.5 Rate limits

The API applies a general limit to every request and tighter limits to sending and reads.

Response	What it means	What your client should do
429 Too Many Requests	You exceeded the limit for this window.	Pause, queue the work, and retry later with a backoff.
403 Forbidden	The IP is not whitelisted.	Stop retrying. Fix the whitelist.
401 Unauthorized	The token is wrong, disabled, or regenerated.	Stop retrying. Correct or regenerate the token.

- Queue messages and send at a controlled rate rather than in parallel bursts.
- Use per_page up to 100 when reading logs instead of many small requests.
- Send transactional messages as they happen; batch any bulk work in the portal's campaigns instead of looping the send endpoint.

5 Data reference

5.1 Send result

Returned in data by POST /sms/send.

Field	Type	Meaning
recipient	string	Normalised recipient number, for example 255712345678.
sender_id	string	Approved sender ID used for the message.
segments	integer	Number of billable segments the message was split into.
credits_charged	number	Credits deducted from your wallet (equals the segment count).
status	string	accepted when the gateway accepted the message.
provider_message_id	string or null	Gateway message ID. Store it to track delivery.

5.2 Log entry

Returned in data.logs by GET /sms/logs.

Field	Type	Meaning
id	integer	Your SMS attempt ID.
status	string	accepted or failed.
delivery_status	string	Submit-level status: pending, submitted, delivered, undelivered or failed.
source	string	portal, api or admin.
purpose	string	campaign, test, api, quick, marketing or provider_test.
recipient	string	Normalised recipient number.
sender_id	string	Sender ID used.
message_preview	string	Short preview of the message text.
segments	integer	Billable segments.
credits_charged	number	Credits charged for the attempt.
provider_message_id	string or null	Gateway message ID.
delivered_at	ISO 8601 or null	When delivery was confirmed, if available.
error	string or null	Safe failure reason for failed attempts.
created_at	ISO 8601	When the attempt was created.

Example log entry


```
{
  "id": 1,
  "status": "accepted",
  "delivery_status": "submitted",
  "source": "api",
  "purpose": "api",
  "recipient": "255712345678",
  "sender_id": "SHOP",
  "message_preview": "Your order is ready for collection.",
  "segments": 1,
  "credits_charged": 1,
  "provider_message_id": "msg_123",
  "delivered_at": null,
  "error": null,
  "created_at": "2026-10-02T08:00:00+00:00"
}
```


5.3 Delivery report

Returned in data.delivery_reports by GET /sms/delivery-reports.

Field	Type	Meaning
id	integer	Your SMS attempt ID.
provider_message_id	string or null	Gateway message ID; use it to match a send.
recipient	string	Normalised recipient number.
sender_id	string	Sender ID used.
status	string	accepted or failed.
delivery_status	string	pending, submitted, delivered, undelivered or failed.
segments	integer	Billable segments.
submitted_at	ISO 8601	When the attempt was submitted.
delivered_at	ISO 8601 or null	When delivery was confirmed.
error	string or null	Failure reason for failed attempts.

Example delivery report


```
{
  "id": 1,
  "provider_message_id": "msg_123",
  "recipient": "255712345678",
  "sender_id": "SHOP",
  "status": "accepted",
  "delivery_status": "submitted",
  "segments": 1,
  "submitted_at": "2026-10-02T08:00:00+00:00",
  "delivered_at": null,
  "error": null
}
```


Delivery status is submit-level until the gateway delivery-report feed is connected, so treat submitted as "accepted by the network" rather than "handset confirmed".

6 Integration examples

6.1 Quick start: send your first message

1. Sign in to reseller.co.tz, open **SMS Reseller**, and copy the bearer token and SMS base URL from the dashboard.
2. Confirm you have at least one **approved** sender ID and enough credit in your SMS wallet.
3. Add your server's outbound IP to the whitelist on the **API Access** page.
4. Store the token as RESELLER_SMS_TOKEN in your server environment.
5. Send a test message to your own phone with the request below and confirm it arrives.

```
curl -X POST "https://reseller.co.tz/api/v1/sms/send" \
  -H "Authorization: Bearer $RESELLER_SMS_TOKEN" \
  -H "Content-Type: application/json" \
  -H "Accept: application/json" \
  -d '{"recipient": "0712345678", "message": "This is a test message from our system.", "sender_id": "SHOP"}'
```

6.2 cURL reference

```
curl -X GET "https://reseller.co.tz/api/v1/sms/logs?status=accepted&per_page=10" \
  -H "Authorization: Bearer $RESELLER_SMS_TOKEN" \
  -H "Accept: application/json"
```

6.3 PHP and Laravel backend client

A small reusable client that reads the token from the environment and keeps send failures explicit.

```php
<?php

use Illuminate\Http\Client\ConnectionException;
use Illuminate\Support\Facades\Http;

final class SmsApiClient
{
    private string $base;
    private string $token;

    public function __construct()
    {
        $this->base = rtrim((string) env('RESELLER_SMS_BASE_URL', 'https://reseller.co.tz/api/v1/sms'), '/');
        $this->token = (string) env('RESELLER_SMS_TOKEN');
    }

    private function request(string $method, string $path, array $query = [], array $payload = [])
    {
        return Http::baseUrl($this->base)
            ->withToken($this->token)          // sends Authorization: Bearer <token>
            ->acceptJson()
            ->timeout(30)
            ->send($method, $path, array_filter([
                'query' => $query,
                'json'  => $payload ?: null,
            ]));
    }

    public function send(string $recipient, string $message, string $senderId)
    {
        return $this->request('POST', '/send', [], [
            'recipient' => $recipient,
            'message'   => $message,
            'sender_id' => $senderId,
        ]);
    }

    public function logs(array $filters = [])
    {
        return $this->request('GET', '/logs', $filters);
    }
}
```

Sending with a duplicate guard

```php
$sms = new SmsApiClient();

try {
    $response = $sms->send('0712345678', 'Your order #1042 is ready for pickup.', 'SHOP');
} catch (ConnectionException $e) {
    // Ambiguous: do not resend. Record it and check the log before trying again.
    report($e);
    return;
}

if ($response->successful()) {
    $data = $response->json('data');
    // Store $data['provider_message_id'], $data['recipient'] and $data['credits_charged'].
} elseif ($response->status() === 422) {
    // Validation, sender ID, credit or gateway rejection. Keep errors server-side.
}
```

6.4 Node.js backend request

```
const BASE_URL = process.env.RESELLER_SMS_BASE_URL || 'https://reseller.co.tz/api/v1/sms';
const TOKEN = process.env.RESELLER_SMS_TOKEN;

async function smsCall(method, path, body) {
  const res = await fetch(`${BASE_URL}${path}`, {
    method,
    headers: {
      'Authorization': `Bearer ${TOKEN}`,
      'Accept': 'application/json',
      ...(body ? { 'Content-Type': 'application/json' } : {}),
    },
    body: body ? JSON.stringify(body) : undefined,
    signal: AbortSignal.timeout(30000),
  });

  const json = await res.json().catch(() => ({}));
  return { status: res.status, ok: res.ok, ...json };
}

// Send once. On a timeout, read the log before considering a resend.
async function sendSms(recipient, message, senderId) {
  return smsCall('POST', '/send', { recipient, message, sender_id: senderId });
}

async function getLogs(filters = {}) {
  const qs = new URLSearchParams(filters).toString();
  return smsCall('GET', `/logs${qs ? '?' + qs : ''}`);
}

export { sendSms, getLogs };
```

6.5 Plain PHP without a framework

```
<?php
function sms_api(string $method, string $path, ?array $body = null): array
{
    $ch = curl_init('https://reseller.co.tz/api/v1/sms'.$path);
    curl_setopt_array($ch, [
        CURLOPT_RETURNTRANSFER => true,
        CURLOPT_CUSTOMREQUEST  => $method,
        CURLOPT_TIMEOUT        => 30,
        CURLOPT_HTTPHEADER     => array_filter([
            'Authorization: Bearer '.getenv('RESELLER_SMS_TOKEN'),
            'Accept: application/json',
            $body !== null ? 'Content-Type: application/json' : null,
        ]),
        CURLOPT_POSTFIELDS     => $body !== null ? json_encode($body) : null,
    ]);

    $raw    = curl_exec($ch);
    $status = curl_getinfo($ch, CURLINFO_HTTP_CODE);
    $error  = curl_error($ch);
    curl_close($ch);

    return ['status' => $status, 'error' => $error, 'data' => json_decode((string) $raw, true)];
}
```

6.6 Environment variables

```
RESELLER_SMS_BASE_URL=https://reseller.co.tz/api/v1/sms
RESELLER_SMS_TOKEN=your-sms-bearer-token   # from the SMS Reseller dashboard
```

7 Endpoint reference

Every SMS endpoint in API version 1.0. All three use the SMS bearer token and return only your own account's data.

7.1 Endpoint index

Method	Path	Purpose	Billable
POST	/send	Send one SMS to one recipient	Yes (credits)
GET	/logs	List your SMS attempts	No
GET	/delivery-reports	Read submit-level delivery status	No

POST /send

METHOD **POST** | BILLING **Billable** | SUCCESS **200**

Send one SMS through your prepaid SMS wallet using an approved sender ID.

Body parameters

Field	Requirement
recipient	required string, max 30. A phone number such as 0712345678 or +255 712 345 678
message	required string, max 918 characters. Plain text; HTML tags are stripped
sender_id	required string, max 11. One of your **approved** sender IDs
client_reference	optional string, max 120, for your own tracking

Implementation notes

- The sender ID must be approved; matching is case-insensitive.
- The wallet must cover the segment cost; otherwise nothing is sent and no credit is deducted.
- A definite rejection is refunded automatically. An unconfirmed outcome remains charged while it is verified.
- Not idempotent: check the log before resending after a timeout.

Example request

```
curl -X POST "https://reseller.co.tz/api/v1/sms/send" \
  -H "Authorization: Bearer $RESELLER_SMS_TOKEN" \
  -H "Content-Type: application/json" \
  -H "Accept: application/json" \
  -d '{"recipient": "0712345678", "message": "Your order #1042 is ready for pickup.", "sender_id": "SHOP"}'
```

Example response

```
{
  "success": true,
  "message": "SMS accepted successfully.",
  "data": {
    "recipient": "255712345678",
    "sender_id": "SHOP",
    "segments": 1,
    "credits_charged": 1,
    "status": "accepted",
    "provider_message_id": "msg_123"
  }
}
```

GET /logs

METHOD GET	BILLING No charge	SUCCESS 200

List your own SMS attempts with filters and pagination.

Parameters

Name	Location and requirement
status	query (optional): accepted or failed
source	query (optional): portal, api or admin
purpose	query (optional): campaign, test, api, quick, marketing or provider_test
per_page	query (optional): integer 1-100, default 25

Implementation notes

- Only attempts owned by the authenticated account are returned.
- Attempts are newest first.
- error is included only for failed attempts and is always a safe, generic message.

Example request

```
curl -X GET "https://reseller.co.tz/api/v1/sms/logs?status=accepted&per_page=10" \
  -H "Authorization: Bearer $RESELLER_SMS_TOKEN" \
  -H "Accept: application/json"
```

Example response


```
{
  "success": true,
  "message": "SMS logs loaded successfully.",
  "data": {
    "logs": [
      {
        "id": 1,
        "status": "accepted",
        "delivery_status": "submitted",
        "source": "api",
        "purpose": "api",
        "recipient": "255712345678",
        "sender_id": "SHOP",
        "message_preview": "Your order is ready for collection.",
        "segments": 1,
        "credits_charged": 1,
        "provider_message_id": "msg_123",
        "delivered_at": null,
        "error": null,
        "created_at": "2026-10-02T08:00:00+00:00"
      }
    ],
    "pagination": {
      "current_page": 1,
      "per_page": 25,
      "total": 1,
      "last_page": 1
    }
  }
}
```


GET /delivery-reports

METHOD	BILLING	SUCCESS
GET	**No charge**	**200**

Read delivery reports for your own SMS attempts, including submit-level delivery status.

Parameters

Name	Location and requirement
provider_message_id	query (optional): the ID returned when you sent the message
status	query (optional): accepted or failed
delivery_status	query (optional): pending, submitted, delivered, undelivered or failed
from	query (optional): ISO date/time lower bound
to	query (optional): ISO date/time upper bound, must not be before from
per_page	query (optional): integer 1-100, default 25

Implementation notes

- Only attempts owned by the authenticated account are returned.
- Use provider_message_id to check a single send after an ambiguous response.
- Delivery status is submit-level until the delivery-report feed is connected.

Example request

```
curl -X GET "https://reseller.co.tz/api/v1/sms/delivery-reports?provider_message_id=msg_123" \
  -H "Authorization: Bearer $RESELLER_SMS_TOKEN" \
  -H "Accept: application/json"
```

Example response


```
{
  "success": true,
  "message": "Delivery reports loaded successfully.",
  "data": {
    "delivery_reports": [
      {
        "id": 1,
        "provider_message_id": "msg_123",
        "recipient": "255712345678",
        "sender_id": "SHOP",
        "status": "accepted",
        "delivery_status": "submitted",
        "segments": 1,
        "submitted_at": "2026-10-02T08:00:00+00:00",
        "delivered_at": null,
        "error": null
      }
    ],
    "pagination": {
      "current_page": 1,
      "per_page": 25,
      "total": 1,
      "last_page": 1
    }
  }
}
```

8 Workflow guidance

8.1 Send a transactional message

1. Build the message text and check its length in your own validation, so the segment cost is predictable.
2. Queue the send with a lock on the business event (for example the order ID) so a double click sends once.
3. Send POST /send once and store provider_message_id, recipient and credits_charged.
4. On success, show a short confirmation to the user, not the raw response.
5. On 422, show a safe message and keep the field errors server-side.
6. On a timeout, do **not** resend. Read GET /sms/logs and match the recipient and message.

```
Order event
  -> build message, estimate segments
  -> lock the order, POST /send (once)
    -> 200: store provider_message_id, credits_charged
    -> 422: safe message, fix input (rejected message is refunded)
    -> timeout: check GET /sms/logs before any resend
```

8.2 Keep segment costs predictable

- Keep messages short and plain: up to 160 standard characters fits one credit.
- Avoid accented characters and emoji in transactional texts unless you accept Unicode pricing (70 characters per segment).
- Build messages from fixed templates so the length cannot grow unexpectedly with customer names.
- Show the credit cost to operators when they compose a message in your admin screens.

8.3 Track delivery

1. Store provider_message_id for every accepted send.
2. Poll GET /delivery-reports?provider_message_id=... at a controlled interval, for example every 15 to 30 minutes for recent sends.
3. Update your record when delivery_status becomes delivered, undelivered or failed.
4. Do not mark a message as delivered based on the send response alone: a send success only means the network accepted it.

8.4 Keep your wallet funded

- Watch your SMS wallet balance from the SMS Reseller dashboard and top up before high-volume periods.
- When the balance is too low, sends stop with a clear error instead of going out unpaid.
- Reconcile credits_charged from your sends against the wallet statement periodically.
- If a send is rejected, the refund is automatic; check the log rather than resending immediately.

9 Testing and go-live

9.1 Minimum test cases before release

Test case	Expected result
Missing or wrong bearer token	401 with `success: false`; no retry loop.
Valid token from a non-whitelisted IP	403 with an IP whitelist message.
Unapproved or misspelled sender ID	422 naming the sender ID field.
Invalid recipient number	422 naming the recipient field; nothing sent.
Insufficient credit	422 with a credit message; nothing sent.
Single-segment message	200 with `segments: 1` and `credits_charged: 1`.
Long or Unicode message	`segments` greater than 1 and matching credits.
Duplicate submission (double click)	Only one message is sent thanks to your lock.
Timeout handling	No automatic resend; the log is checked first.
Log filters	`status`, `source`, `purpose` and pagination return the expected rows.
Delivery report lookup	Filtering by `provider_message_id` returns the single send.
429 handling	Client backs off and retries later without losing the message.

9.2 Production launch checklist

Control	Evidence of completion
SMS token stored in the secret manager	Token present; masked value matches the dashboard.
All production egress IPs whitelisted	403 disappears for every instance.
Approved sender IDs confirmed	Sender IDs show **approved** in the portal.
Wallet funded for expected volume	Balance covers your monthly estimate.
Shared client used for every call	One place adds the token and handles errors.
Duplicate guard implemented	Double submissions are blocked in testing.
No automatic resend on timeout	Code path records and checks the log instead.
Customer-safe messages reviewed	No raw gateway text is shown.
Token rotation runbook written	Documented owner and steps.

10 Troubleshooting and support

10.1 Common errors

Symptom	Likely cause	Next check
401 Unauthorized	Token missing, regenerated or disabled.	Copy the token again from the SMS Reseller dashboard.
403 Forbidden	Calling IP is not whitelisted.	Compare the server's outbound IP with the API Access list.
404 on every call	SMS sending is not available on the platform right now.	Contact support; nothing is being charged.
422 sender ID rejected	The sender ID is not approved or does not exist.	Check Sender IDs in the portal; confirm the ID.
422 recipient rejected	The number is not valid after normalisation.	Use 10 to 15 digits, for example 0712345678.
422 credit message	The wallet cannot cover the segments.	Top up from SMS Reseller → Wallet.
Message says credits remain charged	The gateway did not confirm acceptance.	Wait, then check the log or delivery report.
429 Too Many Requests	Too many requests in the window.	Back off, queue the work and retry.

10.2 Information to collect before requesting support

- The UTC timestamp, HTTP method, endpoint path and HTTP status.
- Your internal reference plus the provider_message_id, without the bearer token.
- The recipient number (masked if it belongs to a customer) and the sender ID used.
- The response message and safe error field names, with the token removed.
- Whether the request was sent once or retried, and what the log showed.

10.3 Support channels

Channel	Details
Live API reference	https://reseller.co.tz/api-docs
SMS settings and wallet	SMS Reseller workspace in the reseller portal
Account and integration help	Open a support ticket from your reseller portal.

Protect credentials in support requests

Never attach your SMS token or full customer phone lists to a ticket or an email. Support can investigate with the timestamp, the status code and the provider message ID.

11 Document information

Item	Value
Document	SMS API integration guide and examples
Version	1.0
API version	1.0 (/api/v1/sms)
Last updated	26 October 2026
Owner	reseller.co.tz
Live reference	https://reseller.co.tz/api-docs

11.1 Maintenance rule

This guide must stay aligned with the live API documentation. When an endpoint or parameter changes, update the source API description first, then revise this guide in the same release. Change the version number and the last-updated date on every published revision.

11.2 Share notice

This guide is intended for reseller partners and their authorised technical teams. It contains public API instructions only: no gateway account names, credentials, internal endpoints or customer data.

11.3 Version history

Version	Date	Change
1.0	26 October 2026	First reseller release: sending, send log and delivery reports.