ZNotice — Customer API v1 Reference

Base URL: `https://znotice.consultbofa.com`

The v1 API is a JSON REST interface for sending WhatsApp messages and reading account/message state. All endpoints are under `/api/v1` and are authenticated with an API key. The API is CSRF-exempt (it is not a browser surface).

> **Status:** v1 preview. Live endpoints: `GET /status`, `GET /messages`, `GET /usage`, `POST /send` (immediate or scheduled), `POST /bulk`, and the scheduled-queue and batch management endpoints (`GET`/`DELETE /scheduled`, `GET`/`DELETE /batches`).

> **Message templates are dashboard-only.** Saved message templates (**Dashboard → Templates**, per-user cap `templates_max_per_user`) only prefill the dashboard WhatsApp/scheduled forms. There are **no** template endpoints in API v1, and API v1 behaviour is unchanged by templates. `{{name}}` / `{{phone}}` placeholders in a prefilled message are resolved at send time by the same scheduler/bulk pipeline documented below.

1. Authentication

Create an API key in **Dashboard → API Keys**. The full key is shown exactly once at creation and is stored only as a SHA-256 hash. Send it on every request using **either** header:

```
Authorization: Bearer zn_live_xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
```

```
X-API-Key: zn_live_xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
```

Key format: `zn_live_` followed by 32 lowercase hex characters. Keys can be revoked at any time from the dashboard; revoked keys immediately return `401 revoked_api_key`.

2. Endpoints

`GET /api/v1/status`

Returns account, credit balance, WhatsApp connection state and the authenticated key.

```
curl https://znotice.consultbofa.com/api/v1/status \
  -H "Authorization: Bearer $ZNOTICE_API_KEY"
```

```
{
  "ok": true,
  "api_version": "v1",
  "user": { "id": 1, "name": "Jane", "email": "jane@example.com" },
  "credits": 1000,
```

```
  "whatsapp": { "session": "zn_1", "status": "disconnected", "phone_number": null },
  "api_keys_active": 1,
  "key": { "label": "Default", "prefix": "zn_live_abcd" }
}
```

`whatsapp.status` is one of `disconnected`, `pending`, `connected`, `error`. `whatsapp` is always present; `session` / `phone_number` are `null` before a session row exists.

`GET /api/v1/messages`

Returns recent messages for the authenticated user, newest first.

Query param	Type	Default	Notes
`limit`	int	`50`	Clamped to `1` – `200`.

```
curl "https://znotice.consultbofa.com/api/v1/messages?limit=20" \
  -H "X-API-Key: $ZNOTICE_API_KEY"
```

```
{
  "ok": true,
  "count": 2,
  "data": [
    {
      "id": 42,
      "to_number": "250788123456",
      "body": "Hello from ZNotice",
      "media_url": null,
      "status": "sent",
      "wpp_message_id": "true_250788123456@c.us_3EB0...",
      "credits_charged": 1,
      "created_at": "2026-09-10T12:00:00.000Z"
    }
  ]
}
```

`GET /api/v1/usage`

Returns per-day message activity and delivery totals for the authenticated user, plus a roll-up, over the last `days` UTC calendar days. **Read-only** — the endpoint never writes to the database.

Query param	Type	Default	Notes
`days`	int	`30`	Window length. Clamped to `1` – `90`; a missing, non-numeric or out-of-range value is clamped, never an error (`0` → 1 day, `500` → 90 days).

```
curl "https://znotice.consultbofa.com/api/v1/usage?days=3" \
  -H "Authorization: Bearer $ZNOTICE_API_KEY"
```

```
{
  "ok": true,
  "days": 3,
  "data": [
```

```
    { "date": "2026-09-09", "sent": 10, "delivered": 5, "read": 3, "failed": 1, "total": 19,
"credits_spent": 18 },
    { "date": "2026-09-10", "sent": 0, "delivered": 0, "read": 0, "failed": 0, "total": 0,
"credits_spent": 0 },
    { "date": "2026-09-11", "sent": 4, "delivered": 2, "read": 1, "failed": 0, "total": 7,
"credits_spent": 7 }
  ],
  "totals": {
    "sent": 14,
    "delivered": 7,
    "read": 4,
    "failed": 1,
    "total": 26,
    "credits_spent": 25,
    "success_rate": 0.9615
  }
}
```

Semantics:

- **UTC day buckets.** `data` always contains exactly `days` points, oldest first, ending on today. The window starts at `UTC_DATE() - INTERVAL (days-1) DAY` **inclusive**, so it includes today. Days with no activity are zero-filled.
- **Completed statuses only.** `total` is `sent + delivered + read + failed`. `queued` messages are excluded from the counts. `failed` counts messages currently in status `failed`.
- **`success_rate`** is `(sent + delivered + read) / total`, a `0`–`1` ratio (`0` when `total` is `0`).
- **`credits_spent`** is `SUM(credits_charged)` over completed statuses only, so a `queued` message never contributes.
- `data[].date` is a `YYYY-MM-DD` UTC calendar date.

`days` is clamped at both the route and inside the analytics service. The global 120 requests/minute/IP limit applies as for every `/api/v1` endpoint.

`POST /api/v1/send`

Sends a WhatsApp text or media message. Costs **1 credit** per message and requires the account's WhatsApp session to be `connected`.

Request body (JSON):

Field	Type	Required	Notes
`phone`	string	yes	Recipient number. See §3 for normalization.
`message`	string	yes*	1–4096 characters. May be omitted if `media_url` is set.
`media_url`	string	no	Public `http(s)` URL (ports 80/443 only). ZNotice fetches the file server-side and `message` becomes the caption. See §2.1 for limits and the egress guard. `mediaUrl` is also accepted.
`scheduled_at`	string	no	Defer delivery to the scheduler queue. ISO-8601 with `Z` /offset, or naive `YYYY-MM-DDTHH:MM` / `YYYY-MM-DD HH:MM` interpreted as **UTC**. Must be ≥ now+60s and ≤ now+ `schedule_max_days_ahead` (default 90) days. Omitted/blank = immediate send (see §2.2).

* At least one of `message` or `media_url` is required.

2.1 `media_url` fetching & limits

`media_url` is fetched by ZNotice (never by the WhatsApp engine) behind an egress guard, then handed to the engine as base64:

- **Schemes:** `http` and `https`. **Ports:** default only (`80` / `443`); URLs with credentials (`http://user:pass@…`) are rejected.
- **Private-address blocking:** the hostname is resolved and **every** returned address must be public. Loopback (`127.0.0.0/8`, `::1`), private (`10/8`, `172.16/12`, `192.168/16`), link-local (`169.254/16`, `fe80::/10`), CGNAT (`100.64/10`), ULA (`fc00::/7`), multicast, reserved and IPv6-mapped forms of all of these are rejected with `400 invalid_media_url`. The validated IP is pinned for the connection so DNS rebinding cannot bypass the check.
- **Redirects:** followed manually, at most **3** hops, and every hop is re-validated with the same rules.
- **Limits: 30s** total timeout and **16MB** maximum size.
- **Content types:** `image/*`, `video/*`, `audio/*` and `application/pdf`. **SVG is excluded:** `image/svg+xml`, `image/svg` and any `+xml` content type (e.g. `application/xhtml+xml`) are rejected because they can carry active content. Anything else is also rejected.

```
curl -X POST https://znotice.consultbofa.com/api/v1/send \
  -H "Authorization: Bearer $ZNOTICE_API_KEY" \
  -H "Content-Type: application/json" \
  -d '{"phone":"250788123456","message":"Hello from ZNotice"}'
```

Success (`200`):

```
{
  "ok": true,
  "id": 42,
  "message_id": "true_250788123456@c.us_3EB0...",
  "credits_remaining": 999
}
```

- `id` is the local `messages.id`.
- `message_id` is the engine/WhatsApp message id, used to correlate delivery webhooks.
- `credits_remaining` is the balance after the atomic charge.

Media example:

```
curl -X POST https://znotice.consultbofa.com/api/v1/send \
  -H "Authorization: Bearer $ZNOTICE_API_KEY" \
  -H "Content-Type: application/json" \
  -d '{"phone":"0788123456","media_url":"https://example.com/receipt.pdf","message":"Your
receipt"}'
```

If the engine rejects the send, the credit is **automatically refunded** and a `failed` message row is stored; the response is `502 send_failed` with a short sanitized `detail` (no stack traces).

2.2 Scheduled sends

When `scheduled_at` is present and non-empty, `POST /api/v1/send` **enqueues** the message instead of sending it immediately and returns `202`:

```
{
  "ok": true,
  "scheduled_id": 17,
  "scheduled_at": "2026-09-12T09:30:00.000Z",
  "credits_balance": 999
}
```

- **No credit is charged at enqueue.** The balance is only pre-checked (balance ≥ 1) as a fast-fail guard; the scheduler charges **1 credit per message at actual send time** and auto-refunds on engine failure, so a job that never sends costs nothing.
- **Enqueue prechecks:** account `active` (`403 account_suspended`), WhatsApp session `connected` (`409 whatsapp_not_connected`), balance ≥ 1 (`402 insufficient_credits`), and the per-user pending cap not exceeded (`429 queue_full`).
- `scheduled_at` accepts an ISO-8601 string with a trailing `Z` or `±HH:MM` offset, **or** a naive `YYYY-MM-DDTHH:MM` / `YYYY-MM-DD HH:MM` (seconds optional) which is interpreted as UTC. An unparseable value, a time earlier than now+60s, or a time beyond the horizon returns `400 invalid_scheduled_at`.
- An omitted/blank `scheduled_at` keeps the existing immediate `200` behavior.
- Track and cancel enqueued messages with the queue endpoints in §2.4.

2.3 Bulk sends — `POST /api/v1/bulk`

Enqueues one message per recipient as a single **batch**. Every recipient is normalized (see §3) and duplicates are removed. A bulk job is created whether or not `scheduled_at` is set; omitted/blank sends on the next worker tick.

Request body (JSON):

Field	Type	Required	Notes
`recipients`	array	yes	1–`bulk_max_recipients` (default 1,000) entries. Each entry is either `{ "to": "...", "name": "..." }` or a bare phone string.
`message`	string	yes*	Template, 1–4,096 characters **after** placeholder resolution. May be omitted if `media_url` is set.
`media_url`	string	no	Same rules as §2.1; the resolved `message` becomes the caption.
`scheduled_at`	string	no	Same semantics as §2.2 (defaults to immediate).
`label`	string	no	≤120-character batch label shown in the dashboard/API.

* At least one of `message` or `media_url` is required.

Template placeholders are resolved **at enqueue time** per recipient: `{{name}}` → the recipient's `name` (empty if omitted) and `{{phone}}` → the normalized phone. Any other `{{token}}` is left untouched.

```
curl -X POST https://znotice.consultbofa.com/api/v1/bulk \
  -H "Authorization: Bearer $ZNOTICE_API_KEY" \
  -H "Content-Type: application/json" \
  -d '{
    "label": "September promo",
    "message": "Hi {{name}}, your order is ready.",
    "scheduled_at": "2026-09-12T09:30",
    "recipients": [
      { "to": "0788123456", "name": "Alice" },
      { "to": "+250788654321" }
```

```
    ]
  }'
```

Success (`202`):

```
{
  "ok": true,
  "batch_id": 8,
  "total": 2,
  "duplicates_removed": 0,
  "scheduled_at": "2026-09-12T09:30:00.000Z",
  "credits_balance": 1000
}
```

- `total` is the number of items actually enqueued (after dedupe).
- `duplicates_removed` counts recipients dropped because their normalized phone already appeared earlier in the list.
- If **any** recipient is invalid the whole request is rejected with `400 invalid_recipients` and up to 10 `{ "line", "value" }` entries (`line` is the 1-based position in `recipients`). No partial batch is created.
- More than `bulk_max_recipients` returns `400 too_many_recipients`.
- The enqueue prechecks match §2.2, except the balance check is balance ≥ `total`.

2.4 Queue management

All queue endpoints are scoped to the authenticated user; another user's ids return `404 not_found`.

Method & path	Purpose
`GET /api/v1/scheduled`	List scheduled items. Defaults to the open queue (`pending` + `sending`); `?status=` filters to one of `pending`, `sending`, `sent`, `failed`, `cancelled` (anything else → `400 invalid_status`). `?limit=` 1–200 (default 50). Newest first.
`GET /api/v1/scheduled/:id`	Fetch one item, including `body`, `media_url`, `next_attempt_at`, `attempts` and `last_error`.
`DELETE /api/v1/scheduled/:id`	Cancel one **pending** item. Returns `200 {ok, id, status:"cancelled"}`; if it is already `sending`/terminal → `409 not_pending`.
`GET /api/v1/batches`	List batches, newest first. `?limit=` 1–100 (default 20).
`GET /api/v1/batches/:id`	Fetch a batch plus a page of its items (ordered by id). `?limit=` 1–500 (default 200).
`DELETE /api/v1/batches/:id`	Cancel a `scheduled`/`processing` batch and all its remaining `pending` items. Returns `200 {ok, batch_id, cancelled}`; terminal batches → `409 not_cancellable`.

```
# Open queue
curl "https://znotice.consultbofa.com/api/v1/scheduled?status=pending&limit=100" \
  -H "X-API-Key: $ZNOTICE_API_KEY"

# Cancel a batch's remaining items
curl -X DELETE https://znotice.consultbofa.com/api/v1/batches/8 \
  -H "X-API-Key: $ZNOTICE_API_KEY"
```

An item returned as `sending` is already claimed by a worker and can no longer be cancelled — this is what makes dispatch **at-most-once**. A `pending` item is moved to `sending` immediately before the engine call; a crash mid-send leaves it `sending` until the reaper marks it `failed` with `last_error = interrupted_by_restart` (see ARCHITECTURE.md §8e and OPERATIONS.md §10).

3. Phone number normalization

`phone` is normalized server-side using `default_country_code` (admin setting, default `250`):

1. Spaces, dashes, parentheses and a leading `+` are stripped.
2. A leading international `00` prefix is removed.
3. If the number still starts with a national trunk `0`, the `0` is replaced by the configured country code. Example for `250`: `0788123456` → `250788123456`.
4. The result must be **8–15 digits**, otherwise `400 invalid_phone`.

Input	Normalized (cc=250)
`0788123456`	`250788123456`
`+250 788 123 456`	`250788123456`
`00250788123456`	`250788123456`
`250788123456`	`250788123456`
`123`	rejected → `invalid_phone`

4. Limits

Scope	Limit	Enforcement
Usage window (`?days=`)	1–90 days (default 30)	Clamped silently (never `400`)
Message length	4,096 characters	`400 message_too_long`
Media URL length	500 characters	`400 invalid_media_url`
Media size	16 MB	`400 invalid_media_url`
Media fetch timeout	30 s (all redirect hops)	`400 invalid_media_url`
Media redirects	3 hops, each re-validated	`400 invalid_media_url`
Bulk recipients	`bulk_max_recipients` (default 1,000 / job)	`400 too_many_recipients`
Schedule horizon	now+60 s … `schedule_max_days_ahead` (default 90 days)	`400 invalid_scheduled_at`
Pending scheduled queue	`scheduled_max_pending` (default 5,000 / user)	`429 queue_full`
Sends per API key	`send_rate_per_min` (default 60 / minute)	`429 rate_limited` + `retry_after`
All API requests	120 / minute / IP	`429 rate_limited` (global, no `retry_after`)

Scope	Limit	Enforcement
Concurrent WhatsApp sessions	`max_concurrent_sessions` (default 15)	Dashboard connect only → `409 at_capacity`

The per-key send limiter is an **in-memory fixed window** and resets when the application restarts. The global IP limiter is `express-rate-limit`.

5. Error codes

All API errors return JSON with `ok: false` and a machine-readable `error`. The global 404 handler returns `{ "ok": false, "error": "not_found" }` for unknown `/api/*` paths.

HTTP	`error`	Shape / extra fields	Meaning
400	`phone_required`	`{ ok:false, error }`	`phone` was empty.
400	`invalid_phone`	`{ ok:false, error }`	Not a valid 8–15 digit number after normalization.
400	`message_required`	`{ ok:false, error, message }`	Neither `message` nor `media_url` provided.
400	`message_too_long`	`{ ok:false, error, message }`	Message exceeds 4,096 characters.
400	`invalid_media_url`	`{ ok:false, error, message }`	`media_url` failed the egress guard: not `http(s)`, non-default port, credentials, non-public target, too many redirects, timeout, >16MB, a disallowed content type, or longer than 500 characters.
400	`invalid_scheduled_at`	`{ ok:false, error, message }`	`scheduled_at` is unparseable, earlier than now+60s, or beyond the schedule horizon.
400	`recipients_required`	`{ ok:false, error, message }`	Bulk request has no (or an empty) `recipients` array.
400	`too_many_recipients`	`{ ok:false, error, message }`	More than `bulk_max_recipients` in one bulk job.
400	`invalid_recipients`	`{ ok:false, error, invalid: [{line,value}] }`	One or more recipients failed normalization; up to 10 entries are returned.
400	`invalid_status`	`{ ok:false, error, message }`	`?status=` on `GET /scheduled` is not an allowed value.
400	`invalid_json`	`{ ok:false, error, message }`	Request body is not valid JSON.
413	`payload_too_large`	`{ ok:false, error, message }`	Request body exceeds the 1MB limit.
401	`missing_api_key`	`{ ok:false, error, message }`	No key in `Authorization` / `X-API-Key`.
401	`invalid_api_key`	`{ ok:false, error, message }`	Key hash not found.

HTTP	`error`	Shape / extra fields	Meaning
401	`revoked_api_key`	`{ ok:false, error, message }`	Key exists but was revoked.
402	`insufficient_credits`	`{ ok:false, error, balance }`	Immediate: balance < 1. Bulk enqueue: balance < recipient count. A scheduled item can also fail with this at dispatch if the balance was spent before it ran (no credits are lost).
403	`account_suspended`	`{ ok:false, error }`	User status is not `active`.
404	`user_not_found`	`{ ok:false, error }`	User record no longer exists.
404	`not_found`	`{ ok:false, error, message }`	Unknown `/api/*` endpoint, or a scheduled item/batch that does not exist or belongs to another user.
409	`whatsapp_not_connected`	`{ ok:false, error }`	No `connected` WhatsApp session.
409	`not_pending`	`{ ok:false, error, message }`	Cancel target is not `pending` (it may already be sending).
409	`not_cancellable`	`{ ok:false, error }`	Batch is already terminal (`completed` / `cancelled`).
429	`queue_full`	`{ ok:false, error, message }`	Per-user pending scheduled queue cap (`scheduled_max_pending`) reached.
429	`rate_limited` (send)	`{ ok:false, error, message, retry_after }`	Per-key send limit exceeded.
429	`rate_limited` (global)	`{ ok:false, error, message }`	120 req/min/IP exceeded.
500	`server_error`	`{ ok:false, error, message }`	Unexpected server error.
502	`send_failed`	`{ ok:false, error, detail }`	Engine rejected the send; credit refunded.

6. Delivery webhooks (outbound status)

ZNotice receives delivery updates from the WhatsApp engine and records them against your message. The status lifecycle is:

Status	Trigger (engine ack)	Meaning
`sent`	ack `1`	Accepted by WhatsApp servers.
`delivered`	ack `2`	Delivered to the recipient's device.
`read`	ack `3` or `4`	Read / played by the recipient.
`failed`	negative ack (`-1` … `-7`)	Send failed.

Statuses are **monotonic** — a later webhook never downgrades a status (for example, `read` is not overwritten by `delivered`). `queued` is the initial database default; successful API sends are stored as `sent` immediately.

To track delivery you can either poll `GET /api/v1/messages` and match on `wpp_message_id`, or subscribe to customer outbound webhooks (§6.1).

6.1 Customer outbound webhooks

Configure an endpoint from **Dashboard → Webhooks**. ZNotice POSTs a JSON body to your URL whenever one of your messages changes state. Delivery is best-effort with retries and is **disabled by default** until you save a URL.

Events

Event	Fired when	Payload `data` fields
`message.sent`	The engine accepted the send.	`message_id`, `to_number`, `status`, `wpp_message_id`, `credits_charged`
`message.failed`	The engine rejected the send (credit refunded).	`message_id`, `to_number`, `status`, `error`, `credits_charged`
`message.delivered`	The recipient's device received it (ack `2`).	`message_id`, `to_number`, `status`, `ack`
`message.read`	The recipient read/played it (ack `3` / `4`).	`message_id`, `to_number`, `status`, `ack`
`batch.completed`	A scheduler-driven bulk batch finished with **zero** cancellations.	`batch_id`, `total`, `sent`, `failed`, `cancelled`
`test.ping`	You click **Send test event** in the dashboard.	`message`, `user_id`

Events are emitted only on a real status transition, so a repeated engine ack never produces a duplicate delivery. `batch.completed` is emitted **at most once**, only by the scheduler when the last open item finishes and `cancelled_count` is `0`. A fully-cancelled batch becomes `cancelled` silently and a partially-cancelled batch becomes `completed` silently — neither emits the event. Batch status/progress is also available from `GET /api/v1/batches/:id`.

Request

```
POST /your/endpoint HTTP/1.1
Content-Type: application/json
User-Agent: ZNotice-Webhooks/1.0
X-ZNotice-Event: message.delivered
X-ZNotice-Signature: sha256=<hex hmac_sha256(secret, rawBody)>
```

```
{
  "event": "message.delivered",
  "created_at": "2026-09-11T05:30:00.000Z",
  "data": { "message_id": 42, "to_number": "250788123456", "status": "delivered", "ack": 2 }
}
```

The `X-ZNotice-Signature` value is `sha256=` followed by the hex digest of `HMAC-SHA256(signing_secret, raw_request_body)`. The signing secret is shown on **Dashboard → Webhooks** and can be regenerated at any time (regeneration invalidates the previous secret).

Endpoint requirements (SSRF guard): the URL must be a public `http(s)` URL on port `80` / `443`. Loopback (`127.0.0.0/8` , `::1`), private (`10/8` , `172.16/12` , `192.168/16`), link-local (`169.254/16` , `fe80::/10`), CGNAT (`100.64/10`), ULA (`fc00::/7`) and reserved addresses are rejected at save time and re-validated (with DNS pinning) before every delivery.

Retry policy: a request has a **10s timeout**; any **2xx** response is success. Any non-2xx response or network error increments the attempt counter and schedules a retry **+1 minute**, then **+5 minutes**. After **3 failed attempts** the delivery is marked `failed` and no further attempts are made. The last 25 attempts (status, attempts, response code, error) are visible in the dashboard.

6.2 Verifying the signature (Node.js)

```
const crypto = require('crypto');

// Keep the raw body bytes — do not JSON.parse before verifying.
function verify(rawBody, header, secret) {
  const expected = 'sha256=' + crypto.createHmac('sha256', secret).update(rawBody).digest('hex');
  const a = Buffer.from(String(header || ''));
  const b = Buffer.from(expected);
  return a.length === b.length && crypto.timingSafeEqual(a, b);
}

// Express example (raw body available):
// const ok = verify(req.rawBody, req.get('X-ZNotice-Signature'),
process.env.ZNOTICE_WEBHOOK_SECRET);
// if (!ok) return res.sendStatus(401);
```

Always compare with a timing-safe function and reject requests whose signature does not match before acting on the payload.

7. Node.js integration example

```
// send-whatsapp.mjs — Node 18+ (global fetch)
const BASE = 'https://znotice.consultbofa.com';
const API_KEY = process.env.ZNOTICE_API_KEY; // zn_live_...

async function sendWhatsApp(phone, message, mediaUrl, scheduledAt) {
  const res = await fetch(`${BASE}/api/v1/send`, {
    method: 'POST',
    headers: {
      'Authorization': `Bearer ${API_KEY}`,
      'Content-Type': 'application/json',
    },
    body: JSON.stringify({
      phone,                 // e.g. "0788123456" (default cc 250) or "250788123456"
      message,
      ...(mediaUrl ? { media_url: mediaUrl } : {}),
      // ISO-8601, or "YYYY-MM-DDTHH:MM" interpreted as UTC. Omit to send now.
      ...(scheduledAt ? { scheduled_at: scheduledAt } : {}),
    }),
  });

  const data = await res.json();
```

```
  if (!res.ok || !data.ok) {
    // data.error is a stable code; data.detail is present for send_failed
    throw new Error(`ZNotice send failed (${res.status}): ${data.error} ${data.detail || ''}`);
  }
  // Immediate 200: { ok, id, message_id, credits_remaining }
  // Scheduled 202: { ok, scheduled_id, scheduled_at, credits_balance }
  return data;
}

async function scheduleBulk(recipients, message, { scheduledAt, label } = {}) {
  const res = await fetch(`${BASE}/api/v1/bulk`, {
    method: 'POST',
    headers: {
      'Authorization': `Bearer ${API_KEY}`,
      'Content-Type': 'application/json',
    },
    body: JSON.stringify({
      recipients,            // [{ to: "0788123456", name: "Alice" }, ...]
      message,               // supports {{name}} and {{phone}}
      ...(scheduledAt ? { scheduled_at: scheduledAt } : {}),
      ...(label ? { label } : {}),
    }),
  });

  const data = await res.json();
  if (!res.ok || !data.ok) {
    throw new Error(`ZNotice bulk failed (${res.status}): ${data.error}`);
  }
  return data; // { ok, batch_id, total, duplicates_removed, scheduled_at, credits_balance }
}

// Usage
sendWhatsApp('0788123456', 'Hello from my app')
  .then((r) => console.log('sent, credits left:', r.credits_remaining))
  .catch((e) => console.error(e.message));

scheduleBulk(
  [{ to: '0788123456', name: 'Alice' }, { to: '+250788654321' }],
  'Hi {{name}}, your order is ready.',
  { scheduledAt: '2026-09-12T09:30', label: 'September promo' }
).then((r) => console.log('batch', r.batch_id, 'queued:', r.total));
```

Retry guidance

- `429 rate_limited`: wait `retry_after` seconds before retrying.
- `502 send_failed`: the credit was refunded; a retry is safe (a new credit will be charged).
- `402 insufficient_credits`: top up via the dashboard, then retry.
- `202` from `/send` or `/bulk`: the message(s) are **queued**, not delivered. Do not retry — poll `GET /api/v1/scheduled` / `GET /api/v1/batches/:id` to follow progress.
- `429 queue_full`: the per-user pending cap is reached; wait for the queue to drain before enqueuing more.
- Do **not** blindly retry `4xx` validation errors.