Mocha Media

Digital marketing — innovative thinking and creation

LEAD DISTRIBUTION NETWORK

Lead Buyer API Integration Guide

Ping / Post specification for receiving real-time leads

This document explains how to connect your platform to the Mocha Media Lead Manager & Distribution System so you can receive leads in real time. It covers the two-step Ping/Post protocol, the endpoints you must expose, the authentication we support, the request and response formats we expect, and the exact information we need from you to complete the integration on our side.

What you will find inside

- **Integration overview.** How our distribution engine calls your endpoints and how quotas work.
- **Ping/Post protocol.** The Ping request/response and Post request/response contracts.
- **Authentication.** Headers, tokens and the return-token flow between Ping and Post.
- **Data mapping.** The field mapping we configure so your payload matches your schema.
- **What we need from you.** A single onboarding form of everything we need from you.

PROTOCOL: PING / POST

FORMAT: JSON OVER HTTPS

METHOD: POST

1. How the integration works

We capture leads from our campaigns and distribute each lead to eligible buyers assigned to that campaign. For every lead, our system contacts your platform on your behalf — you do not push data to us, we push to you. Buyers are processed in order until one accepts the lead.

The distribution flow

- **1. Capture.** A lead is captured against a specific campaign and queued for distribution.
- **2. Quota check.** For each buyer assigned to that campaign, we first check the daily/period quota. If the quota is reached, the buyer is skipped.
- **3. Ping (optional).** If you provide a Ping URL, we send a lightweight Ping to ask whether you want the lead. If you do not use Ping, we go straight to Post.
- **4. Post.** On a successful Ping we send the full lead to your Post URL. Values returned by your Ping (e.g. an access token or reference id) can be injected into the Post request.
- **5. Accept / reject.** An HTTP 200 from your Post endpoint marks the lead as Sold to you and increments your quota. Any other status is treated as a rejection and we move to the next buyer.

> **The single rule that matters**
> A lead is considered ACCEPTED only when your Post endpoint returns HTTP status 200. Any other status code (4xx, 5xx, redirects, etc.) is treated as a rejection, and we offer the lead to the next buyer in line. Please return 200 only when you have truly accepted and stored the lead.

What you build on your side

- **Post endpoint.** A Post endpoint (required) that accepts a JSON lead and returns 200 on acceptance.
- **Ping endpoint.** A Ping endpoint (optional) that accepts a partial lead and returns availability.
- **Transport.** Both endpoints must accept HTTP POST with a JSON body and be reachable over public HTTPS.

2. Authentication

Authentication is driven entirely by the HTTP headers you specify. When you onboard, you tell us the exact headers (and values) to send with your Ping and Post requests, and we attach them to every call to your endpoints. This lets you use whatever scheme your platform already uses.

Common header schemes we support

```
Authorization: Bearer <your-token>
X-API-Key: <your-api-key>
Content-Type: application/json
```

You may specify any number of custom headers. We store them per endpoint (separate header sets for Ping and Post are supported). Sensitive header values such as Authorization, Cookie and X-API-Key are automatically redacted in our internal logs.

> **Return-token flow (Ping !' Post)**
> Any field your Ping response returns can be referenced inside the Post request using the placeholder {{ping_response_<key>}}. For example, if your Ping returns {"access_token":"abc"}, we can place {{ping_response_access_token}} into a Post header or body field. This is how you pass a per-lead token or reference id from Ping into Post.

Note on inbound security (our side)

Separately, leads entering our system are protected by a shared secret header (X-Lead-Secret). This secures traffic into our platform and is not something you need to implement to receive leads — it is mentioned only for completeness.

3. The Ping request

The Ping is an optional pre-check. It lets you decide whether to accept a lead and return a reference id or token before we send the full data. If you do not expose a Ping URL, we skip this step and send directly to Post.

Request we send

Method POST, JSON body, with the headers you configured for Ping. The body contains the fields we mapped for your Ping payload:

```
POST https://your-platform.com/api/ping
Authorization: Bearer <your-token>
Content-Type: application/json

{
  "campaign_id": "A1B2C3D4",
  "zip": "90210",
  "state": "CA",
  "lead_id": 10432
}
```

Response we expect

Return HTTP 200 with a JSON object. Every top-level key you return becomes available to the Post step as {{ping_response_<key>}}. Typical fields:

ping_id	An identifier for this Ping. Echo it back in Post so you can correlate the two calls.
access_token	A short-lived token to authorize the follow-up Post request.
accepted	Optional boolean signalling whether you want the lead.

If your Ping call fails at the network level, we log the error and move on to the next buyer. Design your Ping to respond quickly (well under 30 seconds).

4. The Post request

The Post delivers the full lead. This is the only required endpoint. We build the JSON body from the field mapping you provide, injecting any values returned by your Ping.

Request we send

```
POST https://your-platform.com/api/post
Authorization: Bearer {{ping_response_access_token}}
Content-Type: application/json

{
  "campaign_id": "A1B2C3D4",
  "ping_id": "{{ping_response_ping_id}}",
  "first_name": "Jane",
  "last_name": "Doe",
  "email": "jane.doe@example.com",
  "phone": "3105550142",
  "address": "123 Main St",
  "city": "Beverly Hills",
  "state": "CA",
  "zip": "90210",
  "lead_id": 10432
}
```

The exact field names, nesting and which lead attributes are included are all configurable — we shape the payload to match your documented schema during onboarding.

Response we expect

Return HTTP 200 to accept. We read these optional fields from your JSON response body:

document_sign_url	A URL where the lead can review/sign a document. If present, we surface it back to the lead source.
error_description	A human-readable reason when you accept with 200 but flag an issue. Captured for reporting.

We also record your HTTP status code and raw response body against the lead for auditing. On a non-200 response we mark the attempt as a rejection with your returned code and body, then try the next buyer.

> **Idempotency recommended**
> Because a lead can be retried, please de-duplicate on your side using a stable key such as our lead_id (or your own ping_id). Returning 200 for a duplicate you have already stored is safe and preferred.

5. Data mapping & available fields

Lead content varies per campaign. We use a placeholder mapping to translate our captured lead into the precise JSON your endpoints expect, so you receive keys named the way your system wants them.

How mapping works

- **Your schema.** You give us your target field names and structure (including any nested objects).
- **Placeholders.** We fill each field using {{placeholder}} tokens that reference lead data, e.g. {{email}} or {{address.zip}} for nested source data.
- **System tokens.** System values are also available: {{lead_id}} (our unique id) and {{campaign_id_slug}} (the campaign's public id).
- **Ping tokens.** Ping return values are available in Post as {{ping_response_<key>}}.

Commonly available lead fields

These are typical fields captured across campaigns. The precise set is confirmed per campaign during onboarding.

```
first_name    last_name     email         phone
address       city          state         zip
campaign      lead_id       campaign_id_slug
```

> **Test-lead filtering**
> Any lead whose first name contains the word "test" is automatically filtered on our side and is NOT distributed to buyers. Keep this in mind when validating your endpoints — use a non-test first name for live end-to-end checks.

6. What we need from you to integrate

Please complete the details below and return them to the Mocha Media team (mochamedia.co.uk). Once we have these, we configure you as a buyer, map your fields, and run a test lead with you before going live.

A. Endpoints

Post URL (required)	The HTTPS endpoint that receives the full lead and returns 200 on acceptance.
Ping URL (optional)	The HTTPS endpoint for the pre-check step, if you use Ping/Post.
Supported method	Confirm POST with a JSON body (this is what we send).

B. Authentication

Post headers	Exact header name(s) and value(s) to send with Post (e.g. Authorization: Bearer ...).
Ping headers	Header set for Ping, if different from Post.
Token flow	If Post auth comes from a Ping response, tell us which response key holds it.

C. Payload schema

Field list	The exact JSON keys you expect, including data types and any nested structure.
Required vs optional	Which fields must be present for you to accept a lead.
Sample payload	An example JSON body your endpoint accepts, so we can mirror it.

D. Responses & business rules

Success signal	Confirm HTTP 200 means accepted (and any body fields like document_sign_url you will return).
Rejection codes	Status codes/reasons you return when declining a lead.
Daily/period cap	The maximum number of leads you want per campaign per period (your quota).
Campaigns / verticals	Which campaigns or lead types you want to receive.

Go-live checklist

1) Endpoints reachable over HTTPS. 2) Auth headers confirmed. 3) Field mapping agreed. 4) Test lead returns 200 end-to-end. 5) Quota and campaigns set. Once all five are green, we switch you to live traffic.

Appendix A — Worked example

Below is a real buyer configuration shown end to end so you can see how the Ping/Post pieces fit together. Credentials have been redacted — the live values are exchanged securely and never sent in this document.

Step 1 — Ping: obtain an access token

For this buyer the Ping endpoint is an OAuth 2.0 token endpoint. We POST client credentials and receive a short-lived access token in the response. Nothing else about the lead is sent at this stage.

```
POST <PING ENDPOINT URL>
Content-Type: application/json

{
  "client_id": "MochaMedia",
  "client_secret": "<redacted — shared securely, not stored in this doc>",
  "grant_type": "client_credentials",
  "scope": "lead:get lead:create"
}
```

The token endpoint replies with JSON containing an access_token. Every key in that response becomes available to the Post step as {{ping_response_<key>}} — so {{ping_response_access_token}} carries the bearer token into Step 2.

Step 2 — Post: deliver the lead

We then POST the full lead to the leads endpoint, injecting the token from Step 1 into the Authorization header. The body is built from the agreed field mapping (placeholders shown).

```
POST <LEADS ENDPOINT URL>
Content-Type: application/json
Authorization: Bearer {{ping_response_access_token}}

{
  "lead_id": "{{lead_id}}",
  "user": {
    "person": {
      "first_name": "{{first_name}}", "last_name": "{{last_name}}",
      "birthdate": "{{dob}}", "category": "adult",
      "reservation_number": "{{booking_reference}}"
    },
    "email": "{{email}}", "country": "{{country}}",
    "phone": "{{phone}}", "postcode": "{{postcode}}", "locale": "en"
  },
  "flights": [{
    "flight_number": "{{flight_number}}",
    "airline": { "name": "{{airline}}", "iata": "{{iata}}" },
    "departure_airport": { "iata": "{{departure_airport}}" },
    "arival_airport": { "iata": "{{arrival_airport}}" },
    "departure_time": "{{date_of_travel}}T00:00:00+00:00",
    "flight_was": "{{flight_was}}", "booking_reference": "{{booking_reference}}"
  }],
  "additional_passengers": [ /* pass_2 ... pass_9, same shape */ ],
  "reason": {
    "type": "{{flight_was}}", "delayed_by": "{{delayed_by}}",
    "date_of_travel": "{{date_of_travel}}"
  },
  "user_comment": "{{comments}}",
  "problem_flight": 1
}
```