Neural Ads Partner Data API: test kit

This kit lets your engineering team test sending us two kinds of data before you go live:

- **Conversions.** Your server posts conversion events (purchases, leads, sign-ups) as they happen. The body follows Meta's Conversions API, so if you already send to Meta, you can keep the same payload.
- **Audiences.** You create an audience, then add or remove people in batches of hashed emails, IP addresses or mobile ad IDs.

Everything you send with this kit goes to **Neural Ads' own test account on our staging environment**, not to your account and not to production. Test data is stored and shown to us so we can check it with you, but it is never used for measurement and never sent to an ad platform. Send whatever you need to; you cannot break anything.

When the test passes, we give you production credentials for your own account. Only the URL, the token and the IDs change; your code and payloads stay the same. See Moving to production.

The fastest way to check everything is the script that comes with this guide:

```
python3 neuralads_stage_test.py
```

It needs only Python 3.8 or later, sends one round of conversions and audience members, reads back the results and prints PASS or FAIL for each step. The rest of this guide shows the same calls one at a time with `curl`.

Your test credentials

API base URL	`https://stage.neuralads.ai/v1`
Token	[TOKEN REMOVED]
Advertiser ID	`cb66550d-cca6-4c08-b0cb-d3bad004265b` (Neural Ads test account)
Conversions dataset ID	`ds_33659f9a2043b3b3a8e0e2ee`
Token permissions	send events, create and update audiences, read reports
Token expires	November 19, 2026 (UTC)

Treat the token like a password even though it only reaches test data: keep it on your servers and out of browsers, mobile apps and public repositories. If it leaks, tell us and we will revoke it and issue a new one.

Set these once in your shell for the examples below:

```
export NEURALADS_URL="https://stage.neuralads.ai/v1"
export NEURALADS_TOKEN="[TOKEN REMOVED]"
export NEURALADS_DATASET_ID="ds_33659f9a2043b3b3a8e0e2ee"
```

Two headers on every call

```
Authorization: Bearer <token>
X-Neural-Test: true
```

- **`Authorization`** carries the token. Send it only in this header; a token in the URL or the body is refused with `400 token_in_url_or_body`.
- **`X-Neural-Test: true`** marks the call as a test. Keep it on every call in this kit. In production you will remove it.

The token belongs to one advertiser, so you do not need to send `advertiser_id`. If you do send it, it must be `cb66550d-cca6-4c08-b0cb-d3bad004265b`; anything else is refused with `403 advertiser_mismatch`.

Hashing emails

Hash emails exactly this way, or conversions will not match the people in your audiences:

1. Trim leading and trailing spaces.
2. Lowercase.
3. SHA-256, written as 64 lowercase hex characters.

Do not remove dots or `+tags`. Check your code against this value:

Input	Expected hash
`alice@example.com`	`ff8d9819fc0e12bf0d24892e45987e249a28dce836a85cad60e28eaaa8c6d976`
`Alice@Example.COM`	the same

```python
import hashlib

def hem(email: str) -> str:
    return hashlib.sha256(email.strip().lower().encode("utf-8")).hexdigest()
```

```sql
-- Snowflake
SELECT SHA2(LOWER(TRIM(email)), 256) FROM customers;
-- BigQuery
SELECT TO_HEX(SHA256(LOWER(TRIM(email)))) FROM customers;
```

You may also send plaintext emails and phone numbers. We hash them as soon as the request arrives and never store the original.

Part 1. Conversions

Send one event

```
curl -sS -X POST "$NEURALADS_URL/datasets/$NEURALADS_DATASET_ID/events" \
  -H "Authorization: Bearer $NEURALADS_TOKEN" \
  -H "X-Neural-Test: true" \
  -H "Content-Type: application/json" \
  -d '{
    "data": [{
      "event_name": "Purchase",
      "event_time": '"$(date +%s)"',
      "event_id": "test-order-1",
      "action_source": "website",
      "user_data": {"em": ["ff8d9819fc0e12bf0d24892e45987e249a28dce836a85cad60e28eaaa8c6d976"]},
      "custom_data": {"value": 49.99, "currency": "USD", "order_id": "test-order-1"}
    }]
  }'
```

A working setup answers `200`:

```
{"events_received": 1, "events_stored": 1, "events_deduplicated": 0, "events_rejected": 0,
 "messages": [], "is_test": true, "trace_id": "a91f3c07d2b84e15"}
```

Send the same command again. `events_deduplicated` becomes `1`, because the `event_id` is the same. That is what makes retries safe: if a request times out, send it again with the same `event_id`s.

What an event needs

Field	Required	Rules
`event_name`	yes	A Meta standard event (`Purchase`, `Lead`, `CompleteRegistration`, `AddToCart`, `Subscribe`, ...) or your own name, up to 64 letters, digits, spaces, `_`, `.` or `-`.
`event_time`	yes	Unix time in **seconds** (not milliseconds). At most 62 days old and at most five minutes in the future.

Field	Required	Rules
`action_source`	yes	`website`, `app`, `email`, `phone_call`, `chat`, `physical_store`, `system_generated`, `business_messaging` or `other`.
`event_id`	strongly recommended	Your unique ID for the event, up to 128 characters. Used to drop duplicates.
`user_data`	yes	At least one of `em` (email), `ph` (phone), `external_id` (your customer ID). Hashed or plaintext.
`custom_data.value`	`Purchase`; optional elsewhere	The transaction's monetary value, a non-negative number.
`custom_data.currency`	with `value`	Three-letter currency code such as `USD`.
`custom_data.order_id`	no	Your order or transaction ID.
`custom_data.content_ids`	no	Product IDs, up to 50.
`custom_data.content_type`	no	For example `product`.
`custom_data.num_items`	no	Number of items.
`custom_data.predicted_ltv`	no	Predicted lifetime value of the customer.

`event_name` is the conversion type. Use a Meta standard name where one fits, or your own (for example `QuoteRequested`); custom names are stored as sent and listed in a warning. Other `custom_data` keys are dropped. For example, a lead worth $120:

```
{"event_name": "Lead", "event_time": 1761000000, "event_id": "lead-5531", "action_source": "websit
        e",
 "user_data": {"em": "jane@example.com"},
 "custom_data": {"value": 120.00, "currency": "USD", "predicted_ltv": 900}}
```

Send up to 1,000 events per request. If some events in a request are invalid, the valid ones are still stored, and `messages` lists each rejected event by its position (`index`, from 0) with the reason:

```
{"events_received": 2, "events_stored": 1, "events_rejected": 1,
 "messages": [{"index": 1, "error": "event_time is older than 62 days"}], "is_test": true, "trace_i
        d": "..."}
```

Send `em` or a plaintext `ph` on as many events as you can. Those are the events we can match to people for measurement. A phone that arrives already hashed, or an event with only `external_id`, is stored but cannot be matched.

Check what we received

```
curl -sS "$NEURALADS_URL/datasets/$NEURALADS_DATASET_ID/requests?limit=10" \
  -H "Authorization: Bearer $NEURALADS_TOKEN"
```

This lists your latest calls, including refused ones, with the status code, the counts and the reason for every rejection.

Part 2. Audiences

Create an audience

```
curl -sS -X POST "$NEURALADS_URL/audiences" \
  -H "Authorization: Bearer $NEURALADS_TOKEN" \
  -H "X-Neural-Test: true" \
  -H "Content-Type: application/json" \
  -d '{"name": "Test audience", "purpose": "targeting"}'
```

```
{"id": "aud_7c1e...", "name": "Test audience", "purpose": "targeting", "source": "api",
 "is_test": true, "status": "building", "member_count": 0, "trace_id": "..."}
```

Keep the `id`:

```
export NEURALADS_AUDIENCE_ID="aud_7c1e..."
```

`purpose` is `targeting` (show ads to these people), `suppression` (never show ads to them) or `roster` (the people in a lift experiment). It cannot change later.

Add people

`schema` names the columns and `data` holds one row per person. A row can carry several identifiers; leave a cell empty when you don't have that one.

```
curl -sS -X POST "$NEURALADS_URL/audiences/$NEURALADS_AUDIENCE_ID/members" \
  -H "Authorization: Bearer $NEURALADS_TOKEN" \
  -H "X-Neural-Test: true" \
  -H "Content-Type: application/json" \
  -d '{
    "schema": ["EMAIL_SHA256", "IP", "MAID_AAID"],
    "data": [
      ["ff8d9819fc0e12bf0d24892e45987e249a28dce836a85cad60e28eaaa8c6d976", "", ""],
      ["", "8.8.8.8", ""],
      ["", "", "38400000-8cf0-11bd-b23e-10b96e40000d"]
    ],
    "upload_id": "test-upload-1"
  }'
```

The answer (`202`) says how many rows were accepted. Members are applied in the background, usually within one to two minutes:

```
{"upload_id": "test-upload-1", "status": "processing", "rows_received": 3, "rows_accepted": 3,
 "rows_rejected": 0, "rejections": [], "trace_id": "..."}
```

Column	Value
`EMAIL_SHA256`	SHA-256 of the trimmed, lowercased email (the rule above).
`EMAIL`	A plaintext email; we hash it.
`PHONE`	A plaintext phone number; we normalize and hash it. Pre-hashed phones are refused.
`IP`	A public IPv4 address.
`MAID_IDFA` , `MAID_AAID`	An Apple or Google mobile advertising ID.

Send up to 10,000 rows per request. **`upload_id`** makes a batch safe to resend: the same body with the same `upload_id` returns the first answer and changes nothing. Use a new `upload_id` for every new batch.

Check the result

```
curl -sS "$NEURALADS_URL/audiences/$NEURALADS_AUDIENCE_ID/reports/test-upload-1" \
  -H "Authorization: Bearer $NEURALADS_TOKEN"
```

```
{"upload_id": "test-upload-1", "op": "add", "status": "applied", "rows_accepted": 3, "rows_rejecte
       d": 0,
 "membership": {"added": 3, "removed": 0, "unchanged": 0}, "active": {"hem_sha256": 1, "ip": 1, "ma
       id": 1}}
```


`GET $NEURALADS_URL/audiences/$NEURALADS_AUDIENCE_ID` shows the audience's status and counts. A test audience under 100 people shows `too_small`; that is expected.

Remove people

Same body, `DELETE` instead of `POST`, with a new `upload_id`:

```
curl -sS -X DELETE "$NEURALADS_URL/audiences/$NEURALADS_AUDIENCE_ID/members" \
  -H "Authorization: Bearer $NEURALADS_TOKEN" \
  -H "X-Neural-Test: true" \
  -H "Content-Type: application/json" \
  -d '{"schema": ["EMAIL_SHA256"],
       "data": [["ff8d9819fc0e12bf0d24892e45987e249a28dce836a85cad60e28eaaa8c6d976"]],
       "upload_id": "test-upload-2"}'
```

To send a complete list instead of changes, use `POST .../members/replace`; see "Audiences" in the reference guide that comes with this kit.

If something goes wrong

Every refusal has the same shape:


```
{"error": {"code": "scope_missing", "message": "..."}, "trace_id": "a91f3c07d2b84e15"}
```


Response	Fix
`401 token_invalid`	The token is wrong or not sent as `Authorization: Bearer <token>`.
`401 token_expired`, `token_revoked`	Ask us for a new token.
`400 token_in_url_or_body`	Move the token into the `Authorization` header.
`403 advertiser_mismatch`	Remove `advertiser_id` from the body, or set it to `cb66550d-cca6-4c08-b0cb-d3bad004265b`.
`404 dataset_not_found`	Use the dataset ID above.
`400 all_events_rejected`	Every event failed; `messages` says why for each `index`.
`400 all_rows_rejected`	No row was valid; `rejections` gives the reason for each `index`.

Response	Fix
`409 upload_id_conflict`	That `upload_id` was already used for a different batch. Use a new one.
`429 rate_limited`	Wait the number of seconds in `Retry-After`, then retry.
`5xx`	Our side. Retry with exponential backoff.

Limits

Limit	Value
Event requests (`POST /v1/datasets/{id}/events`)	100 per second per token
Audience member requests (add, remove, replace)	20 per second per token
All requests per token, combined (including reads and MCP calls)	100 per second
Short bursts	Up to 200 requests above the rate (40 for member requests), then refused
Request body	10 MB (`413 payload_too_large` above it)
Events per request	1,000
Member rows per request	10,000

Event and member responses carry `X-RateLimit-Limit` and `X-RateLimit-Remaining`. A refused request gets `429 rate_limited` with `Retry-After` (seconds); wait that long and resend the same batch. Send large volumes as fewer, bigger batches rather than many small requests.

Rejected rows never echo the identifier back, only the row's `index` and the reason. When you contact us, include the `trace_id` (or `upload_id`) and the time you sent it. Never put raw emails or phone numbers in a message.

Moving to production

When your test passes, tell your Neural Ads contact. We will then give you:

	Test (this kit)	Production
API base URL	`https://stage.neuralads.ai/v1`	`https://platform.neuralads.ai/v1`
Token	the one above	a new token for your account
Advertiser ID	`cb66550d-cca6-4c08-b0cb-d3bad004265b` (Neural Ads)	your own advertiser ID
Dataset ID	`ds_33659f9a2043b3b3a8e0e2ee`	your production dataset ID

Then:

1. Swap in the production URL, token and dataset ID. Nothing else in your code changes.
2. Remove the `X-Neural-Test` header.
3. Create your production audiences **without** `X-Neural-Test`. A test audience stays a test audience; it cannot be switched to live.
4. Before switching over, check:
   - [ ] The token is in a secret manager and only used server-side.
   - [ ] Every event has a unique `event_id`, and `event_time` is in seconds.
   - [ ] Most events carry `em` or a plaintext `ph`.
   - [ ] Every member batch has a unique `upload_id`, reused only when resending that batch.
   - [ ] Retries back off on `5xx`, timeouts and `429`; other `4xx` responses are logged, not retried.
   - [ ] `messages` and `rejections` are logged with the `trace_id`.

The full reference (every field, limit and error code, full-list replacement, and using the API from an AI assistant) is in `partner-guide.md`, included with this kit. Its examples use the production URL; during testing, use `https://stage.neuralads.ai/v1` instead.