

Neural Ads Data MCP server: conversions and audiences

The Neural Ads Data MCP server lets any MCP-compatible client send us two kinds of data:

- **Conversions.** Conversion events (purchases, leads, sign-ups) in Meta's Conversions API format.
- **Audiences.** Lists of people, as hashed emails, IP addresses or mobile ad IDs, that you add to and remove from.

It is the same service as the Neural Ads Partner Data REST API, exposed as [Model Context Protocol](#) tools. How you call it is up to you: an AI assistant, an automation platform, an agent you build, or your own code.

This document covers your **test connection**. Everything you send 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.

Connection details

Server URL	https://stage.neuralads.ai/partner/mcp
Transport	MCP Streamable HTTP (protocol version 2025-03-26)
Authentication	HTTP header Authorization: Bearer [TOKEN REMOVED]
Advertiser	Neural Ads test account, cb66550d-cca6-4c08-b0cb-d3bad004265b
Conversions dataset ID	ds_33659f9a2043b3b3a8e0e2ee
Token permissions	send events, create and update audiences, read reports
Token expires	November 19, 2026 (UTC)

The token is the only credential. Send it on every HTTP request to the server, in the `Authorization` header. Keep it out of shared documents and public repositories; if it leaks, tell us and we will revoke it and issue a new one.

Client requirements

Your MCP client must be able to:

1. Connect to a **remote** MCP server over Streamable HTTP, and

2. Send a **custom Authorization header** with each request.

The server does not offer OAuth sign-in. If your client can only start local servers (stdio) or only connects with OAuth, put a local bridge between them that forwards the header, such as the open-source `mcp-remote` package (needs Node.js 22 or later):

```
npx -y mcp-remote https://stage.neuralads.ai/partner/mcp --header "Authorization:Bearer [TOKEN REMOVED]
J_CCzRX9j8ZjA0i38vntqIBR56aDWhDLxJkvE"
```

What the token can reach

The token belongs to the Neural Ads test advertiser only. It cannot see or change any other advertiser's data, and the server lists only the tools its permissions allow: 10 of the 11 below. It does not include `partner_create_dataset`, because your test dataset already exists.

Rules for every call

Test mode by default. Every write (events, audiences, members) runs in test mode unless the call passes `"live": true`. Each write result says which mode it used in `mode`. For this test connection, never pass `live`: there is nothing to gain on staging.

Hashed identifiers only. People are identified by hashes, never by plaintext. The server refuses plaintext emails and phone numbers with `plaintext_not_allowed_over_mcp` before anything is stored. (MCP calls often pass through AI providers and automation logs, so plaintext is not accepted here; the REST API accepts it.)

Hash an email like this:

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

Do not remove dots or `+tags`.

Input	SHA-256 hash
<code>alice@example.com</code>	<code>ff8d9819fc0e12bf0d24892e45987e249a28dce836a85cad60e28eaaa8c6d976</code>
<code>Alice@Example.COM</code>	the same

Phone numbers sent already hashed are accepted for conversions but cannot be matched to people for measurement, so prefer `em` (email). Your own customer IDs (`external_id`) may be plaintext.

Batch sizes. At most 100 events, or 1,000 audience rows, per call. For bulk loads use the REST API.

Safe retries. Events with the same `event_id` are stored once. Member batches with the same `upload_id` are applied once. Reuse them when you resend after a timeout.

All tools

The server has 11 tools: one to check the connection, four for the Conversions API and six for audiences.

Area	Tool	Purpose	Permission	Changes data
Connection	<code>partner_whoami</code>	Advertiser, permissions, expiry and tools of the token	any	no
Conversions API	<code>partner_list_datasets</code>	List conversion datasets with status and counts	read	no
Conversions API	<code>partner_create_dataset</code>	Create a dataset (one per conversion source)	datasets:manage	yes
Conversions API	<code>partner_send_events</code>	Send conversion events	events:write	yes
Conversions API	<code>partner_get_reject_log</code>	Recent calls to a dataset and why events were rejected	read	no
Audiences	<code>partner_list_audiences</code>	List audiences with status and counts	read	no
Audiences	<code>partner_get_audience</code>	One audience in detail	read	no
Audiences	<code>partner_create_audience</code>	Create an audience	audiences:write	yes
Audiences	<code>partner_add_members</code>	Add people to an audience	audiences:write	yes
Audiences	<code>partner_remove_members</code>	Remove people from an audience	audiences:write	yes
Audiences	<code>partner_get_upload_report</code>	Outcome of one add or remove	read	no

Your test token has `read`, `events:write` and `audiences:write`.

`partner_whoami`

Shows the token this connection uses: advertiser ID, permissions, expiry and the tools it can call. Takes no arguments. Use it first to confirm the connection works.

Conversions API

Conversions go to a **dataset**: one per conversion source, the way you would create one Meta pixel per source. Your test dataset is `ds_33659f9a2043b3b3a8e0e2ee` . The flow is: send events with `partner_send_events` , then check them with `partner_get_request_log` and `partner_list_datasets` .

partner_list_datasets

Lists the advertiser's conversion datasets with their status and counts. Takes no arguments. Test events are counted separately (`test_events_7d`) from live ones (`events_24h` , `events_7d`).

partner_create_dataset

Creates a conversion dataset and returns its ID (`ds_...`) for `partner_send_events` . Needs a token with the `datasets:manage` permission; your test token does not have it, because your test dataset already exists. Datasets have no test mode; the events sent to them do.

Argument	Required	Description
<code>name</code>	yes	Up to 255 characters, for example "Website purchases"

partner_send_events

Sends conversion events to a dataset.

Argument	Required	Description
<code>dataset_id</code>	yes	<code>ds_33659f9a2043b3b3a8e0e2ee</code>
<code>events</code>	yes	1 to 100 events (fields below)
<code>test_event_code</code>	no	A label for test events; any value keeps the batch in test mode
<code>live</code>	no	<code>true</code> for a live write. Leave out for test mode

Each event:

Field	Required	Rules
<code>event_name</code>	yes	A Meta standard event (<code>Purchase</code> , <code>Lead</code> , <code>CompleteRegistration</code> , <code>AddToCart</code> , <code>Subscribe</code> , ...) or your own name, up to 64 letters, digits, spaces, <code>_</code> , <code>.</code> or <code>-</code>
<code>event_time</code>	yes	Unix time in seconds . At most 62 days old, at most five minutes in the future
<code>action_source</code>	yes	<code>website</code> , <code>app</code> , <code>email</code> , <code>phone_call</code> , <code>chat</code> , <code>physical_store</code> , <code>system_generated</code> , <code>business_messaging</code> or <code>other</code>

Field	Required	Rules
<code>event_id</code>	strongly recommended	Your unique ID for the event, up to 128 characters; used to drop duplicates
<code>user_data</code>	yes	At least one of <code>em</code> (hashed email), <code>ph</code> (hashed phone), <code>external_id</code> (your ID). Each a string or a list
<code>custom_data.value</code>	<code>Purchase</code> ; optional elsewhere	The transaction's monetary value, a non-negative number
<code>custom_data.currency</code>	with <code>value</code>	Three-letter currency code such as <code>USD</code>
<code>custom_data.order_id</code>	no	Your order or transaction ID
<code>custom_data.content_ids</code>	no	Product IDs, up to 50
<code>custom_data.content_type</code>	no	For example <code>product</code>
<code>custom_data.num_items</code>	no	Number of items
<code>custom_data.predicted_ltv</code>	no	Predicted lifetime value of the customer

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

Example arguments:

```
{
  "dataset_id": "ds_33659f9a2043b3b3a8e0e2ee",
  "events": [{
    "event_name": "Purchase",
    "event_time": 1761000000,
    "event_id": "order-1001",
    "action_source": "website",
    "user_data": {"em": ["ff8d9819fc0e12bf0d24892e45987e249a28dce836a85cad60e28eaaa8c6d976"]},
    "custom_data": {"value": 49.99, "currency": "USD", "order_id": "order-1001"}
  }]
}
```

Use the current Unix time for `event_time` ; the value above is only an example.

Result `data` :

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

Valid events are stored even when others in the same call are rejected; `messages` lists each rejected event by position (`index` , from 0) with the reason. Sending the same event again reports `events_deduplicated: 1` instead of storing it twice.

`partner_get_request_log`

Recent calls to a dataset, newest first, including refused ones: status, counts, test flag and the reason for each rejected event.

Argument	Required	Description
<code>dataset_id</code>	yes	<code>ds_33659f9a2043b3b3a8e0e2ee</code>
<code>limit</code>	no	1 to 500, default 20

Audiences

An audience is a list of people you keep up to date: create it once with `partner_create_audience` , then add and remove people in batches with `partner_add_members` and `partner_remove_members` . Each batch is applied in the background; check it with `partner_get_upload_report` , and the audience as a whole with `partner_get_audience` .

`partner_list_audiences`

The advertiser's audiences with status, member counts per identifier type and test flag.

Argument	Required	Description
<code>include_archived</code>	no	<code>true</code> to include archived audiences

`partner_get_audience`

One audience: status, member counts per identifier type, which ad platforms it would upload to, and its last upload report.

Argument	Required	Description
<code>audience_id</code>	yes	<code>aud_...</code> from <code>partner_create_audience</code> or <code>partner_list_audiences</code>

Audience `status` is `building` (no members yet), `processing` , `ready` , `too_small` (under 100 identifiers), `error` or `archived` . Test audiences are usually `too_small` ; that is expected.

`partner_create_audience`

Creates an audience. Creating a name that already exists returns that audience instead of a second one.

Argument	Required	Description
name	yes	Up to 255 characters
purpose	yes	targeting (show ads to these people), suppression (never show ads to them) or roster (your customer list, for measurement). Cannot change later
id_type	no	Limit members to one identifier type: hem_sha256 , ip or maid
live	no	true for a live audience. Leave out for a test audience

A test audience only accepts test member writes and is never sent to an ad platform. It cannot be switched to live later; create a new live audience instead.

Example arguments:

```
{"name": "Test audience", "purpose": "targeting"}
```

Result data includes id (aud_...), status and is_test .

partner_add_members

Adds people to an audience. schema names the columns; data holds one row per person. A row may carry several identifiers; use an empty string for a column you don't have.

Argument	Required	Description
audience_id	yes	aud_...
schema	yes	Columns, any of EMAIL_SHA256 , IP , MAID_IDFA , MAID_AAID , EFFECTIVE_TIME
data	yes	1 to 1,000 rows, one value per column
upload_id	recommended	Your ID for this batch, up to 128 of A-Z a-z 0-9 . _ : - . Resending the same batch with it changes nothing
effective_time	no	ISO 8601 time with offset for every row; default now. Orders changes to the same person
live	no	true for a live write. Leave out for test mode

Column	Value
EMAIL_SHA256	SHA-256 of the trimmed, lowercased email
IP	A public IPv4 address, as is
MAID_IDFA , MAID_AAID	An Apple or Google mobile advertising ID (UUID), as is
EFFECTIVE_TIME	ISO 8601 with offset, per row

Example arguments:

```
{
  "audience_id": "aud...",
  "schema": ["EMAIL_SHA256", "IP", "MAID_AAID"],
  "data": [
    ["ff8d9819fc0e12bf0d24892e45987e249a28dce836a85cad60e28eaaa8c6d976", "", ""],
    ["", "8.8.8.8", ""],
    ["", "", "38400000-8cf0-11bd-b23e-10b96e40000d"]
  ],
  "upload_id": "upload-2026-10-20-1"
}
```

Result data :

```
{"upload_id": "upload-2026-10-20-1", "status": "processing", "rows_received": 3,
  "rows_accepted": 3, "rows_rejected": 0, "rejections": []}
```

The batch is applied in the background, usually within one to two minutes; read the outcome with `partner_get_upload_report` . Rejected rows are listed by `index` with a reason, never with the identifier itself.

`partner_remove_members`

Removes people from an audience. Same arguments and rules as `partner_add_members` ; use a new `upload_id` .

`partner_get_upload_report`

The outcome of one add or remove batch.

Argument	Required	Description
<code>audience_id</code>	yes	<code>aud...</code>
<code>upload_id</code>	yes	The <code>upload_id</code> of the batch

Result data :

```
{"upload_id": "upload-2026-10-20-1", "op": "add", "status": "applied", "rows_accepted": 3,
  "rows_rejected": 0, "membership": {"added": 3, "removed": 0, "unchanged": 0},
  "active": {"hem_sha256": 1, "ip": 1, "maid": 1}}
```

`status` is `processing` until the batch is applied, then `applied` .

Not available over MCP

These are in the REST API (see `README.md` and `partner-guide.md`), not in the MCP server:

- **Conversions:** reading back the stored events themselves. Over MCP you see what each call stored and rejected (`partner_get_request_log`) and the dataset's counts (`partner_list_datasets`); the events are also visible to us in the platform.
- **Audiences:** sending a complete list that replaces the audience's members (`members/replace`), renaming an audience, and archiving one.
- **Plaintext emails and phone numbers**, and batches larger than 100 events or 1,000 rows.

Results and errors

Every tool returns one text item containing a JSON object. Success:

```
{"ok": true, "status": 200, "trace_id": "eac37e34c3a13dcf", "data": {...}, "mode": "test"}
```

Failure:

```
{"ok": false, "status": 404, "trace_id": "5b32bdc5388a1728",  
  "error": {"code": "audience_not_found", "message": "No audience with this id for the token's advertiser."}}
```

Check `ok` rather than the MCP-level error flag: a refused request is still a normal tool result. `mode` appears on writes. Some failures include `details` with per-item reasons, for example every event rejected:

```
{"ok": false, "status": 400, "error": {"code": "all_events_rejected", "message": "Every event was rejected; see messages."},  
  "details": {"events_received": 1, "events_stored": 0, "events_rejected": 1,  
    "messages": [{"index": 0, "error": "event_time is older than 62 days"}], "is_test": true},  
  "mode": "test"}
```

Code	Meaning
<code>token_invalid</code> , <code>token_expired</code> , <code>token_revoked</code>	The token is wrong, expired or revoked. Ask us for a new one
<code>plaintext_not_allowed_over_mcp</code>	An email or phone was not a 64-character SHA-256 hash
<code>dataset_not_found</code> , <code>audience_not_found</code> , <code>report_not_found</code>	The ID is wrong or belongs to another advertiser

Code	Meaning
<code>all_events_rejected</code> , <code>all_rows_rejected</code>	Nothing in the call was valid; see <code>details</code> for each <code>index</code>
<code>schema_invalid</code>	<code>schema</code> is empty, repeats a column or uses an unknown one
<code>upload_id_conflict</code>	That <code>upload_id</code> was already used for a different batch
<code>sandbox_mismatch</code>	A test write was sent to a live audience, or the other way round
<code>rate_limited</code>	Too many calls; wait <code>retry_after</code> seconds and repeat the same call

Limits

Limits are per token and shared with any REST API calls made with the same token.

Limit	Value
<code>partner_send_events</code> calls	100 per second
<code>partner_add_members</code> / <code>partner_remove_members</code> calls	20 per second
All calls combined	100 per second
Short bursts	Up to 200 calls above the rate (40 for member calls), then refused
Events per <code>partner_send_events</code> call	100
Rows per <code>partner_add_members</code> / <code>partner_remove_members</code> call	1,000

A refused call returns `{"ok": false, "status": 429, "error": {"code": "rate_limited", ...}, "retry_after": 1}`.

Common reasons for a rejected audience row: `invalid_hem` (not 64 hex characters), `hash_of_empty_string`, `invalid_ip`, `non_public_ip`, `ipv6_not_supported`, `invalid_maid`, `zero_maid`.

When you contact us about a result, include its `trace_id` (or `upload_id`) and the time. Never send raw emails or phone numbers.

Test checklist

Run these calls in order. When all of them behave as described, the test is complete; tell your Neural Ads contact and mention your audience name.

Connection

1. `partner_whoami` : returns advertiser `cb66550d-cca6-4c08-b0cb-d3bad004265b` and lists 10 tools.

Conversions API

2. `partner_list_datasets` : lists `ds_33659f9a2043b3b3a8e0e2ee` .
3. `partner_send_events` with one event (current time, a unique `event_id`) : `events_stored: 1` , `mode: test` .
4. The same call again: `events_deduplicated: 1` .
5. `partner_send_events` with one event whose `event_time` is 70 days old: `all_events_rejected` , and `details.messages` gives the reason.
6. `partner_get_request_log` for `ds_33659f9a2043b3b3a8e0e2ee` : all three calls listed, the last one refused.
7. `partner_list_datasets` again: `test_events_7d` has gone up by one.

Audiences

8. `partner_create_audience` with a new name, purpose `targeting` : returns an `aud_...` ID with `is_test: true` .
9. `partner_add_members` with a few rows and an `upload_id` : all rows accepted.
10. `partner_get_upload_report` for that `upload_id` : `status: applied` .
11. `partner_remove_members` with one of those rows and a new `upload_id` : `accepted` .
12. `partner_get_audience` : the member count dropped by one.

Moving to production

When your test passes, we give you a production token for your own advertiser. Then:

	Test	Production
Server URL	<code>https://stage.neuralads.ai/partner/mcp</code>	<code>https://platform.neuralads.ai/partner/mcp</code>
Token	the one above	a new token for your account
Advertiser	Neural Ads test account	your advertiser
Dataset ID	<code>ds_33659f9a2043b3b3a8e0e2ee</code>	your production dataset ID

Writes still run in test mode unless a call passes `"live": true` . Send production events live, and create production audiences live; test audiences cannot be switched to live.