Request API: JSON endpoint

The JSON endpoint is the foundation of every display integration with Adhese. It gives you direct access to the ad server and Gateway without an SDK or library: you build the request, and Adhese returns the ads for your slots as JSON. Any environment that can send an HTTPS request and parse JSON can use it — a backend service, a CMS, a mobile app, ...

The Typescript Web SDK and the Prebid adapter use this same endpoint under the hood. If you use one of those, you do not need this page for implementation, but it remains the reference for what actually happens under the hood.

The API is public and requires no authentication and **cookies are not required** for an integration to work.

Endpoint URL

Each Adhese account has its own endpoint. The default pattern is:

```
https://ads-[customer].adhese.com/json/
```

Making a POST request

Send a **POST request over HTTPS** with a JSON body (structure below). This is the recommended method for all new integrations.

> The endpoint also accepts **GET requests,** where the slots and parameters are encoded in the URL path instead of a body. We prefer the POST request but documentation can be retrieved

Request body

The POST body is a JSON object with up to three top-level properties:

```
{
    "slots": [
        {
            "slotname": "some_slot"
        },
        ...
    ],
    "parameters": {
        "kw": [
        "cheese",
        "wine"
        ],
        ...
    },
    "user": {
      "ext": {
        "eids": []
      }
    }
}
```

Property	Required	Purpose
`slots`	**Yes**	The ad placements you are requesting
`parameters`	No	Request-level targeting data (page, user, context)
`user`	No	External user IDs for downstream buyers (open market)

slots

An array of slot objects — one per ad placement needed for the current page or application state. **At least one slot is required.**

Each slot object contains at minimum a `slotname`: the unique identifier of the placement as configured in your Adhese account. Campaigns are targeted against these names.

```
"slots": [
    { "slotname": "some_slot" },
    { "slotname": "another_slot" }
]
```

> ⚠ Each `slotname` may appear only once per request. Duplicates cause the request to be rejected with status `442`.

parameters

An object of targeting attributes describing the page, user, or context. Keys are **fixed two-character codes** defined in your Adhese account; values are **always arrays of strings**.

All parameters are optional — omit a key entirely, or send an empty array, when there is nothing to pass.

```
"parameters": {
    "kw": ["cheese", "wine"],
    "mi": ["ABCDEF123456"]
}
```

In this example, `kw` carries search keywords and `mi` a member ID. Reserved parameter codes are listed on Request target parameters.

Slot-level parameters

A slot object can carry its own `parameters` property with the same structure. For each slot, request-level and slot-level parameters are **merged**. Use this for attributes that differ per placement, such as position on the page:

```
"slots": [
    {
        "slotname": "some_slot",
        "parameters": { "ps": ["top"] }
    },
    {
        "slotname": "another_slot",
        "parameters": { "ps": ["bottom"] }
    }
]
```

user

Reserved for passing identifiers from external ID providers to buyers further down the chain, mainly in an open-market context:

```
"user": {
    "ext": {
        "eids": []
    }
}
```

If you think you need this, contact Adhese Support before implementing.

Complete request example

```
{
    "slots": [
        { "slotname": "homepage_banner" },
        { "slotname": "homepage_rectangle_top" },
        { "slotname": "homepage_rectangle_bottom" },
        {
            "slotname": "homepage_halfpage",
            "parameters": { "ps": ["right"] }
        }
    ],
    "parameters": {
        "id": ["1234567890ABCD"],
        "ct": ["computers", "laptops"],
        "kw": ["cheese", "wine"]
    }
}
```

Here `id` is a user ID, `ct` product categories, and `kw` search keywords.

Making a GET request

The GET version of the ad request contains the same information as the POST version, **slot level parameters are however not supported.**

Example request

```
https://ads-[account].adhese.com/json/sl_sdk_example_-
leaderboard/ctsports;soccer?t=244.18664863333106
```

Section	Meaning	Details
https://ads-[account].adhese.com	Ads domain	The domain is either the default ads domain for your account or a configured first-party domain. More info on first-party domains can be found here.

Section	Meaning	Details
/json/	Endpoint identifier	
/sl[location code]-[format code]/	Slots	This section can be added more than once. Each placement starts with the prefix 'sl', followed by the full slot code.
/ctsports;soccer/	Target(s)	This section can be added more than once. Each target section starts with its predefined prefix and is followed by either one value or a list of values separated by ';'.
?t=[timestamp]	Timestamp	A timestamp to avoid caching issues

Response codes

Code	Meaning	Details
`200`	OK	Body contains an array of ads. If no ads are available, the array is empty — an empty array is a **successful** response, not an error.
`442`	Duplicate slots	One or more `slotname` values appear more than once. Header `x-adhese-bad-request` lists the duplicates.
`454`	No slots	The request body contains no slots. Header `x-adhese-bad-request` contains `slots cannot be empty`.
`500`	Internal server error	If this was a debug request, check the debug log; otherwise contact Adhese Support.

When handling errors programmatically, read the `x-adhese-bad-request` response header for the specific message.

Response object

A `200` response contains a JSON array with one object per delivered ad. The objects contain many attributes; the complete field reference lives at General JSON response structure *(editor note: previously linked to the GitHub SDK readme — decide on a single canonical reference)*.

For a server-side or custom display integration, these are the attributes you need:

Attribute	Description
`slotName`	The slot this ad belongs to. Use it as the key to match responses to the slots in your request when requesting multiple slots at once.
`tag`	The ad markup, returned as a string. For display ads this is typically an HTML fragment to insert into the slot's container. For video/audio it is a VAST-compliant XML document; for native ads it is a JSON object (delivered as a string that your application must parse).
`width`	Width of the ad container in pixels, required for correct display.
`height`	Height of the ad container in pixels, required for correct display.
`trackedImpressionCounter`	Unique URL to call when the ad is **added to the page**, even if not yet visible. Registers an IAB *paid impression*. Fire-and-forget: the call can be asynchronous and the response ignored.
`viewableImpressionCounter`	Unique URL to call when the ad has been **at least 50% in the viewport for at least one second**. Registers an IAB *viewable impression*. Fire-and-forget.
`clickTag`	Unique URL that counts a click and redirects the user to the ad's landing page. Use it as the href wrapping the creative. In applications without links, the URL can be called directly to register the click, ignoring the response.

Trimmed response example

```
[
    {
        "slotName": "homepage_banner",
        "adFormat": "714x224",
        "width": "714",
        "height": "224",
        "tag": "<div>…ad markup…</div>",
        "trackedImpressionCounter": "https://ads-{customer}.adhese.com/track/…",
        "viewableImpressionCounter": "https://ads-{customer}.adhese.com/track/…-
Adhese_IABview/…",
        "clickTag": "https://ads-{customer}.adhese.com/raylene/…/UR",
        "orderName": "Example Campaign",
        "creativeName": "Example Creative",
        "extension": {
            "mediaType": "banner",
            "prebid": {
                "cpm": { "amount": "13.047", "currency": "EUR" }
            }
        }
```

```
    }
]
```

This example is abbreviated for readability. Responses contain many additional fields — see the full field reference linked above.

Rendering and tracking workflow

For each ad in the response, a correct display integration does the following, in order:

1. **Match** the ad to its container using `slotName`.
2. **Render** the `tag` markup inside a container sized by `width` × `height`.
3. **Call** `trackedImpressionCounter` the moment the ad is added to the page.
4. **Observe** viewability (e.g. with an Intersection Observer) and call `viewableImpressionCounter` once the ad has been ≥50% in view for ≥1 second.
5. **Wrap** the creative's landing-page link in the `clickTag` URL so clicks are counted and redirected.

Skipping step 3 or 4 - or executing them at the wrong time - is the most common cause of reporting discrepancies between Adhese and third-party measurement.

Event tracking

Beyond impressions and clicks, you can register **custom events** (e.g. video quartiles, expansions, interactions) by constructing tracking URLs from the response data.

Building a custom event tracking URL

```
https://[ad_request_host]/track-
[your_event_string]/[response.id]/sl[response.slotID]/II[response.impressionID]
```

- `your_event_string` — a name of your choosing for the event; it will appear in reporting
- `response.id`, `response.slotID`, `response.impressionID` — taken from the ad's response object

Revision #12
Created 6 July 2026 13:58:30 by Kim Van Crombrugge
Updated 6 July 2026 15:14:49 by Kim Van Crombrugge