Ad Delivery: Integration methods

In this chapter, you will find the different ways to integrate with the **Adhese ad server** and **Gateway**, which will allow you to request and display ads across channels.

Both components are **integrated in the same way**. The difference lies in the internal configuration and demand orchestration - not in the way requests are made.

All integrations are secure by design and aligned with Adhese's **privacy-first** approach. The platform operates **without third-party cookies** and relies on server-side processing of first-party data to deliver targeted advertising.

- Typescript Web SDK
- Prebid
- Request API: Headsup endpoint
- Request API: Stack endpoint
- Request API: JSON endpoint

Typescript Web SDK

The Adhese TypeScript SDK simplifies ad integration across web environments by handling the underlying request and response logic, allowing you to implement ad placements without dealing with low-level API calls.

At the same time, the SDK remains flexible: it allows you to plug in custom logic where needed, giving you full control for more advanced or tailored integrations.

The getting started guide walks you through:

- Installing the SDK
- Initialising a client
- Making ad requests
- Rendering ad responses

> The full documentation can be found on https://adhese.github.io/sdk_typescript/

Integration options

The SDK supports multiple integration approaches, depending on your environment:

NPM

Install the SDK via NPM and integrate it into modern JavaScript or TypeScript projects.

React

Use the SDK within React applications to manage ad slots and lifecycle in a component-based way.

Standalone JS script

Include the SDK as a standalone script for simple integrations without a build step. Ideal for quick setups or environments where bundling is not available.

> The standalone script could be hosted and maintained by the Adhese team. Contact support if you wish to discuss this option in more detail.

Key Features

Slot Management

Slots are the fundamental units for ad placement. The SDK supports three slot registration methods:

- **Automatic DOM detection** - scan for elements with `adunit` class
- **Pre-initialization slots** - declare slots before page load for early ad fetching
- **Manual registration** - dynamically add slots via API

Additional slot capabilities include:

- Device-specific format selection (responsive ads via media queries)
- Lazy loading with viewport detection
- Dual render modes (iframe or inline)
- Custom setup hooks for advanced control

Configuration & Parameters

At initialization, you can set:

- Account ID (required)
- Targeting parameters (2-character prefixes matching your adserver configuration)
- URL and referrer logging preferences
- Lazy loading settings
- Device type detection rules (customizable media queries)

Event System

Subscribe to lifecycle events:

- Slot lifecycle (add, remove, dispose)
- Request/response events
- Consent & parameter changes
- Debug mode toggles
- Impressions & viewable tracking

Consent Management

Supports regulatory compliance via:

- Binary consent (yes/no)
- TCF API v2 integration

React Integration

Dedicated React SDK provides:

- `AdheseProvider` context wrapper
- `useAdhese` hook for instance access
- `useAdheseSlot` hook for slot creation
- `AdheseSlot` component with JSX rendering support
- Placeholder support during ad loading

SDK Repository

The SDK is actively maintained on GitHub and publicly available:

https://github.com/adhese/sdk_typescript

Prebid

Prebid is a free and fully open source header bidding solution available to any publisher, dedicated to header bidding in the ad tech industry.

Header bidding allows publishers to create a short delay in ad serving to obtain bids from multiple demand partners before deciding on which ad to display, enabling competitive pricing and higher revenue.

> More information on Prebid | Prebid.js Bidder | Prebid Server Bidder

How do Adhese and prebid work together

When a publisher implements Adhese through Prebid, they're participating in a **header bidding auction** where multiple demand partners (including Adhese) compete to deliver ads. Here's how the process works:

Auction Initiation

When a page loads, Prebid sends simultaneous bid requests to multiple bidders - including Adhese - rather than sequential requests. This concurrent approach is one of Prebid's key features for managing latency. All bidders (including Adhese) receive their request at roughly the same time.

Adhese's Role in the Auction

Adhese acts as a **bidder/SSP** within Prebid's ecosystem. Through the Prebid adapter, Adhese receives ad requests and returns competitive bids and creatives. The adapter translates Prebid's standardized request format into Adhese's API requirements and vice versa.

Bid Submission & Timeout Management

Publishers set a timeout value (typically 1-3 seconds) that determines how long Prebid waits for responses. Adhese must return its bid within this window. If Adhese doesn't respond in time, Prebid excludes it from that auction round. This timeout mechanism prevents slow bidders from degrading page performance.

Ad Selection

After all bidders respond, Prebid passes the winning bid to the publisher's ad server. If Adhese wins, its creative is selected and delivered, leading to an impression in Adhese.

Targeting & Parameters

Both Prebid and Adhese support custom targeting. Publishers can pass first-party data (like content categories, user segments, or location) through both systems, giving Adhese context for more accurate bidding.

Impression Tracking

After an ad is selected and rendered, Adhese can track impressions and viewability through Prebid's event system, feeding performance data back to their platform.

Bid Configuration Params

Pbjs param name	Scope	Description	Example	Type

Request API: Headsup endpoint

The headsup endpoint returns **all** the materials for the campaigns running on the requested DOOH position.
It can be used by the DOOH screens to **retrieve and cache all materials** at the beginning of the day to avoid delays by having to download materials during the day.

> To ensure that the assets are included in the response, the *Use in heads-up file* setting **must be enabled at format level**. This action can be performed by all admin users in the Adhese UI.

```
https://headsup-[customer].adhese.org/api/headsup/download-list/sl[positioncode]
```

Parameters	Value	Required
customer	The name of your Adhese account	Yes
position code	The code of the position you wish to request: [location code][optional position code]-[format code]. This value is always prefixed by 'sl'.	Yes

Response

The JSON response consists of a "media" array that may be empty or populated with multiple ad objects, depending on the number of active campaigns. **The structure of an ad object is fixed and can not be customized.**

The ID and URL from which to download the file is identical and is also available in the stack endpoint that will be used to download DOOH playlists during the day.

Example


```
{
    "media":[
        {
            "ad":{
                "id":"https://pool-demo.adhese.com/pool/lib/562_2nd_1.mp4",
                "mime":"video/mp4",
                "curl":"https://pool-demo.adhese.com/pool/lib/562_2nd_1.mp4",
                "filesize": 15470592,
```

```
                "checksum":"071f717962db99cc137d138696d33209f2e4818d42a54f70dfa6606eeb1b640b"
            }
        },
        {
            "ad":{
                "id":"https://pool-demo.adhese.com/pool/lib/560_2nd_1.mp4",
                "mime":"video/mp4",
                "curl":"https://pool-demo.adhese.com/pool/lib/560_2nd_1.mp4",
                "filesize": 15470592,
                "checksum":"4e78745a33518ff22c1c9f39852a49c1c0fadd0adf722e3394ad1215d5e5ff5b"
            }
        },
        {
            "ad":{
                "id":"https://pool-demo.adhese.com/pool/lib/561_2nd_1.mp4",
                "mime":"video/mp4",
                "curl":"https://pool-demo.adhese.com/pool/lib/561_2nd_1.mp4",
                "filesize": 6582272,
                "checksum":"64d2eab6d51b208c0cec3fc4d64c53381e5d41ae2a1d4c5b7704b1818bdb6158"
            }
        }
    ]
}
```

Request API: Stack endpoint

The stack endpoint allows you to request **one position** for which Adhese will return a **list of ads**. There are 2 versions:

1. /m/stack/: returns a limited amount of ads
2. /e/stack/: returns an unlimited[1] amount of ads[2]

> **1** The adserver rules still apply and will influence which campaigns are part of the response. Not all booked campaigns are necessarily part of the returned array.

> **2** The **/e/stack/** endpoint must be enabled by Adhese support before it becomes available

m/stack endpoint

```
https://ads-[customer].adhese.com/m/stack/sl[positioncode]/[target prefix][target value]?max_ads=[amount]
```

Parameters	Value	Required
customer	The name of your Adhese account	Yes
position code	The code of the position you wish to request is [location code]-[format code]. This value is always prefixed by 'sl'.	Yes
custom target	Similar to the basic JSON GET endpoint, the stack request can contain target information by adding a prefix followed by a value. More values can be separated with a ;	No
max_ads[1]	The maximum number of ads you wish to request	Yes

> **1** A configuration on the adserver can be activated to ensure the max_ads value never exceeds a specific limit. When the parameter is left empty, the configured limit will be used instead.

e/stack endpoint

```
https://ads-[customer].adhese.com/e/stack/sl[positioncode]/[target prefix][target value]
```

Parameters	Value	Required
customer	The name of your Adhese account	Yes
position code	The code of the position you wish to request is [location code]-[format code]. This value is always prefixed by 'sl'.	Yes
custom target	Similar to the basic JSON GET endpoint, the stack request can contain target information by adding a prefix followed by a value. More values can be separated with a ;	No

Stack response

Returned code

The response contains JSON code: an **ads** array that will either be empty or populated with one or more objects, depending on the number of active campaigns.

The structure of the objects within the array is determined by the advar template used when setting up the creatives.

An example response used in a DOOH setup


```
{
    "ads":[
        {
            "id":"https://pool-demo.adhese.com/pool/lib/562_2nd_1.mp4",
            "dur":28.07,
            "prio":15,
            "booking_prio":0,
```

```
            "publisher":"1",
            "key":"",
            "proofOfPlay":"https://ads-
demo.adhese.com/track/3474/sl357/tlnone/piplayer_id/A2?1746011926824",
            "error":"https://ads-demo.adhese.com/track/3474-
PLAY_ERROR_[ERRORCODE]/sl357/tlnone/piplayer_id/A2/?1746011926824"
        },
        {
            "id":"https://pool-demo.adhese.com/pool/lib/561_2nd_1.mp4",
            "dur":11.45,
            "prio":15,
            "booking_prio":0,
            "publisher":"1",
            "key":"",
            "proofOfPlay":"https://ads-
demo.adhese.com/track/3444/sl357/tlnone/piplayer_id/A2?1746011926824",
            "error":"https://ads-demo.adhese.com/track/3444-
PLAY_ERROR_[ERRORCODE]/sl357/tlnone/piplayer_id/A2/?1746011926824"
        },
        {
            "id":"https://pool-demo.adhese.com/pool/lib/560_2nd_1.mp4",
            "dur":18.85,
            "prio":15,
            "booking_prio":0,
            "publisher":"1",
            "key":"",
            "proofOfPlay":"https://ads-
demo.adhese.com/track/3455/sl357/tlnone/piplayer_id/A2?1746011926824",
            "error":"https://ads-demo.adhese.com/track/3455-
PLAY_ERROR_[ERRORCODE]/sl357/tlnone/piplayer_id/A2/?1746011926824"
        }
    ]
}
```


Sorting the returned list

The order in which the ads array is populated can be be configured by using the **sequence** property in the advar template.

As value you can use a number that is filled in through the advar form, or use one of the Adhese macro's that return an ID.

When the **sequence** property is added and given a value, the adserver will sort the available ads based on its value. In the example below we use the CREATIVE ID to sort the array **from lowest ID to highest ID**.

```
"sequence": <ADHESE_LIB_ID>
```

> The sequence property is only used by the adserver and will be removed before theresponse is returned

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 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.

POST request

Host

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

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

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.

GET request

The GET version of the ad request contains the same information as the POST version with one limitation:
slot level parameters are not supported.

Example request

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

Section	Description
https://ads-[account].adhese.com	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.
/json/	
/sl[location code]-[format code]/	This section can be added more than once. Each placement starts with the prefix 'sl', followed by the full slot code.
/ctsports;soccer/	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]	A timestamp to avoid caching issues

Response codes

Code	Name	Description
`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.

Code	Name	Description
`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 successful 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.

The most important attributes required for custom integrations are:

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.

Attribute	Description
`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" }
            }
        }
    }
]
```

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 domain]/track-[event
label]/[response.id]/sl[response.slotID]/II[response.impressionID]
```

- `event label` : 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