ID5 User ID Module for Prebid Server Java

10/15/2026 6:44 am EDT

Overview

The ID5 User ID module enriches the OpenRTB auction requests your Prebid Server sends to bidders with the ID5 universal identifier. During each auction, it fetches an ID5 ID and injects it into the bid request as an Extended Identifier (EID), improving addressability whilst honoring user privacy choices.

- Fetches the ID5 universal identifier and injects it into bid requests as an EID (`user.eids`).
- Skips fetching and injection when an ID5 ID is already present in the request
- Honors privacy signals (GDPR/TCF, CCPA/US Privacy, COPPA, GPP).
- Lets you control rollout: filter by account, country, or bidder, and sample a percentage of traffic.
- Runs asynchronously and non-blocking - a failed or slow ID5 fetch never blocks the auction.
- Processes both web and mobile in-app traffic, including requests sent to your Prebid Server by the Prebid Mobile SDK.

Prerequisites

Register with ID5

The ID5 ID is free to use, but requires a registration with us.

Deploying for yourself: If you're integrating ID5 as part of your own Prebid Server setup, visit our website to sign up and request your **ID5 Partner ID**.

Deploying on behalf of clients: If you're integrating ID5 as part of a Prebid Server setup for your clients, each client must sign our ID5 agreement, and you must include their Partner ID in any requests you send us from their media property. If it's useful, ID5 can set up a co-branded sign-up page for your clients.

To deploy on behalf of your clients, you need to declare yourself as the provider name — please check with ID5 for the exact value to use in the `provider-name` field.

Add the Dependency

The module is published to **Maven Central**. Add it to the Maven build that assembles your Prebid Server distribution (the module/POM that produces your runnable PBS JAR), so the module is on the server's runtime classpath:

```
<dependency>
  <groupId>io.id5.prebid.server.hooks.modules</groupId>
  <artifactId>id5-user-id</artifactId>
  <version>3.43-rc.1</version>
</dependency>
```

Contact the ID5 team for advice on which version to use. In general, the module version tracks the Prebid Server version line it is built against, in the form `<PBS-major>.<PBS-minor>-<release>` . In Example:

Your Prebid Server Version	Module Version
3.43.x	`3.43-rc.1`

Browse all available versions on Maven Central

Enable and Configure the Module

Add the module configuration to your Prebid Server configuration file and configure hooks

Minimal Setup

```
hooks:
  id5-user-id:
    enabled: true
    provider-name: "my-pbs-host"                    # identifies who operates this PBS instance
    fetch-endpoint: "https://api.id5-sync.com/gs/v2"  # ID5 API endpoint
    partner: 173                                    # your ID5 Partner ID - see "Partner ID" below
```

Partner ID

> The Partner ID argument given to the **partner** parameter must be decided together with the ID5 team

The right choice depends on your specific case - it may be the host's own Partner ID, or, in some cases, the publishers' Partner IDs. The ID5 team will work with you to determine the appropriate Partner ID strategy and implement it according to your case and needs.

There are two ways to supply the Partner ID, matching the two common strategies:

1. Static Partner ID (default). Set the `partner` property (see the Configuration Reference below). The same Partner ID is used for every request. This is the simplest option when a single host-level Partner ID fits.

2. Dynamic Partner ID. When the Partner ID must vary per request - for example, derived from the account, channel, or publisher the auction comes from - implement the `Id5PartnerIdProvider` interface and register it as a Spring bean. The module picks it up automatically.

```
package com.mycompany.pbs;

import org.prebid.server.auction.model.AuctionContext;
import org.prebid.server.hooks.modules.id5.userid.v1.model.Id5PartnerIdProvider;
import org.springframework.stereotype.Component;

import java.util.Optional;

@Component
public class MyId5PartnerIdProvider implements Id5PartnerIdProvider {

    @Override
    public Optional<Long> getPartnerId(AuctionContext auctionContext) {
        // Derive the Partner ID from the auction context
        // (account, bid request, privacy data, ...).
        // Return Optional.empty() to skip the ID5 fetch for this request.
        return Optional.of(1234L);
    }
}
```

When you use a dynamic provider:

- `getPartnerId` receives the full `AuctionContext` (account, bid request, privacy information), so you can select the Partner ID per request.
- Returning `Optional.empty()` skips the ID5 fetch for that request.
  - **Omit the `partner` property.** The built-in constant provider is only created when `partner` is set, so leaving it out avoids having two competing `Id5PartnerIdProvider` beans. (Alternatively, mark your bean `@Primary`.)

Configuration Reference

Property	Required	Type	Default	Description
`enabled`	Yes	boolean	-	Must be `true` to activate the module.
`provider-name`	Yes	string	-	Identifier sent to the ID5 API describing who operates this Prebid Server instance (e.g. `"my-company-pbs"`).
`fetch-endpoint`	Yes	string	-	ID5 API endpoint URL: `https://api.id5-sync.com/gs/v2`.
`partner`	Conditional	long	-	Your ID5 Partner ID, for a single static Partner ID. Required unless you provide a custom `Id5PartnerIdProvider` - see Partner ID above.
`inserter-name`	No	string	*none*	Canonical domain of the entity adding the EID (e.g. `"my-company.com"`). Written to the EID's `inserter` field.
`fetch-sampling-rate`	No	double	`1.0`	Fraction of requests to process, `0.0` - `1.0` (e.g. `0.1` = 10%). Useful for gradual rollout.
`account-filter`	No	filter	*none*	Limit which accounts trigger an ID5 fetch.
`country-filter`	No	filter	*none*	Limit which countries trigger an ID5 fetch.
`bidder-filter`	No	filter	*none*	Limit which bidders receive the ID5 EID.

Filters

Each filter takes an `exclude` flag and a list of `values`:

```
<filter-name>:
  exclude: false     # false = allowlist (only these), true = blocklist (all except these)
  values:
    - value1
    - value2
```

- `exclude: false` → **allowlist**: only matching requests proceed. Empty list = allow all.
- `exclude: true` → **blocklist**: matching requests are skipped. Empty list = allow all.

Full Configuration Example

```
hooks:
  id5-user-id:
    enabled: true
    provider-name: "my-pbs-host"
    fetch-endpoint: "https://api.id5-sync.com/gs/v2"
    partner: 1234                 # your ID5 Partner ID - see "Partner ID" above
    inserter-name: "my-company.com"
    fetch-sampling-rate: 0.8
    account-filter:               # all accounts except "test-account"
      exclude: true
      values: [test-account]
    country-filter:               # only these countries
      exclude: false
      values: [US, GB, DE, FR]
    bidder-filter:                # only these bidders receive the ID5 EID
      exclude: false
      values: [rubicon, appnexus, pubmatic]
```

Register the Module's Hooks

The module runs through two hooks that you must register in a Prebid Server **execution plan**:

- `id5-user-id-fetch-hook` - stage `processed-auction-request`
- `id5-user-id-inject-hook` - stage `bidder-request`

You can register them globally (host execution plan) or per account. Per-account example:

```
accounts:
  - id: "1001"
    status: active
    hooks:
      execution-plan:
        {
          "endpoints": {
            "/openrtb2/auction": {
              "stages": {
                "processed-auction-request": {
                  "groups": [
                    {
                      "hook-sequence": [
                        { "module-code": "id5-user-id", "hook-impl-code": "id5-user-id-fetch-hook" }
                      ]
                    }
                  ]
                },
                "bidder-request": {
                  "groups": [
                    {
                      "hook-sequence": [
                        { "module-code": "id5-user-id", "hook-impl-code": "id5-user-id-inject-hook" }
                      ]
                    }
                  ]
                }
              }
            }
          }
        }
```

What Gets Added to the Bid Request

When an ID5 ID is fetched, the module adds an EID to the `user.eids` array of requests sent to the selected bidders.

Example EID added:

```
{
 "user": {
  "eids": [
   {
    "source": "id5-sync.com",
    "uids": [
     {
      "id": "ID5*YsvxY...",
      "atype": 1,
      "ext": { "linkType": 2, "pba": "jWwv+..." }
     }
    ],
    "inserter": "my-company.com"
   }
  ]
 }
}
```

Privacy & Data Sharing

The module forwards available privacy signals to the ID5 API, which honors them when generating an identifier:

- **GDPR / TCF** - consent string and GDPR-applies flag
- **CCPA** - US Privacy string
- **COPPA** - COPPA flag
- **GPP** - GPP string and applicable section IDs

To resolve an ID5 ID, the module may send request and device signals (such as site domain/referrer domain, app bundle, device IP, user agent, and mobile advertising ID) to the ID5 API.

> Ensure that your or your client's privacy policy and consent setup cover data sharing with ID5.

Verify the Integration

1. Confirm the dependency is on your server's classpath and the module is `enabled: true`.
2. Confirm the execution plan registers both hooks.
3. Send an auction request and confirm that an `id5-sync.com` EID appears in `user.eids` of the bidder requests (for the bidders allowed by your filters).

If no EID appears, check that the required properties are set, the execution plan is applied to the account/endpoint in use, and no filter (account, country, bidder, or sampling) is excluding the request.

Support

For help with the module or the ID5 service, contact **support@id5.io**.