Mobile In-App Integration - Client-to-Server

09/23/2026 10:49 am EDT

> **Beta:** The client-to-server integration and the ID5 Mobile SDK are currently in beta. APIs and behaviour may still change. We actively welcome your feedback - if you run into any issues, reach out at `support@id5.io,` and we'll address them quickly. Adapters for Prebid and Google Ad Manager will be released in the near future.

The **client-to-server** integration is an alternative to the server-to-server method for publishers who prefer not to operate backend infrastructure to call ID5. Instead of building and sending requests from your own servers, you embed the **ID5 Mobile SDK** directly in your app. The SDK handles the entire identity workflow on your behalf - collecting signals, enforcing consent, building and signing the request, caching the response and signature on the device, and delivering the encrypted ID5 ID to your app.

This approach lets you integrate ID5 with no server-side development or maintenance. The following guide walks through the integration step by step.

1. Register with ID5

You need an ID5 **partner ID** to integrate. If you don't have one yet, contact ID5 or email `contact@id5.io`.

> **NOTICE: TERMS OF SERVICE & BINDING AGREEMENT** - Use of the ID5 SDK and ID5 identifiers is subject to your ID5 partner agreement and applicable data-protection obligations. Ensure you have a valid legal basis and the necessary consent before collecting and sharing user signals.

2. Add the ID5 Mobile SDK

Android

Add the SDK dependency to your Gradle build:

```
implementation("io.id5:id5-sdk-core:0.1.0")
```

Full installation, configuration, and API reference are maintained in the repository:
github.com/id5io/id5-sdk-android

iOS

Release soon. The iOS Mobile SDK is in progress. This section will be updated with installation and usage details when it ships.

3. Configure consent

If you operate under GDPR, initialize your **CMP** (Consent Management Platform) *before* the SDK. The SDK reads the IAB TCF snapshot and waits for a complete consent signal before fetching.

- Under GDPR, both **Purpose 1** and **vendor 131 (ID5)** consent are required for the SDK to read the device cache and the advertising ID.
- On **Android 13+**, declare `com.google.android.gms.permission.AD_ID` in your manifest to allow

advertising-ID lookup.
- On **iOS**, App Tracking Transparency (ATT) authorization is also required to read the IDFA. *(Applies when the iOS SDK ships.)*

4. Optimize SDK API Configuration

To maximise addressability and produce the highest-quality ID5 ID, provide as many signals as possible. The more signals you supply, the higher the ID5 match rate and the better the cross-device linking. In the client-to-server flow, you supply these signals through the SDK's `PartnerConfig` when you initialize it - the SDK hashes and packages them for you.

Signals you can provide:

- **Hashed email (HEM)** - the single strongest signal for match rate. Pass the user's email; the SDK produces the normalized hash.
- **Hashed phone number** - a strong deterministic signal where available. Pass the user's phone; the SDK produces the normalized hash.
- **Partner user ID** - your own stable user identifier, used for cross-device reconciliation.
- **Advertising ID (IDFA / GAID)** - collected automatically by the SDK, subject to consent (ATT on iOS, device-access consent on Android).

Example (Android):

```
PartnerConfig partner = PartnerConfig.builder()
    .partnerId(<your-partner-id>)
    .hashedEmail(HashedSignal.fromEmail("user@example.com"))
    // additional signals (hashed phone, partner user ID) are configured
    .build();
```

Provide these signals whenever you have them and a valid legal basis to do so. Supplying at least one deterministic signal (typically HEM) is strongly recommended. For the full list of supported signals and exact builder methods, see the SDK README.

5. Retrieve and use the ID5 ID

Once initialized, the SDK delivers the encrypted ID5 ID to your app. To share it with demand partners, extract the EID objects from the ID5 response and include them in your bid requests as an `eids` array - the same pattern as the server-to-server integration.

See the SDK README for how to read the ID (listener or one-shot) and the exact response fields.

Related Articles

- Mobile In-App Integration (Server-to-Server)
- ID5 SDK for Android (GitHub)