

Pro / OSS split (cast-core + cast-pro AAR)

From PRO_AAR.md

Field	Value
Author	Anton Afanasyeu
Revision	R1
Creation date	2026-05-23
Last modification date	2026-05-23
Co-authored	
Severity	medium
State	postponed
Document type	technical

Table of contents

- Goals
- Entitlements API (in core)
- Module layout (Gradle)
- Play Billing flow
- Build & release
- Migration steps (suggested order)
- Relation to sender resolution test

Source: *PRO_AAR.md* — section links work in PDF viewers that support internal anchors.

Design notes for a commercial build without forking the cast protocol. **Not implemented yet** — complements COMMERCIAL.md (dev-gated Play Billing placeholders).

Goals

- **OSS** (cast-core) — protocol, UDP/TCP transport, receiver, sender engine, crash reporter client, settings model. Publishable repo for trust and forks.
- **Closed** (cast-pro **AAR**) — entitlements, license check, optional premium codecs/UI, Play Billing glue. Distributed via private Maven or Play App Bundle module (not on public Maven).
- **Single APK** on Play: app depends on cast-core + cast-pro at compile time; F-Droid / OSS build uses cast-core only with stub entitlements.

Entitlements API (in core)

Keep a thin interface in core so free code never imports Play Billing:

```
public interface CastEntitlements {  
    boolean isPro(Context ctx);  
    int maxReceivers();  
    boolean allowsResolution(CastSettings.Resolution r);  
    boolean allowsStreamProtection();  
}
```

- **Default (OSS):** FreeCastEntitlements — e.g. max 1 receiver, cap outgoing long edge at 480 unless dev override, FEC/NACK off or dev-only.
- **Pro:** ProCastEntitlements in AAR — reads signed license / Play purchase cache, returns higher limits.

Call sites (minimal first pass):

Area	Gate
CastTuningEngine / sender encode	allowsResolution()
Multi-receiver connect	maxReceivers()
Stream protection UI	allowsStreamProtection()
Developer resolution UI modes	Always allowed when BuildConfig.DEBUG or AppPreferences dev flags

CommercialFeatures stays **developer toggles** for ads/IAP/Play review; **pro entitlement** is runtime product state from Billing + AAR.

Module layout (Gradle)

```
cast-core/      # android library, published OSS  
cast-pro/      # android library, proprietary, implementation project OR maven coords  
app/           # application
```

app/build.gradle:

- implementation project(':cast-core')
- implementation 'com.foxx.androidcast:cast-pro:...' (release) or implementation project(':cast-pro') (internal)
- Optional ossRelease flavor: implementation project(':cast-core') only + FreeCastEntitlements

ServiceLoader or **manifest meta-data** to bind CastEntitlements implementation avoids compileOnly reflection hacks.

Play Billing flow

1. User buys SKU (see InAppPurchaseCoordinator.SKU_PREMIUM).
2. Pro AAR verifies purchase token, caches entitlement locally (encrypted prefs or Play Library BillingClient + acknowledge).
3. CastEntitlements refreshes; UI shows unlocked resolutions / receivers.
4. Offline grace period (e.g. 7 days) documented in product terms.

Build & release

- Version **cast-core** and **cast-pro** independently; app pins both in gradle/libs.versions.toml.
- CI: OSS job builds cast-core + F-Droid APK; private job signs cast-pro and release APK.
- Do not embed license secrets in OSS; pro AAR may contain obfuscated verification only.

Migration steps (suggested order)

1. Extract CastEntitlements + FreeCastEntitlements in app (no module split yet).
2. Wire gates at encode + multi-receiver (behind dev “simulate free tier” toggle).
3. Create :cast-core library — move network/, cast/, shared settings (large refactor).
4. Add :cast-pro with ProCastEntitlements + Billing bridge.
5. Play Console product + privacy policy (COMMERCIAL.md).

Relation to sender resolution test

Developer **Sender resolution UI mode** (`SenderResolutionUi`) is for UX experiments only. Production free tier should still enforce caps in `CastTuningEngine` via entitlements, not by hiding spinner entries alone.