CLOUDNIX TECHNICAL REPORT · CNX-TR-2026-02

Building Subscription Video and Content Platforms

Mani Krishna, Mohammed Irfan, Avinash KS, Rohit Kumar and Avinash Nishant

Cloudnix Software Labs, Bengaluru, India

CNX-TR-2026-02 · Version 1.1 · November 2026

Abstract

Media platforms are judged at three moments: when a viewer finds something to watch, when they pay for it, and when they press play. Each hides engineering that is easy to get wrong. This report describes the patterns behind subscription video and content platforms we have designed, built and operated: incremental synchronisation of channel catalogues from external video platforms, continuity of live channels across broadcaster restarts, a server-side entitlement rule evaluated on every play, pricing derived from a single daily rate, signed payment requests with verified replies, store-aware app design, and an AI-assisted editorial pipeline in which different model providers write and review against a rulebook before an editor approves. We compare embedded delivery with operator-run adaptive streaming, discuss anti-piracy and security threats, propose operational measures, and conclude with an anonymised case study, an implementation roadmap and a 20-point checklist.

Keywords: *OTT; subscription video; catalogue synchronisation; live streaming; entitlement; paywall; payment verification; app-store policy; HLS; DASH; DRM; AI-assisted editorial; maker–checker*

Contents

1. Executive summary
2. Where media platforms struggle
3. Reference architecture
4. Channel catalogues and synchronisation
5. Live broadcasting
6. Delivery: embedded players or your own streaming
7. Entitlement: who can watch what
8. Plans, pricing and renewals
9. Payments and verification
10. Apps and app-store rules
11. Editorial automation with AI agents
12. Security and anti-piracy
13. Operations and analytics
14. Case study: a subscription channel platform
15. Implementation roadmap
16. Checklist: 20 questions
17. Glossary
18. References

1 Executive summary

Media platforms are judged on three moments: when a viewer finds something to watch, when they pay for it, and when they press play. Each moment hides engineering that is easy to get wrong: catalogues that drift away from their source, live streams that break whenever a broadcaster restarts, paywalls that check the wrong thing, payment replies that are trusted without verification, and app-store rules that block a release. On the content side, automated writing that is not checked produces volume without trust.

Six principles run through this paper.

1. **One catalogue, every screen.** Web, Android and iOS read the same catalogue; promotions and categories are managed once.
2. **The catalogue follows the source.** Channels are synced incrementally from where the content lives, newest first, without duplicates.
3. **Live means "what is live now".** The platform, not the viewer, keeps track of the current stream.
4. **Check entitlement on every play.** Not at sign-in, not once a day: every time a subscriber presses play.
5. **Verify, then grant.** A payment reply is verified against its signature before any access is switched on.
6. **Automate writing, not judgement.** AI agents draft and review against a rulebook; an editor approves every piece.

2 Where media platforms struggle

Problem	What it looks like	Pattern (chapter)
Stale catalogues	New episodes appear days late or twice	Incremental newest-first sync (4)
Broken live links	Each broadcaster restart breaks every viewer's player	Live stream rotation (5)
Leaky paywalls	Premium content reachable without a plan	Entitlement per play; delivery choice (6, 7)
Complex pricing	Every plan priced by hand; mistakes in renewals	Plans derived from one rate (8)
Trusted payment replies	Access granted on an unverified reply	Signed requests, verified replies (9)
Store rejections	iOS release blocked by purchase flows	Store-aware purchase design (10)
Content at scale without trust	AI articles with invented claims	Maker–checker agents with a rulebook and an editor (11)

3 Reference architecture

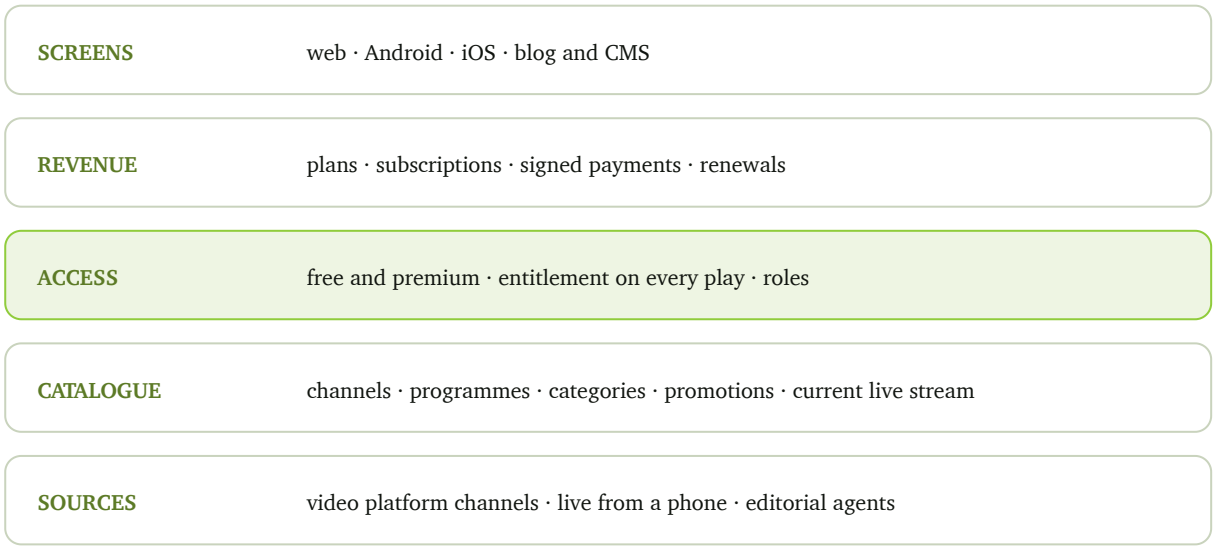


Figure 1. Five layers. Access sits between the catalogue and revenue so every play passes through it.

3.1 *Core data model*

Entity	Key fields
Channel	name, category, artwork, promo video, price, premium flag, live flag, source link, current live stream
Programme	channel, source id, title, artwork, published date
Subscription	subscriber, type (channel or programme), item, start and end dates, status, amount, payment request and reply
Promotion	photo or video, link, order, active dates
User	role: administrator, broadcaster or subscriber

Keep the payment request and reply with each subscription: they are the evidence when a subscriber disputes a charge.

4 Channel catalogues and synchronisation

Many channels already publish on a video platform. Rather than re-uploading, the catalogue links each channel to its source and keeps itself current.

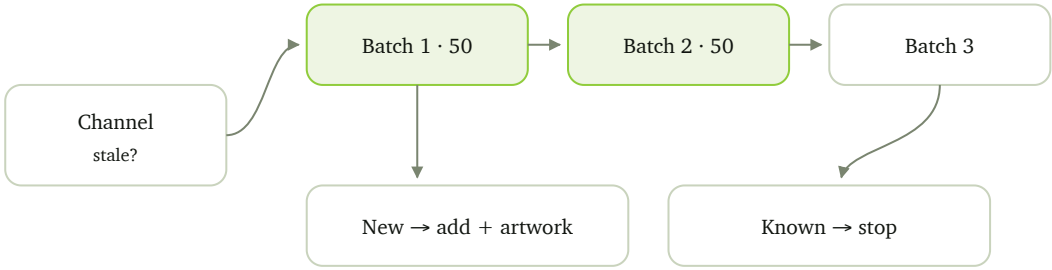


Figure 2. Newest first, fifty at a time, stopping at the first programme already in the catalogue.

4.1 *Rules for a clean sync*

- **Newest first, stop at known.** Fetch in date order and stop as soon as a batch contains only programmes you already have.
- **Identify by source id.** Each programme is stored once, keyed by its id at the source.
- **Own the artwork.** Copy thumbnails into your own media store; source links change.
- **Sync on staleness, not on every view.** Record the last sync time; refresh when it is older than the freshness window, or when an editor asks.
- **Respect quotas.** [8] Video platforms limit requests; incremental, newest-first syncs keep usage low.
- **Store keys server-side.** Never put a platform's access key in an app or page.

5 Live broadcasting

Broadcasters increasingly go live from a phone. The phone encodes and pushes the stream (typically over RTMP [7]) to a streaming service, which publishes it to viewers. The difficulty is continuity: each time a broadcaster stops and restarts, the stream gets a new identity, and every viewer's player is left pointing at a stream that has ended.

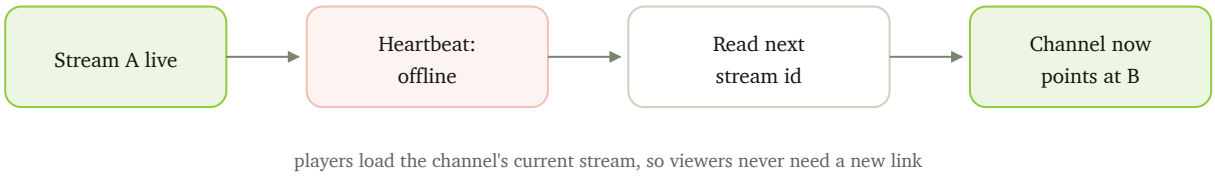


Figure 3. Live stream rotation: the platform watches each live channel and switches it to the next stream.

5.1 *Design notes*

- Store the **current live stream** on the channel, and have players request the channel, not a stream.
- Check live channels on a short schedule, and also when a viewer opens one.
- Give broadcasters their own role, limited to their channels.
- Plan for latency: phone-to-viewer delays of several seconds are normal; interactive formats need low-latency protocols.
- Record start and end times per stream for scheduling and reporting.

6 Delivery: embedded players or your own streaming

There are two ways to put video in front of subscribers. Each is right for different businesses.

	Embedded platform player	**Own streaming (HLS/DASH + CDN)**
Set-up	Fast; uses the platform's hosting and player	Encoding ladder, storage, CDN and player to run
Cost	Low	Bandwidth and encoding grow with viewers
Paywall strength	Weak: a public video can often be watched on the platform directly	Strong: signed, expiring URLs and optional DRM
Quality control	The platform's	Yours: bitrates, adaptive streaming, start-up time
Best for	Free and community channels, discovery, early stages	Premium catalogues, sports and films, licensed content

Know the limit of an embedded paywall. If premium video is hosted publicly on a video platform, the paywall controls the app, not the video. Use unlisted or members-only hosting at minimum, and move premium content to your own streaming with signed URLs when the catalogue is valuable.

6.1 *If you run your own streaming*

- Encode an adaptive ladder (for example 1080p, 720p, 480p, 360p) and package as HLS and/or DASH [1, 2].
- Serve from a CDN with short-lived signed URLs bound to the subscriber's session.
- Add DRM (Widevine, FairPlay, PlayReady) [6] for licensed or high-value content.
- Measure start-up time, rebuffering and bitrate per session.

7 Entitlement: who can watch what

Entitlement is the rule that decides whether a play request is allowed. It should be simple enough to state in one sentence and checked on every play.

```
# checked every time a subscriber presses play
if not channel.premium or channel.price == 0:      allow
elif exists subscription where
        subscriber = me and type = CHANNEL and item = channel
        and end_date >= today and payment.status = PAID:   allow
elif exists subscription for this programme (same rules):   allow
else:                                                  offer plans
```

Listing 1. Pseudocode.

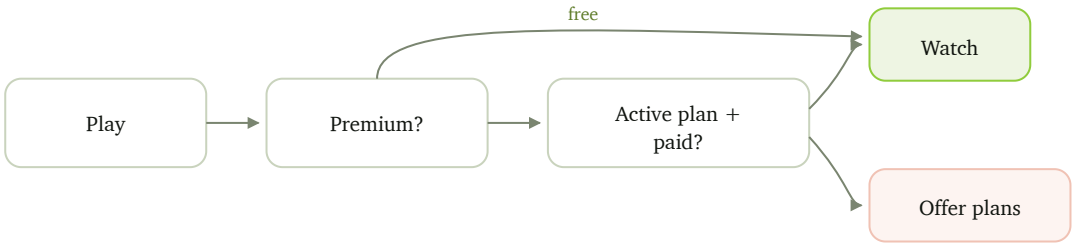


Figure 4. The entitlement decision on every play.

7.1 *Good practice*

- One active subscription per subscriber, type and item; extend rather than duplicate.
- Check entitlement on the server, never only in the app.
- Log refused plays (anonymously) to measure demand for premium channels.

8 Plans, pricing and renewals

Pricing every plan by hand creates errors. Deriving plans from one daily rate keeps them consistent and makes promotions simple.

Plan	Days	Price	Typical use
Daily	1	rate × 1	events, trials
Weekly	7	rate × 7	festivals, short series
Monthly	30	rate × 30	regular viewing
Yearly	365	rate × 365	loyal subscribers

Discounts for longer plans are applied on top of the derived price, so the base stays consistent. Renewal reminders go out before the end date; a renewal extends the current subscription instead of creating a parallel one. Per-programme subscriptions use the same mechanics for courses, series or events.

9 Payments and verification

Subscribers pay on the payment provider's own hosted page; card details never touch the platform. [5] The integrity of the flow rests on two signatures.

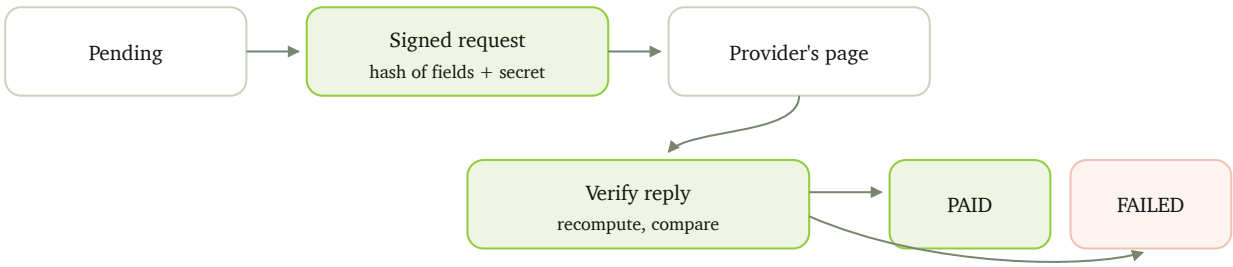


Figure 5. Sign the request; verify the reply; only a match activates the subscription.

- **Sign the request** with a hash over the transaction fields, the custom fields that carry subscription details, and a secret held only on the server.
- **Verify the reply** by recomputing the provider's reply hash (often the same fields in reverse order with the status) and comparing it.
- **Compare carefully.** Activate only on an exact match, and fail closed. A reversed comparison grants access on forged replies; it deserves a unit test of its own.
- **Use server-to-server notifications** where the provider offers them, in addition to the browser return.
- **Keep the evidence:** request, reply and verification result with each subscription.

10 Apps and app-store rules

One catalogue should drive the web, Android and iOS apps. Hybrid apps (one codebase, native shells) keep features in step; native apps make sense where playback performance or deep platform features matter.

Store topic	What to plan for
Digital purchases	Stores generally require their own in-app purchase for digital content bought inside the app [3, 4]; plan either store billing or a reader-style app that hides purchase.
Reader-style apps	Apps that only let existing subscribers sign in and watch, with purchase elsewhere, must not link to external purchase unless the store allows it.
Review	Provide demo accounts and explain premium content to reviewers.
Updates	Keep the catalogue server-driven so content and promotions change without app releases.

Our approach for iOS has been a store-aware mode: purchase is hidden in the iOS app, and subscribers manage plans on the web, keeping the app within store guidelines.

11 Editorial automation with AI agents

Publishing at volume is now possible with AI, but volume without trust damages a brand. The editorial engine we run treats AI as writer and reviewer, never as editor. [10]

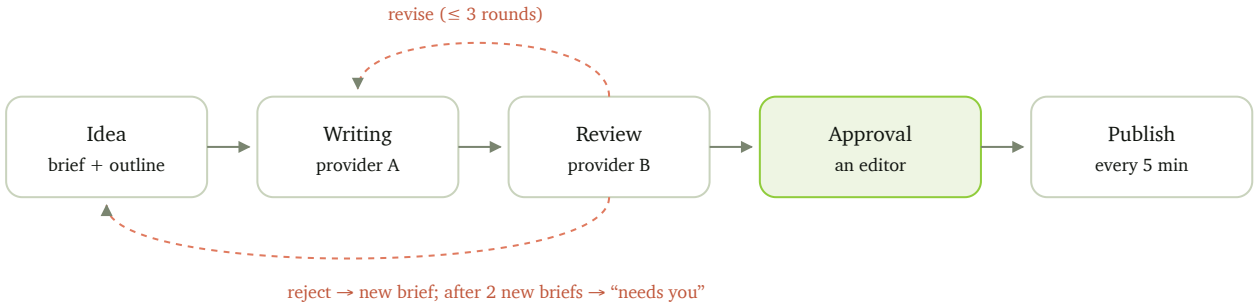


Figure 6. The editorial pipeline: maker and checker on different providers, an editor at the gate.

11.1 *How it stays trustworthy*

Control	How it works
Rulebook	83 rules every agent follows, edited by the editorial team on the review board.
Verified facts	Articles may only claim what is on a verified list; links that are not real pages are removed.
Maker–checker	The model that writes never reviews: providers swap roles each run.
Bounded retries	Up to three review rounds; repeated failure goes back to the brief, then to a person.
Editor approval	Publish, send back to writing or back to the idea, with a note.
Cost visibility	Every model call logged with tokens and images.

11.2 *Voice and imagery*

Writer personas give consistent voices: tone, habits and topics per writer. Each article mixes generated illustrations (no text or interface in them), licensed photos with credits, diagrams drawn as vector graphics, and real product screenshots. The lead image rotates between seven kinds so the feed never looks templated. Approved articles are published with category, tags, author and images on a five-minute schedule.

12 Security and anti-piracy [9]

Threat	Mitigation
Premium video reached outside the app	Unlisted or members-only hosting; own streaming with signed, expiring URLs; DRM for licensed content
Forged payment replies	Verify the reply signature; fail closed; server-to-server notifications
Account sharing	Device and concurrent-stream limits per subscription (with own streaming)
Leaked platform keys	Keys only on the server; never in apps or pages; per-environment keys
Broadcaster account takeover	Separate broadcaster role, strong sign-in, alerts on new devices
Unchecked AI claims	Verified-facts list, reviewer model, editor approval

13 Operations and analytics

13.1 *What to measure*

Area	Measures
Catalogue	time from source publish to catalogue; sync failures; duplicate rate
Live	switch-over time after a restart; viewers lost at restart
Revenue	plan mix; conversion from preview to paid; renewal and churn
Payments	success rate; verification failures; disputes
Playback	start-up time; refused plays by channel
Editorial	ideas to published; review rounds; cost per article

13.2 *Running it*

- Scheduled jobs for catalogue sync, live checks, renewal reminders and publishing.
- Nightly backups of catalogue, subscriptions and payment evidence; regular restore drills.
- Alerts on payment-verification failures and sync errors.
- A staging copy with test payment keys for every release.

14 Case study: a subscription channel platform

Anonymised. A platform of regional and niche channels, sold by subscription, on web, Android and iOS.

14.1 *The challenge*

Channel owners already published on a popular video platform and broadcast live from phones. They needed a branded place to sell access to premium channels, by the day or longer, without re-uploading everything, and without breaking live viewing each time a broadcaster restarted.

14.2 *What was built*

- A catalogue of channels by category, each linked to its source and synced newest-first, with artwork stored locally.
- Free and premium channels, with plans priced from a daily rate: daily, weekly, monthly and yearly; programme-level subscriptions too.
- Entitlement checked on every play.
- Live broadcasting from a phone, with the channel switched automatically to each new stream.
- Hosted checkout with signed requests and verified replies; request and reply stored per subscription.
- Photo and video promotion banners managed by editors.
- Web, Android and iOS apps from one catalogue, with a store-aware iOS mode.
- Roles for administrators, broadcasters and subscribers.

14.3 *What it changed*

Channel owners kept publishing where they already did, while selling access under their own brand; viewers stayed on air through broadcaster restarts; and pricing changes became one number per channel.

15 Implementation roadmap

Phase	Outcome	Key activities
1. Discovery	Catalogue, plans and delivery choice agreed	Content audit; embedded or own streaming; pricing model; store strategy
2. Pilot	A few channels live on web and Android	Sync, entitlement, payments with verification; promotions
3. Scale	All channels; iOS; live	Live rotation; broadcaster onboarding; renewals; editorial engine
4. Operate	Measured, improving service	Dashboards; payment and sync alerts; monthly reviews

16 Checklist: 20 questions

1	Does one catalogue drive every screen?
2	Do channels sync newest-first and stop at known programmes?
3	Is each programme stored once, keyed by its source id?
4	Is artwork stored in your own media store?
5	Do players follow the channel's current live stream?
6	Do broadcasters have their own role and channels?
7	Is your delivery choice (embedded or own streaming) matched to the value of the content?
8	Are premium videos protected beyond the app?
9	Is entitlement checked on the server on every play?
10	Are plans derived from one rate?
11	Do renewals extend rather than duplicate?
12	Is every payment request signed?
13	Is every reply verified, failing closed, with a unit test?
14	Is payment evidence kept with each subscription?
15	Is your iOS purchase strategy store-compliant?
16	Are platform keys only on the server?
17	Does AI content follow a rulebook and verified facts?
18	Does a different model review what one writes?
19	Does an editor approve every article?
20	Who runs it after launch: syncs, payments, support?

17 Glossary

Entitlement	The rule deciding whether a subscriber may play an item.
Catalogue sync	Keeping programmes in step with their source.
Live rotation	Switching a channel to a broadcaster's new stream automatically.
HLS / DASH	Adaptive streaming formats that switch quality to match the network.
DRM	Digital rights management: encryption and licences that control playback.
Signed URL	A link that works only for a short time and a specific viewer.
Hosted checkout	Payment taken on the provider's page, not yours.
Maker–checker	One party produces, a different one reviews.

18 References

1. R. Pantos and W. May. *HTTP Live Streaming*. IETF RFC 8216, 2017.
2. ISO/IEC 23009-1. *Dynamic adaptive streaming over HTTP (DASH), Part 1: Media presentation description and segment formats*.
3. Apple Inc. *App Store Review Guidelines*, section 3.1 (Payments).
4. Google. *Google Play Payments policy*.
5. PCI Security Standards Council. *Payment Card Industry Data Security Standard*, v4.0, 2022.
6. W3C. *Encrypted Media Extensions*. W3C Recommendation, 2017.
7. H. Parmar and M. Thornburgh (eds.). *Adobe's Real Time Messaging Protocol*, 2012.
8. Google. *YouTube Data API v3*, reference documentation (quota and search).
9. OWASP Foundation. *Application Security Verification Standard*, v4.0.
10. NIST. *Artificial Intelligence Risk Management Framework (AI RMF 1.0)*, 2023.

About this report. Prepared by the Cloudnix engineering team from systems we have designed, built and operated. Client and product names are withheld; illustrative values are marked as examples. Cloudnix Software Labs, Bengaluru, India · cloudnix.com. This report may be shared freely in its entirety.

Document control

Report	CNX-TR-2026-02	Version	1.1
Issued	November 2026	Status	Published
File	cloudnix-media-ott-platforms-v1.1.pdf		