

dash.js production checklist

Versions, services, ABR rules, LL-DASH config, DRM matrix, error classes, pitfalls.

Services you actually touch

- **MediaPlayer:** the public facade; .initialize(video, url, autoplay) does everything.
- **Settings:** singleton tree; mutate via player.updateSettings({ streaming: { ... } }).
- **StreamController:** manifest lifecycle, period switch, ad insertion, end-of-stream.
- **ProtectionController:** EME bridge to the CDM; setProtectionData({ ... }).
- **ABRController:** runs the rules pipeline; pick strategy via streaming.abr.ABRStrategy.
- **ThroughputController:** default or MOOF_PARSING bandwidth estimator (LL-DASH).
- **CatchupController:** holds the live edge; default or LOLP mode.
- **BufferController:** one per media type; owns the MSE SourceBuffer.

ABR rules pipeline (lowest-quality recommendation wins)

- **ThroughputRule** EWMA / arithmetic / geometric bandwidth pick.
- **BolaRule** Buffer-based BOLA (Lyapunov optimisation, 2016).
- **InsufficientBufferRule** Guard: drop bitrate when buffer dangerously low.
- **SwitchHistoryRule** Stability filter; prevents thrashing.
- **DroppedFramesRule** Cut bitrate when decoder is dropping frames.
- **AbandonRequestRule** Cancel slow fragment download; try lower rendition.
- **L2A (low-latency)** Online Convex Optimisation; sub-2-s latency.
- **LoL+ (low-latency)** Hybrid buffer + latency catchup; FOKUS group.

Error classes → runbooks

- **FRAGMENT_LOADER_LOADING_FAILURE** → CDN runbook; toast not banner — dash.js already retried.
- **MANIFEST_LOADER_LOADING_FAILURE** → Asset-catalogue runbook; check steering + geo-rights.
- **MEDIASOURCE_TYPE_UNSUPPORTED** → Codec / container runbook; unload() then re-init.
- **KEY_ERROR / KEY_SESSION_CREATED_ERROR** → License-server runbook; rotate token; modal 'Session expired'.
- Telemetry every error first; surface to user second.

Seven-line player (canonical pattern)

```
<video id='v' controls></video>
<script src='https://cdn.dashjs.org/latest/modern/umd/dash.all.min.js'></script>
const url = '.../manifest.mpd';
const player = dashjs.MediaPlayer().create();
player.initialize(document.
querySelector('#v'), url, true);
```

Listen via player.on(dashjs.MediaPlayer.events.PLAYBACK_STARTED, ...).

Low-latency DASH configuration

- **streaming.delay.liveDelay** target latency seconds (e.g. 3).
- **streaming.liveCatchup.maxDrift** seek-to-live threshold (e.g. 0.5).
- **streaming.liveCatchup.playbackRate** {min: -0.5, max: 0.5} bounds.
- **streaming.liveCatchup.mode** LIVE_CATCHUP_MODE_DEFAULT or LOLP.
- **streaming.abr.ABRStrategy** 'abrL2A' or 'abrLoLP' for LL-DASH.
- **streaming.abr.throughput.lowLatencyDownloadTimeCalculationMode** MOOF_PARSING for clean LL-DASH bandwidth estimates.
- Packager must produce CMAF chunks with availabilityTimeOffset.
- CDN must forward HTTP/1.1 chunked transfer encoding through to player.

Top pitfalls in production

- Targeting legacy UMD bundle on a modern Webpack/Vite project.
- Using seek(seconds) instead of seekToPresentationTime(t) for SCTE-35 ads.
- Mixing throughput modes on LL-DASH and blaming the ABR rule.
- Surfacing every fragment-loader retry as a red banner.
- Forgetting to enable CMCD on a paid CDN that supports it.
- Loading the sample reference UI in production instead of a thin component.

Version timeline (v5 line)

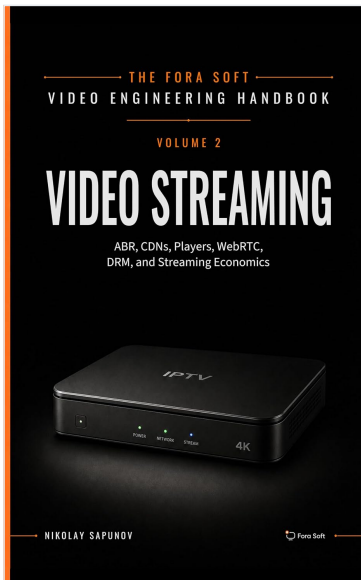
Version	Date	Notable changes
v5.0.0	17 Feb 2025	UMD legacy + UMD modern + ESM modern bundles; CMCD v2 keys ltc & msd; seekToPresentationTime().
v5.1.0	21 Nov 2025	MPEG-5 LCEVC integration; Cue Interval Tree; preselection element; ABR rule priority; new sample UI.
v5.1.1	23 Dec 2025	Current published patch line — bug fixes only on top of v5.1.0.

Multi-DRM scope

- Widevine (com.widevine.alpha) — Chrome / Edge / Firefox / Android / ChromeOS · cenc, cbcs
- PlayReady (com.microsoft.playready) — Edge on Windows / Xbox / Tizen / webOS · cenc, cbcs
- ClearKey (org.w3.clearkey) — testing only, not production · cenc

FairPlay

NOT supported in dash.js. Use native AVPlayer on iOS / Safari for the HLS-FairPlay path.



GO DEEPER · VOLUME 3 OF 7

Video Streaming: A Complete Guide to Video Delivery

From ABR and CDNs to Players, WebRTC, DRM, and the Economics of Streaming

By Nikolay Sapunov, CEO at Fora Soft · 1,182 pages · Kindle & paperback on Amazon

[Read it on Amazon →](#)
amazon.com/dp/B0HH76ZYWC

This checklist covers what a dash.js build needs in production: services, ABR rules, LL-DASH settings, DRM scope and error runbooks. Volume 3 covers the player side in depth: player architecture, ABR algorithms from buffer-based to hybrid, recovery from stalls, DRM and QoE monitoring.

WHAT YOU WILL FIND INSIDE

- Buffer-based and throughput-based ABR
- Multi-DRM in the player
- Player recovery from stalls and errors
- QoE monitoring in production

THE FULL HANDBOOK · 7 VOLUMES · 9,000+ PAGES · ALL ON AMAZON

Vol. 1 · Video Encoding — pixels and color to H.264, HEVC, AV1, quality metrics and AI

Vol. 2 · Video Streaming I — network transport, contribution and ingest, HLS, MPEG-DASH, CMAF

Vol. 3 · Video Streaming II — ABR, CDNs, players, WebRTC, DRM and the economics of delivery

Vol. 4 · Audio for Video I — PCM, codecs, loudness, accessibility, real-time voice processing

Vol. 5 · Audio for Video II — jitter buffers, lip sync, Atmos, quality metrics, neural codecs

Vol. 6 · AI for Video I — models and inference, computer vision, speech, search, generative video

Vol. 7 · AI for Video II — real-time pipelines, agents, serving, cost, compliance, verticals

Taking a dash.js player to production for live or VOD?

We will tune ABR and low-latency settings, wire DRM and error runbooks, and test the player on your device matrix. 250+ video products shipped since 2005.

[Book a 30-min call →](#)
[WhatsApp →](#)
[Email us →](#)