Shaka Player production checklist

Versions, engines, ABR knobs, LL-* config, DRM matrix, error categories, pitfalls.

Engines you actually touch

NetworkingEngine: every HTTP request; plugin per URI scheme; request and response filters.

ManifestParser: one per format (DASH, HLS, Smooth); produces internal Variant objects.

DrmEngine: EME bridge to the CDM; drm.servers config block; per-key-system advanced.

StreamingEngine: segment-fetch state machine; appends bytes into MSE SourceBuffer.

AbrManager: throughput EWMA + safety factor; replace via configure({abrFactory}).

Storage: shaka.offline.Storage; IndexedDB downloads; persistent EME licenses.

UI Overlay: optional shaka-player.ui.js bundle; accessible controls; Cast button.

ABR knobs that matter

abr.bandwidthDowngradeTarget (0.95) steady-state safety factor for the throughput pick.

abr.bandwidthUpgradeTarget (0.85) stricter factor for an upshift; asymmetry is intentional.

abr.defaultBandwidthEstimate starting bandwidth before first segment is timed.

abr.restrictions lock min/max width/height/bandwidth; hardware-tier safety.

v5.1 dropped-frame ABR auto-enabled; downshifts on decoder strain (smart TV 4K HEVC).

v5.1 low-latency hint tells AbrManager whether the stream is low-latency; live-edge stickier.

Error categories → runbooks

NETWORK → CDN health, failover, retry pressure; user: 'Network problem'.

STREAMING → Same CDN runbook + check origin shielding behaviour.

MEDIA → Packaging / MSE; unload() then load() to reset SourceBuffer.

DRM → License-server runbook; rotate token; user: 'Session expired'.

MANIFEST → Asset-catalogue; geo-rights; user: 'Not available'.

Recoverable severity → small toast; CRITICAL → red banner. Never red-banner recoverable.

Seven-line player (canonical pattern)

```
import shaka from 'shaka-player';
shaka.polyfill.installAll();
if (!shaka.Player.isBrowserSupported()) throw …;
const video = document.getElementById('v');
const player = new shaka.Player();
await player.attach(video);
await player.load('https://…/asset.mpd');
```

Same code plays .mpd, .m3u8, or .ism — Shaka picks the parser.

Low-latency configuration (DASH + HLS)

streaming.lowLatencyMode true → enables LL-DASH chunks AND LL-HLS parts.

streaming.inaccurateManifestTolerance 0 → tighter live-edge tracking.

streaming.rebufferingGoal 0.01 → start play as soon as one segment is ready.

streaming.bufferingGoal smaller target for low-latency live; pair with rebufferingGoal.

Packager must produce CMAF chunks (DASH) or EXT-X-PART parts (HLS).

CDN must forward chunked transfer encoding without stripping it at edge.

Top pitfalls in production

Skipping polyfill.installAll() → Safari edge cases break silently.

Mixing attach() and load() order on retries → use unload() + load(), not re-attach.

Surfacing recoverable errors to the user → working player looks broken.

lowLatencyMode without packager support → endless startup stall loop.

Sharing one retryParameters for all classes → license outages back off like CDN outages.

Loading the UI bundle without its controls.css → buttons render as unstyled rectangles.

Multi-DRM configuration (drm.servers map)

Key system	drm.servers key (Shaka)	Browsers / platforms	Schemes
Widevine	`com.widevine.alpha`	Chrome, Edge, Firefox, Android, ChromeOS	cenc, cbcs
FairPlay	`com.apple.fps`	Safari (macOS, iOS, iPadOS), tvOS	cbcs only + serverCertificateUri
PlayReady	`com.microsoft.playready`	Edge on Windows, Xbox, Tizen, webOS	cenc, cbcs