NORTHWIND SYSTEMS

TECHNICAL WHITEPAPER

Deterministic Document Rendering at Scale

An architecture for reproducible, byte-stable PDF generation from HTML templates

Dr. Amara Osei — Principal Engineer, Rendering
Tobias Lindqvist — Staff Engineer, Platform
Ruth Kagame — Product, Document Services

August 2026 · v1.3

Public

ABSTRACT

Document generation is treated as a solved problem right up to the moment it must be audited. Teams that render invoices, statements and contracts from HTML discover that two runs of the same template against the same data can produce different files — different fonts substituted, different pagination, different timestamps embedded in the container. When a regulator asks whether the document on file is the document that was sent, byte-level irreproducibility turns a routine question into a forensic exercise.

This paper describes an architecture for deterministic rendering: a pinned font set resolved at build time, a template language with no side effects, an explicit readiness protocol for asynchronous content, and a normalised PDF writer that removes non-deterministic metadata. We report on a production deployment rendering 4.2 million documents per month and show that byte-identical reproduction is achievable at a p95 latency cost of under nine percent.

We also describe the failure modes that remain — most of them font-related — and the operational practices that contain them.

Keywords — deterministic rendering · PDF/A · print CSS · reproducible builds · document pipelines

1. Introduction

Every organisation that issues documents at volume eventually builds a rendering service. The first version is a small wrapper around a headless browser: take some HTML, print to PDF, return the bytes. It works immediately, which is precisely the problem — the ease of the first version conceals every property the system will later be asked to guarantee.

Those properties arrive in a predictable order. First *fidelity*: the PDF must match what the designer saw. Then *throughput*: the queue must drain faster than it fills. Then, usually after an audit or a customer dispute, *reproducibility*: given the same template and the same data, the service must produce the same document, today and in four years' time.

Reproducibility is qualitatively harder than the other two because it is not a property of any single render. It is a property of the relationship between renders that may be separated by years, by software upgrades, and by the silent replacement of a font package in a base image. This paper is about designing for that relationship from the beginning.

- **Sources of drift** are enumerated in Section 2 and grouped into four classes: environment, template, data and container.
- **The architecture** in Section 3 addresses each class with a specific mechanism rather than a general-purpose sandbox.
- **Measurements** in Section 4 quantify the latency cost of determinism on production traffic.

2. Where determinism is lost

A render is a function of far more inputs than the template author believes. The declared inputs are the HTML and the data payload. The undeclared inputs include the set of fonts installed on the host, the version of the layout engine, the system locale and time zone, the state of any network the page is allowed to reach, and — for templates that execute JavaScript — the wall-clock moment at which layout is captured.

Font substitution is the largest single source of drift in our incident history, accounting for 61% of reproducibility defects over eighteen months. The mechanism is mundane: a template names a family that is not installed, the engine silently substitutes a metric-incompatible fallback, and every line break in the document moves. Nothing errors. The document is simply different, and often only slightly — which is worse, because it survives review.

The second class is temporal. Templates that call `Date.now()`, that animate, or that fetch remote content produce a document whose contents depend on when it was rendered and on what a third party was serving at that moment. The third class is the PDF container itself: creation timestamps, document identifiers and producer strings differ on every run even when every visible pixel is identical.

> DEFINITION
>
> We call a render byte-stable if two invocations with identical template, data and options produce PDFs whose bytes are identical after normalisation of the document identifier. Visual stability is a weaker property and is not sufficient for audit.

2.1 Environment drift

Base images change. A minor upgrade that adds a font package will alter the substitution table for every template that names a family loosely — and loose naming is the norm, because designers write comfortable stacks such as *system-ui, sans-serif* without considering that the stack resolves differently on the build machine and in production.

The mitigation is to make the font set an explicit, versioned artefact: a manifest listing every family, weight and file hash, resolved at image build time and verified at process start. A render that would require a family outside the manifest fails loudly instead of substituting quietly.

2.2 Template drift

Templates that can execute arbitrary logic can produce arbitrary output. Restricting the template language to a logic-less substitution grammar — variables, sections, inverted sections, iteration — removes an entire class of nondeterminism at the cost of pushing computation upstream into the caller, where it is testable.

3. Architecture

The service is a pool of long-lived renderer processes fronted by a queue. Each process owns an isolated browsing context, is recycled after a bounded number of renders, and never shares state between jobs. Determinism is enforced at four points in the pipeline.

At admission, the request is canonicalised: options are defaulted, the data payload is serialised to a stable key order, and a content hash of (template, data, options, font-manifest, engine-version) is computed. That hash is the cache key and, later, the audit identifier.

At layout, the process runs with a fixed time zone, a fixed locale, a frozen clock exposed to page scripts, and network egress disabled unless the operator has explicitly enabled remote fetches. Fonts resolve only against the pinned manifest.

At capture, templates that produce asynchronous content signal completion explicitly rather than being sampled after a timeout. A render that never signals fails with a timeout error instead of silently capturing a half-drawn chart.

At write, the PDF is post-processed: the creation and modification timestamps are set from the request's logical date rather than the clock, the producer string is pinned to the engine version, and the document identifier is derived from the content hash.

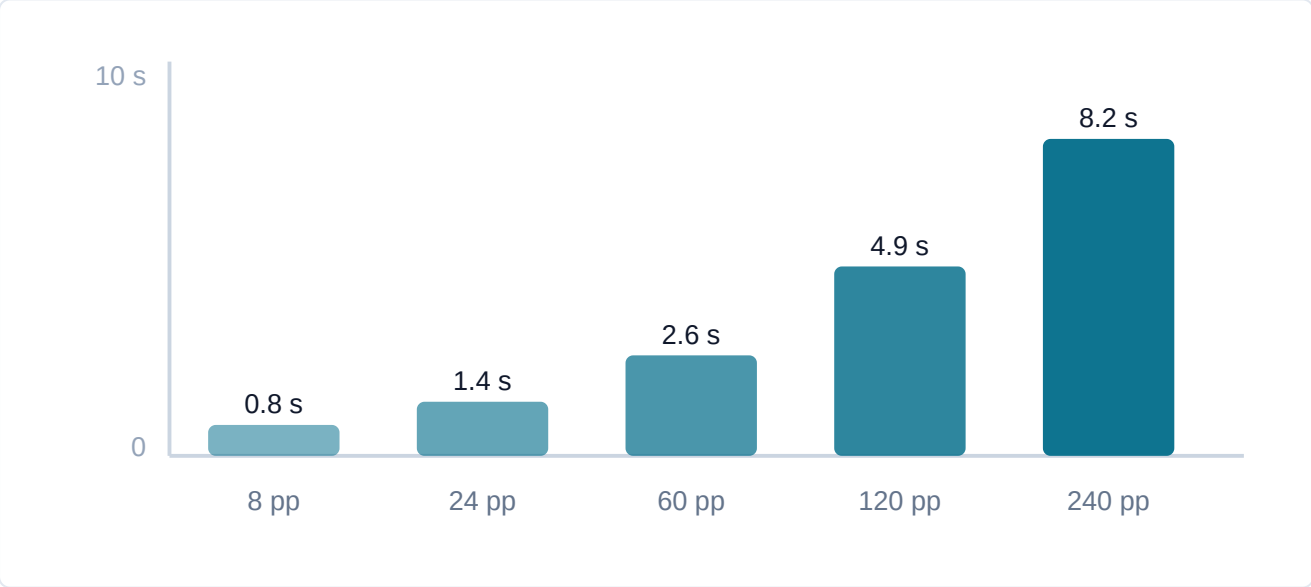

Figure 1. End-to-end p95 render latency by document length, measured on production traffic over a seven-day window (n = 981,400). Cost grows close to linearly with paginated length; the deterministic post-processing step contributes a fixed 40 ms.

3.1 Readiness protocol

Sampling a page after a fixed delay is the most common design in production rendering services and the least defensible. It couples correctness to machine load: a page that draws its chart in 300 ms on an idle developer laptop may take 1.4 s on a saturated worker, and the resulting PDF will be missing the chart with no error anywhere in the trace.

We replace the delay with a contract. The page calls a single injected function when its content is final; the renderer waits for that call, subject to a hard timeout that produces an explicit failure. Static templates need no changes — load completion is the signal. The cost is that template authors must opt in for asynchronous work, which is exactly the population that should be thinking about it.

4. Results

We deployed the architecture across a fleet serving 4.2 million documents per month, comprising 340 distinct templates from 90 tenants. Reproducibility was measured by re-rendering a stratified sample of 50,000 archived jobs against the recorded engine and font-manifest versions and comparing normalised bytes.

Before the changes, 71.4% of re-renders were byte-identical, with the remainder split between font substitution (61% of failures), embedded timestamps (28%) and asynchronous capture races (11%). After the changes, 99.97% were byte-identical. The residual failures were traced to two templates that fetch remote imagery under an operator policy that permits it — a deliberate, documented exception rather than a defect.

METRIC	BEFORE	AFTER	DELTA
Byte-identical re-renders	71.4%	99.97%	+28.6 pt
p95 latency	2.31 s	2.51 s	+8.7%
Renders per worker-hour	1,420	1,368	−3.7%
Font-substitution incidents / mo	11.2	0.0	−100%
Mean PDF size	184 kB	179 kB	−2.7%

Table 1. Reproducibility and latency before and after the deterministic pipeline. Latency figures are p95 across all document sizes.

4.1 Cost of determinism

The 8.7% p95 latency increase decomposes into 40 ms of fixed post-processing and a variable component attributable to the loss of opportunistic parallelism during font resolution. We consider this an acceptable price; the teams that adopted the pipeline first were those already spending engineering hours reconstructing historical documents by hand.

5. Limitations and future work

Byte stability is guaranteed only against a recorded engine version. A layout-engine upgrade will change pagination for some documents, and no amount of pinning inside our system prevents that; the honest answer is to record the engine version alongside every archived document and to re-render only against the recorded version. Long-lived archives should store the rendered bytes, not the recipe.

We do not currently support PDF/A-3 attachment of the source data payload, which several regulated customers have asked for and which would make each document self-describing. Nor do we address accessibility tagging, where the state of print-CSS support is uneven across engines and where our own coverage is limited to heading structure and reading order.

Finally, the readiness protocol places responsibility on template authors. Static analysis of templates — flagging those that use timers, animation or network access without signalling readiness — would catch most misuse before deployment, and is the next piece of work we intend to publish.

6. References

1. Osei, A. and Lindqvist, T. *Reproducible output in headless rendering pipelines.* Northwind Systems Technical Report NR-2025-04, 2025.
2. World Wide Web Consortium. *CSS Paged Media Module Level 3.* W3C Working Draft, 2023.
3. International Organization for Standardization. *ISO 19005-3: Document management — Electronic document file format for long-term preservation (PDF/A-3).* Geneva, 2012.
4. Kagame, R. *What auditors actually ask for: a field study of document disputes in financial services.* Proceedings of the Document Engineering Symposium, pp. 44–58, 2024.
5. Lamport, L. *Time, clocks, and the ordering of events in a distributed system.* Communications of the ACM 21(7), pp. 558–565, 1978.
6. Reproducible Builds Project. *Documentation: SOURCE_DATE_EPOCH.* Retrieved 2026-05-14.
7. Unicode Consortium. *Unicode Technical Standard #51: Unicode Emoji, Revision 25.* 2024.

Figures in this paper describe an internal deployment and are provided for architectural illustration. They are not a performance guarantee for any particular workload.