Explicio

WHITEPAPER · EXPLICIO

Reproducible Decisions on Hardware That Is Not Bit-Stable

Securing the decisions of an AI knowledge-management system against inference-kernel noise



AS OF

Version of September 20, 2026, based on measurements of September 19 and 20, 2026

AUTHOR

Christian Thiede
TACP Operations GmbH & Co. KG (Explicio), Detmold, Germany

Contents

1 **Executive summary** 3

2 **Why this matters** 5

3 **What was measured** 6

- Environment 6
- Tools 6
- Prompts 7
- Rules 7

4 **Results** 8

- No serving switch removes the noise 8
- Flips happen only at near-ties 9
- The block that mixes the experts 10
- Controls 12
- The price of bit-stable operation 12

5 **What Explicio built from this** 14

6 **Reading for audits and the EU AI Act** 16

7 **Limitations** 17

8 **Reproducibility and provenance** 18

- Versions and raw data 18
- About this document 18

CHAPTER 1

Executive summary



A language model served at `temperature=0` is commonly taken to be deterministic: same input, same output. On the hardware Explicio uses to serve its models to EU customers, that assumption does not hold. Twenty identical requests against an idle server produce seventeen different texts. The cause sits in the compute kernel that mixes the experts of a mixture-of-experts model stored in NVFP4; no serving switch removes it, and the inference server's batch-invariant mode refuses exactly this kernel; for this model family the mode is unavailable altogether because of its gated-DeltaNet layers.

For a knowledge-management system, the wording of an answer matters less than the decision behind it. Does the system ask a clarifying question or not? Is the input a question or a contribution of knowledge? Which edges does the knowledge graph receive? The measurements show where those decisions flip: only at positions where the model rates two continuations almost equally. Above a margin of two nats nothing flips; across 5,478 measured positions it never happened once.

Explicio has turned this into a pattern that separates the decision from the long output: a short call decides, a long call extracts. Since then the decisions have been stable in every checked production case, and the completeness of the extraction is measured at every release instead of assumed. Bit-stable operation is possible in the same container in two ways: with fp8 expert weights on the Triton kernel at 1.4 to 1.8 times the single-stream compute time, or with a dense model at 7.4 times.

Figure	Value	Source
Distinct texts in 20 runs, production recipe, idle server	17	Ladder, raw file `2026-09-19-leiter.ndjson`
Flip points with a margin above 2 nats	0 of 5,478	Logprob probe, all cells
Bit-stable runs with the dense model (three prompts)	60 of 60	Raw file `2026-09-19-leiter-2x2.ndjson`
Bit-stable runs of the MoE model with fp8 weights (Triton kernel)	60 of 60	Raw file `2026-09-20-leiter-2x2.ndjson`
Stable decisions after the redesign (eight production cases, three runs each)	8 of 8	`analyze-probe`, report § 7

Figure	Value	Source
First diverging building block inside the running model	expert block, layer 0, prefill	raw file `opdet-nvfp4-insitu-20260920T095347Z.ndjson`
Compute time of the bit-stable alternatives, single stream	fp8 on Triton 1.4× (without speculation) to 1.8× (production recipe), dense 7.4×	Throughput raw files (`perf.ndjson`)

CHAPTER 2

Why this matters



The question came out of a regression test. One case in Explicio's golden set checks that the assistant does not invent a page reference when two documents contradict each other. The case passed on some days and failed on others, with unchanged code. The first suspicion, a flaw in the prompt wording, turned out to be half right: three causes did sit in the prompt contracts and have been fixed. The fourth sat deeper. On every run the model computed slightly differently, and wherever two continuations were almost equally likely, sometimes one won and sometimes the other.

For a knowledge-management system this is not harmless. Explicio takes in knowledge from chats, documents, images, and voice messages, breaks it into the nodes and edges of a knowledge graph, and answers questions with evidence drawn from that graph. At several points along this chain a model takes a decision with consequences: whether an input is ambiguous and triggers a clarifying question; whether it is knowledge at all or a question to the assistant; whether two statements contradict each other; which relation connects two nodes. If these decisions vary from run to run, a customer sees a clarifying question once and none the next time for the same input, and the same evidence is recorded as a contradiction once and not the next time.

Reproducibility has three levels that need to be kept apart:

1. **Bit-identical**
   : two runs produce the same character sequence. This is what `temperature=0` promises.

2. **Decision-identical**
   : two runs take the same structural decisions even if the wording differs.

3. **Meaning-identical**
   : two runs say the same thing in different words.

For operating a knowledge-management system, level 2 is the one that counts. Level 1 is desirable but expensive; level 3 is not measurable enough to build a gate on. This document describes how far the hardware supports level 1, where level 2 is at risk, and how Explicio secures it.

CHAPTER 3

What was measured



Environment

All measurements run on the hardware Explicio ships to customers and operates itself: NVIDIA GB10 with unified memory (no separate GPU memory account), vLLM 0.24.0 from the NGC container 26.07, FlashInfer 0.6.14, CUDA 13.3. The production model is `nvidia/Qwen3.6-35B-A3B-NVFP4`, a mixture-of-experts model with 256 experts and eight active experts per token, whose expert weights are stored in NVFP4 and computed through the Marlin kernel (W4A16, group size 16); the projections of the attention and gated-DeltaNet layers are static FP8. The production recipe uses multi-token speculation (MTP, three tokens), an fp8 key-value cache, prefix caching, and asynchronous scheduling.

The comparison cell without expert mixing uses `nvidia/Qwen3.6-27B-NVFP4`, the dense sibling of the same family in the same format, in the same container.

Tools

Four tools, all without dependencies outside the Python standard library, all producing raw data per run:

The probe sends the same prompt to an endpoint twenty times, once run after run on an idle server (c=1) and once with three concurrent disturbing requests (c=4). It counts distinct texts, distinct decision signatures (the set of extracted keys, the relation triples, and the two flags for clarification and question), and the position of the first divergence.

The ladder drives serving configurations that differ from the previous rung by exactly one change through the same probe: without prefix cache, without speculation, with a bf16 cache, with all four removed together, with the batch-invariant mode, with the dense model. Every rung runs on a box that has been taken out of the serving pool and whose production profile has been stopped; the box shares nothing.

The logprob probe requests, for every generated token, the twenty most likely candidates with their log-probabilities. It aligns runs at the token level, determines at every divergence the margin (log-probability of the chosen token minus that of the runner-up), and measures noise in the common prefix of two runs: the difference between the log-probabilities of the same token at the same position under identical context.

The operator harness calls the individual compute building blocks of the model in isolation inside the container (normalization, router, gated-DeltaNet core, attention, FP8 linear) and compares five repetitions bit for bit, once with an identical batch and once with disturbing rows appended.

Prompts

Two prompt sets were used. The production prompt is the extraction contract Explicio uses in the product, with a factual note of 1,881 tokens about two documents that contradict each other on the operating pressure of a pump. The public set consists of twenty synthetic factual notes (ten German, ten English, 40 to 80 words, no product, customer, or personal data) with a neutral extraction contract; it is published together with the tools and raw data so the results can be reproduced on other hardware.

Rules

Every number in this document has a raw file. Every result that carries a recommendation has been measured twice. A candidate that fails to start is a result and appears in the table. And because a measurement that shows the same configuration green once and red once proves nothing without repetition, each rung is run twenty times before anyone touches a wording.

CHAPTER 4

Results



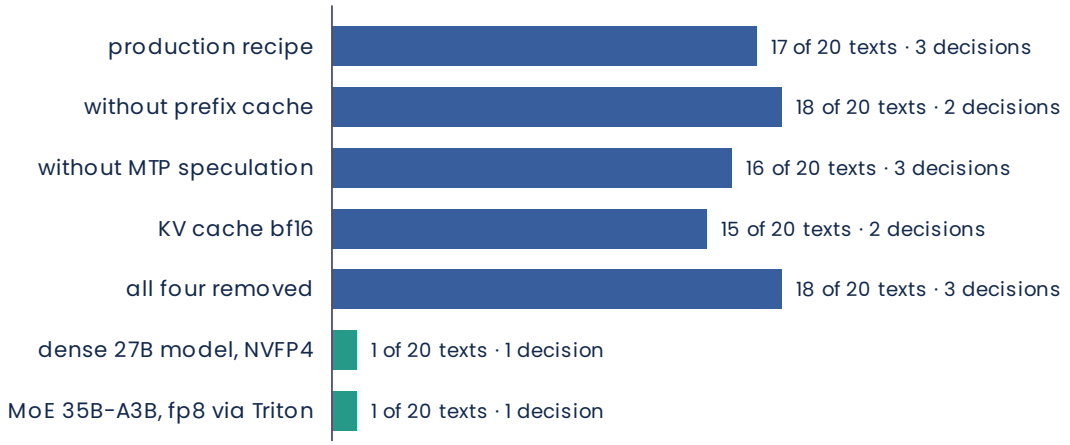
No serving switch removes the noise

The ladder answers the first question: is it a server setting? It is not. Each rung removes a feature that could alter compute paths, and no rung changes the picture.

Configuration	Distinct texts (20 runs, idle server)	Distinct decisions	First divergence (median, characters)
production recipe	17	3	171
without prefix cache	18	2	175
without MTP speculation	16	3	171
key-value cache bf16	15	2	171
all four removed together	18	3	171
batch-invariant mode (`VLLM_BATCH_INVARIANT=1`)	fails to start	n/a	n/a
dense 27B model, NVFP4	1	1	none
MoE 35B-A3B, fp8 weights on the Triton kernel	1	1	none
MoE 35B-A3B fp8, full production recipe (speculation, fp8 cache)	1	1	none

Evidence: `docs/messungen/determinism/2026-09-19-leiter.ndjson`, `2026-09-19-leiter-2x2.ndjson` and `2026-09-20-leiter-2x2.ndjson`, table produced by `tools/bakeoff/determinism-report.py`.

FIG. 1 **Distinct texts per configuration**



1 text = bit-stable. Raw data: 2026-09-19-leiter.ndjson, 2026-09-19/20-leiter-2x2.ndjson

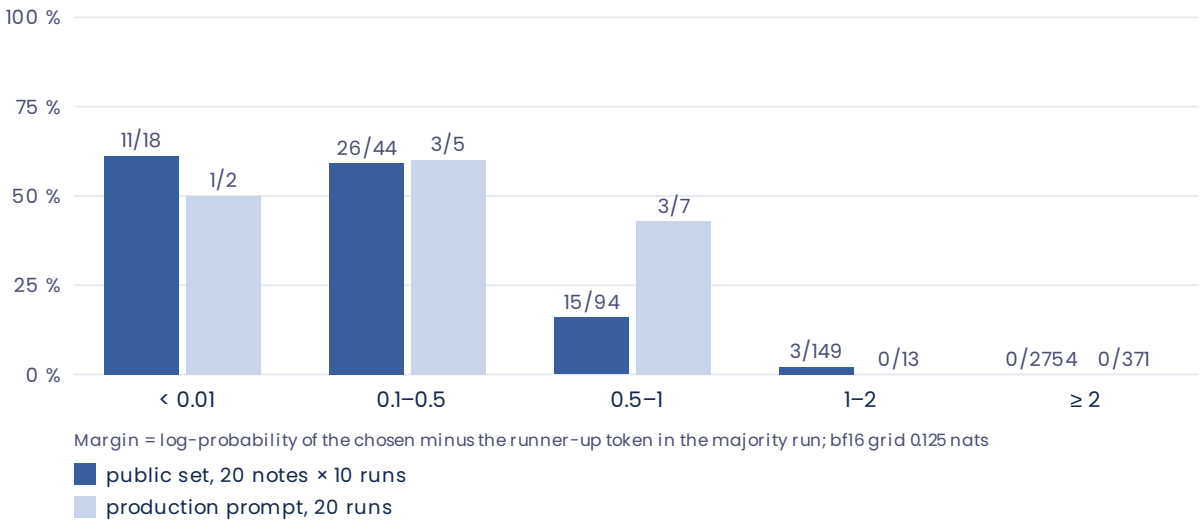
The batch-invariant mode fails to start, and the error message says why: `NvFp4 MoE backend 'MARLIN' does not support the deployment configuration since kernel does not support batch invariance`. Inside the container only one expert kernel declares invariance, the Triton kernel for fp8 weights; every NVFP4 kernel declines. With fp8 weights that kernel is selected, and the mode then fails one layer earlier: `VLLM batch_invariant mode is not supported for GDN_ATTN`, the attention backend of the gated-DeltaNet layers. For this hybrid model family the mode is therefore unavailable regardless of the weight format.

Across all 200 ladder runs the decisions that matter to the product stay stable: the `contradicts` edge between the two documents is set in 200 of 200 runs, three nodes are recognized in 200 of 200 runs, a clarifying question is asked in 0 of 200 runs. What varies is one edge type (36 of 200 runs `documents` instead of `applies_to`) and the spelling of one key (5 of 200).

Flips happen only at near-ties

The logprob probe makes visible what the ladder only counts. At every position where two runs diverge, the majority run has a margin between the chosen and the runner-up token. That margin is small at flip points and large at stable positions: in the production prompt it is 12.2 nats at stable positions (median), 0.375 nats at the seven flip points (median), and never above 0.875 nats.

FIG. 2 **Flip rate per margin class**



On the public set with 20 notes and 10 runs each, the flip points distribute as follows: below 0.01 nats 11 of 18 near-ties flip, between 0.1 and 0.5 nats 26 of 44, between 0.5 and 1 nat 15 of 94, between 1 and 2 nats 3 of 149, and above 2 nats 0 of 2,754. The production prompt shows the same picture with 0 of 371 above 2 nats. The classes between 0.01 and 0.1 are empty because the API returns log-probabilities in bf16 resolution: all values are multiples of 0.125 nats, and a margin below 0.125 is an exact tie on this scale.

Evidence: `2026-09-19-logprobs-public.ndjson` (20 cells) and `2026-09-19-logprobs.ndjson`, table produced by `determinism-report.py --margins`.

How large is the noise? In the common prefix of two runs, that is under identical context, the log-probabilities of the runner-up token differ by 0.375 nats at the median, 1.125 nats at the 90th percentile, and 11.5 nats at the maximum. The chosen token itself shows almost no noise (median $8 \cdot 10^{-6}$), which the log-softmax explains: its normalizer is dominated by the top token, whose own noise cancels out. The outliers of several nats are discrete jumps of the kind produced when a layer selects a different expert. Rounding noise would stay within one or two bf16 increments.

The flip threshold and the noise fit together: flips happen where the margin falls below the noise. A near-tie does not flip for certain, but it flips often: on the public set 157 near-ties held and 55 flipped.

The block that mixes the experts

The second question is which building block produces the noise. Two measurements answer it from two sides.

The dense 27B model of the same family, in the same NVFP4 format, in the same container, with the same Marlin kernel for its weights, produces 60 of 60 bit-identical runs across three prompts. The logprob probe finds no single difference of one bf16 increment across 1,054 positions, and three exact ties (margin 0 on the bf16 grid) fall on the same side 20 of 20 times. That is only possible if the logits are bit-identical. In the dense cell the same building blocks compute as in the production model (dequantization and matrix multiplication of the NVFP4 kernel, FlashInfer attention, the gated-DeltaNet layers, normalization, sampler), with one exception: the block that selects eight of 256 experts per token, dispatches the input to them, and merges their weighted results.

The same MoE model with fp8 expert weights, computed by the Triton MoE kernel with no invariance switch, is bit-identical in 60 of 60 runs across the same three prompts, with zero logit noise at 875 positions. The noise is therefore not a property of expert mixing as such but of the NVFP4 Marlin path. This holds in the full production recipe with speculation, fp8 cache, prefix caching and asynchronous scheduling as well: 60 of 60 bit-identical, also under disturbing load. The dense model in fp8 is bit-stable on three prompts, also under load. A repetition of the fp8 MoE cell after a full server restart produced the same text as the first run, 20 of 20 in both runs; the stability therefore holds across process boundaries. The in-situ harness on the fp8 model shows the same at the activation level: between two runs none of the 251 modules differs, and in the batch variant all 40 layers stay bit-identical; only the final logit projection reacts to the different batch height. Batch dependence appeared once on the fp8 path: the cell without speculation produced two texts in contiguous groups on the production prompt under load, while the five other fp8 cells under load stayed at one text.

The operator harness confirms this for the building blocks that can be called in isolation. Normalization, the router with top-8 selection, the gated-DeltaNet core, FlashInfer prefill, and the FP8 matrix multiplication are bit-identical at 1,881 tokens across five repetitions and under appended disturbing rows, with and without the invariance switch (maximum deviation 0 in every cell). The expert kernel itself cannot be called in isolation because it needs the repacked weights of a real model layer.

The proof inside the running model comes from the harness's second mode: hooks on all 251 modules of the production model record every output during two runs of the same prompt and compare them in execution order. The first tensor that diverges is the output of the expert block of the first layer during prefill. Before it, the input and post-attention normalizations, the router logits, and the shared expert of the same layer are bit-identical; after it, every subsequent layer diverges. The deviation is tiny: 54 of 3,852,288 elements (1,881 tokens × 2,048 dimensions) differ, by at most $6.1 \cdot 10^{-5}$ in absolute terms, which for small values is up to 25 percent relative. A batch with one disturbing request appended shows the same first divergence in the same module.

Evidence: `2026-09-19-leiter-2x2-logprobs.ndjson` (the dense cells), the `opdet-ops` and `opdet-ops-invariant` files and `opdet-nvfp4-insitu-20260920T095347Z.ndjson` in the `opdet` folder.

One qualification belongs here. Under three concurrent disturbing requests the dense model produces two texts on the production prompt, but not randomly distributed: runs 1 to 9 say one phrase at character 782, runs 10 to 20 another. That is the dependence on batch composition which the batch-invariant mode addresses. The dense model is therefore bit-stable at a fixed batch shape, but not batch-invariant.

Controls

Three objections were obvious, and all three have been measured.

First objection, speculation: the production recipe uses an auxiliary model to guess three tokens ahead and lets the main model verify them; that verification step could shape the log-probabilities. The same probe against the same model without speculation shows the same flip threshold: 9 of 9 flip points below the noise p99, and above 2 nats 0 of 798 positions. With speculation: 13 of 13, and 0 of 800.

Second objection, two boxes: the first measurements against the production endpoint ran across two boxes in the pool. The control on a single box during a measurement window changes nothing in the picture.

Third objection, the contract: the production prompt carries the hardening described in the next section. The public set with a neutral extraction contract shows what happens without it. Across 400 runs over 20 notes, 26 runs deviate from the majority in one of the two flags (clarification, question), and 92 runs deviate in structure (key set or relations). That is 6.5 percent and 23 percent. Kernel and server are the same in both measurements; only the contract differs.

Evidence: `2026-09-19-leiter-2x2-logprobs.ndjson` (the `det-prod-pub` and `det-plain-pub` cells), `2026-09-19-public-ref.ndjson`.

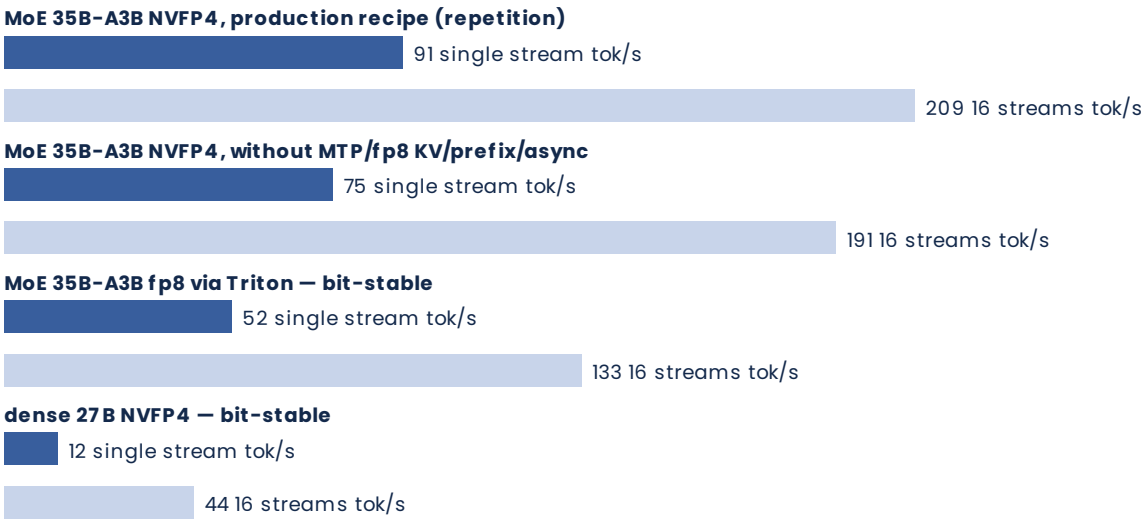
The price of bit-stable operation

Bit stability is available in today's container with the dense model, and the price can be quantified on the same box.

Configuration	TTFT (s)	Single stream	16 streams	Prefill
MoE 35B-A3B NVFP4, production recipe	0.130	91.5	209	5,420
MoE 35B-A3B NVFP4, without speculation, cache, and prefix	0.095	75.4	191	5,556
MoE 35B-A3B fp8 on Triton, without speculation and caches, bit-stable	0.146	52.2	133	3,414
MoE 35B-A3B fp8 on Triton, production recipe, bit-stable	0.206	51.3	143	3,231
dense 27B NVFP4, bit-stable	0.308	12.3	43.6	920
dense 27B fp8, bit-stable	0.401	7.8	29.4	366

Throughput in tokens per second; TTFT = time to first token. Evidence: `2026-09-19-det-prod-pub-perf.ndjson`, `2026-09-19-det-plain-pub-perf.ndjson`, `2026-09-19-det-dense-nvfp4-perf.ndjson`, `2026-09-20-det-moe-fp8-plain-perf.ndjson`, `2026-09-20-det-moe-fp8-perf.ndjson`, `2026-09-20-det-dense-fp8-perf.ndjson`; measurement tool `tools/bakeoff/perf.py`, 256-token prompts.

FIG. 3 **Throughput per configuration**



Raw data: det-{prod-pub,plain-pub,dense-nvfp4,moe-fp8-plain}-perf.ndjson (Sept 19/20, 2026)

The fp8 variant of the same MoE model on the Triton kernel costs, at equal function (production recipe against production recipe), 44 percent of single-stream throughput, 31 percent at 16 streams and 40 percent of prefill, plus about 12 GB of additional weight memory. Without speculation and caches on both sides the figures are 31, 30 and 39 percent; the speculation that gives the NVFP4 recipe 21 percent in a single stream gives the fp8 recipe nothing there. The dense model needs 7.4 times the time in a single stream and 4.8 times at 16 streams because it computes 27 billion parameters per token instead of three, and the dense fp8 variant is slower still. The fp8 cells had failed in the first measurement window on a warm-up error of the container (DeepGEMM, `Unknown recipe`, although vLLM had disabled DeepGEMM for the model itself) and ran in the second window with DeepGEMM switched off.

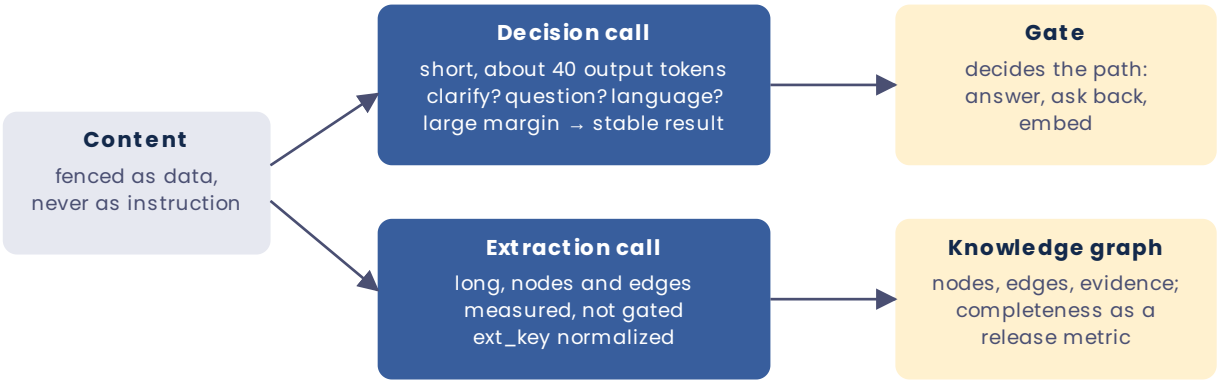
One number in the table is itself a result of the method. The first measurement of the production recipe had yielded 3,288 tokens/s prefill, the repetition 5,420. The first measurement ran immediately after a restart of the box and was not reliable; the table shows the repetition.

CHAPTER 5

What Explicio built from this

The consequence is a pattern at the contract level that moves the decision to where the noise cannot reach it; model and kernel stay unchanged.

FIG. 4 **Decide short, extract long**



The kernel noise hits the long output; the decision sits in a call whose margin the noise does not reach.

Decide short, extract long. The analysis step gives the model two separate tasks. A short call of about forty output tokens answers three questions: is a clarifying question needed, is the input a question to the assistant, and which language is it written in. A long call extracts nodes, edges, and evidence. The decision of the short call steers the path; the extraction fills the graph. A short output has few positions at which a near-tie can arise, and a yes-or-no decision with a clear rule has a large margin at its position. The measurement after the redesign: in eight production cases with three runs each, the decision is stable eight times.

A contract asks only for what exists. Three of the four causes of the golden-set case sat in the contracts themselves. A bare storage date in the contradiction block became a page number, a label reading "no locator" became a parenthesis behind the value, and the marker "p." from the citation rule ended up in "(p. 180 bar)" where there was nothing to cite. Since then every value travels into the prompt with its label, and the citation rule is inserted only when a locator exists. A contract that demands page numbers gets them even where there are none.

Keys are normalized. The spelling of an `ext_key` (5 of 200 runs) is not a model problem but a missing rule: the ingestion path now normalizes keys canonically before they enter the graph.

Completeness is measured, not gated. The extraction still flutters: in four of eight production cases the number of edges or nodes differs between three runs. A gate that turned red on this would always be red on this kernel. So the probe counts the deviation and writes it into every model release, so that a change of weights or container moves it visibly. The release gate additionally checks that the decisions stay stable across three runs.

Two rules have moved from this into Explicio's engineering guidelines. A decision call must not sit on a contract boundary, because there the noise decides. And where a prompt gets the same thing wrong three times, the format is at fault, not the wording.

CHAPTER 6

Reading for audits and the EU AI Act



By its own classification Explicio is not a high-risk AI system in the sense of Regulation (EU) 2024/1689; the classification, its derivation, and every reassessment are recorded in the company's AI system register. The record-keeping and traceability duties of Article 12 therefore do not apply directly. The measurements in this document nevertheless answer a question any auditor may ask, and which without measurement could only be answered with an assertion: what does "reproducible" mean for this system, technically?

The answer in three sentences: on the hardware Explicio operates, the character sequence of an answer is not guaranteed for the same input. The structural decisions (clarification, question, language, and the edges the product needs for its evidence) are, after the redesign, in every checked case. The completeness of an extraction is not, and it is quantified at every release.

For the traceability of individual answers, every output of the assistant carries a marker with origin, model, prompt version, route, and the sources used; for every answer it can therefore be stated which model in which version produced it and from which evidence. The model name appears in the transparency sheet, every channel carries the notice under Article 50, and a model change is an entry in the model register in the same commit as the configuration.

Publishing these measurements does not change the classification. They describe the system; they train no model, alter no weights, and assess no person.

CHAPTER 7

Limitations



- The fp8 cells are measured on three prompts with 20 runs each, the repetition on the production prompt. The server's batch-invariant mode remains unmeasurable for this model family because it does not support the gated-DeltaNet layers; the statement that only the Triton kernel declares invariance is a statement about source code.
- The in-situ proof names the module, not the line in the kernel. Whether the dispatch of tokens to experts, the per-expert matrix multiplication, or the merge of expert results produces the difference is open; that is the question a purpose-built invariant kernel would have to start from.
- One hardware platform, one container, one model family. All numbers hold for GB10, vLLM 0.24.0 from NGC 26.07, and Qwen3.6 in NVFP4. Whether other accelerators, other kernels, or other quantizations show the same picture, this document does not say.
- The throughput numbers are single measurements with one repetition. The repetition exposed an outlier; further repetitions would quantify the spread that stands here only as a warning.
- The public prompt set is synthetic. It is built to produce the class of outputs the product produces, but it is not the product.
- Log-probabilities come in bf16 resolution. Margins below 0.125 nats cannot be resolved on this scale; the class "below 0.01" really means "tie".
- The statement "nothing flips above 2 nats" is an observation over 5,478 positions, not a guarantee. A noise outlier of 11.5 nats has been measured; it hit a position with a large margin, not a flip point.

CHAPTER 8

Reproducibility and provenance



Versions and raw data

Component	Version
Inference server	vLLM 0.24.0+092c4842, container `nvcr.io/nvidia/vllm:26.07-py3`
Accelerator	NVIDIA GB10, CUDA 13.3, FlashInfer 0.6.14, torch 2.13.0a0+nv26.07
Production model	`nvidia/Qwen3.6-35B-A3B-NVFP4`
Dense comparison model	`nvidia/Qwen3.6-27B-NVFP4`, revision `0893e160…`
fp8 checkpoints	`Qwen/Qwen3.6-35B-A3B-FP8` revision `95a723d0…`, `Qwen/Qwen3.6-27B-FP8` revision `e89b16eb…`
Tools	`tools/bakeoff/determinism-probe.py`, `determinism-ladder.sh`, `determinism-logprobs.py`, `determinism-report.py`, `opdet/harness.py`
Raw data	all `.ndjson` files under `docs/messungen/determinism/` and its `opdet/` folder; positions as `.positions.ndjson.gz`
Public prompt set	`docs/messungen/determinism/public-prompts/`, CC BY 4.0
Full measurement report (German)	`docs/messungen/determinism/2026-09-19-leiter.md` (§ 1 to § 10)

Tools, the public prompt set, and the raw data of the public runs will be released under Apache-2.0 in a separate repository, with a single command that repeats the measurement against any OpenAI-compatible endpoint. The production prompt stays internal; this document describes it structurally.

About this document

Author: Christian Thiede, TACP Operations GmbH & Co. KG (Explicio), Detmold, Germany, ORCID 0009-0002-4517-7208. An AI assistant (Claude, Anthropic) contributed to measurement, analysis, and text; responsibility for content and numbers lies with the author. Statements about third-party software (vLLM, FlashInfer, DeepGEMM, NVIDIA containers, Qwen models) refer to the versions named and to the date of measurement. This document may be passed on unchanged.



Explicio®

Das digitale Gedächtnis Ihres Unternehmens

TACP Operations GmbH & Co. KG (Explicio) · Detmold