Chain Solutions

Own Your Data. Own Your Future.

Achieving Digital Independence with a Private Company Blockchain

May 2026

Chain Solutions

Geralde Passchier

Executive Summary

Enterprises today face mounting risks: AI-driven energy shortages pushing cloud costs higher, recurring hyperscaler outages, geopolitical vulnerabilities, and increasingly strict compliance requirements. When the cloud falters, business stops. It is therefore no surprise that cloud repatriation is accelerating, with organizations prioritizing predictable, sovereign operations over elastic but fragile cloud models.

The Chain Engine offers a direct path to digital independence. By rethinking the company's data layer and embedding a reimagined blockchain directly into software applications, it delivers a lightweight, immutable and event-sourced foundation that runs entirely under the company's control.

Built on a single-leader architecture with deterministic Proof-of-History sequencing and in-memory replicas, the Chain Engine combines strong performance with radical simplicity. It supports hundreds of transactions per second on standard hardware and enables 50-70% reductions in development and operational costs. Custom modules can be developed in weeks rather than months.

The engine is designed for seamless integration into existing IT landscapes through flexible embeddability options, straightforward vertical scaling, and built-in compliance features that simplify audits and regulatory requirements such as GDPR.

Looking ahead, the Chain Engine forms the foundation for the Chain AI Mesh, a vision of specialized, sovereign AI models that run on-premise, respect differential privacy, and deliver powerful business intelligence while maintaining full digital sovereignty.

Whether the goal is mitigating energy and geopolitical risks, reducing costs, accelerating development, or preparing for intelligent applications, the Chain Engine empowers organizations to repatriate critical workloads with controlled, sovereign technology.

Own Your Data. Own Your Future.

© 2026 Chain Solutions. All Rights Reserved.

Table of Content

© 2026 Chain Solutions. All Rights Reserved.

1. Introduction

Digital independence has never been more critical. While many organizations are repatriating workloads to regain control and cost predictability, true sovereignty requires more than moving applications on-premise. It demands full ownership of both data and the intelligence derived from that data. At the same time, regulations regarding data security and resilience are tightening, notably through NIS2 and DORA.

The Chain Engine provides the immediate foundation for both independence and compliance. By treating blockchain as a lightweight data layer within software applications, data becomes immutable, tamper-proof and fully under the company's control - eliminating vendor lock-in. Combined with a fully event-sourced application state, the need for complex reconciliations disappears, delivering compliance by design.

The engine runs efficiently on standard hardware and accelerates the development of core business processes through a highly standardized and modular architecture. All blockchain logic is handled by the engine in plain, auditable Rust functions. Developers can therefore focus immediately on company-specific business logic, without requiring blockchain expertise.

In this whitepaper we explain how the Chain Engine delivers immediate value today and how the Chain AI Mesh - our optional, built-in AI layer - represents the next evolutionary step toward complete digital independence: owning not only your data and infrastructure, but your future intelligence as well.

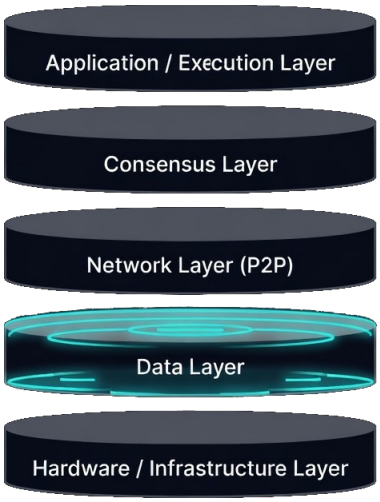
Own Your Data. Own Your Future.

© 2026 Chain Solutions. All Rights Reserved.

2. Applying Blockchain Technology in Applications

It is a widespread misconception that blockchain is only suitable for public or permissioned networks. While blockchain technology has so far been used primarily in public and permissioned environments, this does not mean it is limited to those contexts. On the contrary, it is precisely when blockchain is applied within a company or business application that the technology comes into its own.

The image shows the classic five layer architecture of public and distributed networks.



In this model, the Network Layer provides broad accessibility and connectivity, while the Consensus Layer offers mechanisms such as validation and leader rotation to keep the system secure in a public, trustless environment.

These layers, however, are not blockchain. Although the term “blockchain” has become synonymous with distributed networks, it is in fact only the data layer. Blockchain was originally chosen for the Bitcoin network because it provides a lightweight, append-only and tamper-proof data protocol - ideal for safely operating across thousands of nodes run by unknown and untrusted parties.

The data is written in blocks (bin-logs) that are linked together through cryptographic hashes, hence the name blockchain. Although this approach became widely known through its use in public networks, the same underlying data structure can be applied effectively in ordinary software applications. Companies can benefit from this secure and lightweight method of data storage by using blockchain as an internal data layer.

2.1 A Blockchain Fully under your Control

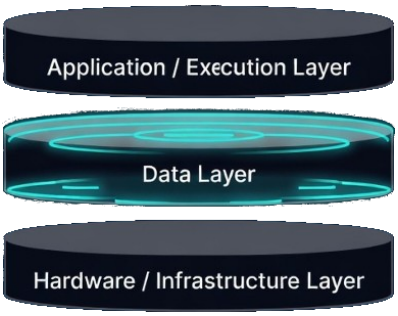
When blockchain technology is applied inside a company rather than across untrusted parties, its nature changes completely.

In public, trustless environments, consensus mechanisms, leader rotation, peer-to-peer networking and economic incentives (such as gas and staking) are essential to reach agreement and protect the system against malicious actors. These layers exist because participants neither trust each other nor control the infrastructure.

Inside an organisation, the situation is fundamentally different. The company controls the hardware, the deployment, the network, and the access policies. The environment is trusted by design. In such a setting, distributed consensus and leader rotation do not add meaningful security, on the contrary they would only introduce complexity and performance overhead. In trustful environments errors are not Byzantine faults that must be tolerated through voting; they are bugs that must be fixed, just like in any other software system.

© 2026 Chain Solutions. All Rights Reserved.

What remains - and what actually matters - is the core property of blockchain: an append-only sequence of cryptographically hashed events. This creates an immutable, tamper-evident ledger without requiring a network or consensus layer. In other words, blockchain becomes a data storage method inside software applications.



By removing the layers that were only necessary for public, trustless networks, the Chain Engine treats blockchain as what it fundamentally is: a lightweight, immutable event log that can be embedded directly into ordinary business applications. In this setup the Network and Consensus Layer are removed entirely from the architecture.

The result is a system that delivers the benefits of blockchain (single source of truth, compliance by design, auditability), while remaining simple, predictable and efficient to operate on standard infrastructure.

This shift in perspective, from blockchain as network technology to blockchain as internal data layer, is the foundation of the Chain Engine.

© 2026 Chain Solutions. All Rights Reserved.

3. Chain Solutions’ Blockchain Engine

Chain Solutions' blockchain engine is a prime example of what a blockchain application within a company should look like. It is a modular yet simple architecture that fulfills all the promises of blockchain, with a special focus on compliance and auditability. It enables companies to accelerate the development of blockchain applications and, consequently, to move workloads from the cloud to on-premises at an accelerated pace. See appendix B for an architectural overview. The most important components and their key characteristics are displayed below.

3.1 The Execution Engine & Ledger

The Execution Engine is the central heartbeat of the Chain Solutions architecture, the only component authorized to produce new immutable events and maintain the single source of truth. It implements a lean, permissioned private blockchain design with a single leader node, eliminating distributed consensus overhead (no distributed consensus, no voting, no Byzantine overhead). It sequences events using 400 ms (configurable) Proof-of-History (PoH) slots for sub-second finality and verifiable ordering. All business writes, whether originating from a public API, internal service, or direct embedded call, eventually reach the engine as transactions. The engine validates, sequences, executes, and persists them atomically.

3.2 Replica

The In-Memory Replica is the read-optimized, high-speed mirror of the application's current global state. It holds all business data (balances, memberships, opportunities, etc.), entirely in RAM, with no persistent storage for hot paths and minimal index bloat. It serves reads for both frontend clients and business logic. The Internal API (trusted network only) acts as a thin shim/facade around this state, providing a stable interface for queries. For ultra-low-latency internal paths, direct calls to the global state are supported. Replica is event-sourced realtime as well as on startup or recovery via the Replica Replay feature. In single-binary deployments Replica runs in-process with the execution engine, enabling direct function calls with zero serialization or network overhead.

3.3 Audit.log

The audit.log is crucial for the compliance-by-design feature of the blockchain engine. To reinforce this, the engine comes with two stand-alone auditor binaries.

The audit-decrypt binary decrypts the runtime-encrypted audit.log and the merkle-proof binary generates and verifies merkle inclusion proofs for events in the ledger. It (re-)computes the leaf has and reconstructs the merkle tree. The audit.log is:

- Human readable, encrypted at rest

© 2026 Chain Solutions. All Rights Reserved.

- Hash chained
- Last hash written to ledger with each entry (each bin)
- Periodic rotations (allows for file management and archving)
- Active audit.log recovery upon application restart/redeployment

3.4 Business Logic Library (BLL)

All business logic, including data validation, enrichment, transformation and decision-making, resides exclusively in the Business Logic Library (BLL). This centralisation is a fundamental architectural invariant. It eliminates duplication, ensures a single source of truth for domain rules, and greatly simplifies auditing, compliance and dispute resolution.

The BLL is deliberately split into two layers:

Policies
Policies are the main entry points for each write flow (for example create_company_policy). They orchestrate the complete sequence of validations, calculations and decisions required for a specific event type. A policy contains the full overview of “what must happen”: input normalisation, existence checks, compliance rules, event creation and more. Because all logic for one event type lives in a single, compact file, policies are easy to review, audit and reason about.

Services
Services are small, stateless helper functions that policies call upon (for example validate_and_normalize_company_input, check_company_exists or generate_pubkey). They perform focused, side-effect-free operations and return simple results (Ok/Err or computed values). Services are easy to unit-test in isolation and can be reused across multiple policies and components.

This clear separation between Policies and Services delivers several important advantages:

- Zero logic drift - There is only one place where domain rules are defined.
- High auditability - All business rules for an event type can be reviewed in a single file.
- Excellent maintainability - The code remains clear and structured, even as the application grows.
- Strong testability - Services can be tested independently, while policies can be tested at a higher level.

The BLL is written in plain Rust, just like the rest of the Chain Engine. This choice brings strong compile-time guarantees, excellent performance, and a robust type system. For most organisations this is not a barrier, as Rust is increasingly adopted in systems programming and backend development.

© 2026 Chain Solutions. All Rights Reserved.

It is technically possible to implement the BLL in another language using Foreign Function Interfaces (FFI). However, doing so introduces significant friction. The engine and the BLL share a common set of core data types (events, structs, identifiers, error types, etc.). These **shared types** form the single source of truth for data representation across the entire system and play a key role in the prevention of developer mistakes. When the BLL is written in the same language as the engine, these types can be used directly. In a multi-language setup, maintaining consistency between the shared types and the BLL becomes a recurring source of friction and potential errors. For this reason, we strongly recommend implementing the BLL in Rust.

Once the business logic has been developed in the BLL, integrating it with the engine requires very little effort. In a typical case, connecting a new BLL to the engine takes only one to two hours on the backend. This low integration cost, combined with the clear separation of concerns, allows development teams to focus almost entirely on business logic rather than on infrastructure or blockchain mechanics.

3.5 The Unified Write Transaction Flow

All write operations in Chain Solutions applications are handled through a single, standardized RPC endpoint (`POST /rpc`) in the Public API. This extreme standardization is one of the key drivers behind the exceptionally high development speed of the engine.

The frontend sends a `TransactionRequest` containing the usual JSON-RPC fields (`jsonrpc`, `method`, `params`) together with a caller identifier. The Public API immediately generates a unique, cryptographically secure request ID using Blake3 and returns it to the client. At the same time, it enriches the request into a full `Transaction` object and forwards it asynchronously to the Execution Engine.

The Rust structure of a `Transaction` looks as follows:

```
pub struct Transaction {
    pub jsonrpc: String,      // "2.0"
    pub id: [u8; 32],         // Blake3-based request ID
    pub caller: Caller,
    pub method: String,       // e.g. "chain.create_company"
    pub params: Vec<u8>,      // serialized business wire
}
```

Because the response (containing the request ID) is returned immediately, the client experiences low latency. The client can use the returned request ID to poll the status of the operation (i.e. whether the resulting event has been applied to the in-memory Replica). The request ID is also stored together with the final event in the immutable ledger, ensuring full traceability from the original client request to the committed state.

On the engine side, generic program handlers receive the serialised parameters and delegate them directly to the corresponding function in the Business Logic Library (BLL). The handlers themselves contain virtually no business logic; they act as thin, typed adapters between the engine and the BLL.

© 2026 Chain Solutions. All Rights Reserved.

Internal Processing Model

Write processing inside the engine follows a two-stage model:

1. **Execution stage**
   Incoming transactions are placed in a FIFO queue. The Engine Manager processes them sequentially. For each transaction it calls the relevant program handler, which in turn executes the policy and services in the BLL. The resulting event (if any) is placed in a pending events queue. Heavy or complex business logic in the BLL can therefore increase the time until an event is produced, but it does not block other transactions from entering the system (async function calls).

2. **Sequencing and persistence stage**
   The main slot loop (running on deterministic 400 ms Proof-of-History slots) takes events from the pending queue, assigns them a slot, and appends them to the ledger. Only at this point does the event become final and visible to the Replica.

Because ordering and persistence happen asynchronously after BLL execution, the system can maintain good responsiveness and throughput even when individual policies contain relatively heavy logic. For the large majority of enterprise use cases this model provides sufficient performance on a single leader. Only in very specific high-throughput scenarios (for example industrial process logging at large scale) is horizontal scaling across multiple engines required.

Benefits of the Unified Write Path

This architecture delivers several important advantages:

- A single, battle-tested write path for every operation.
- New features or applications are created primarily by extending the BLL; the engine, RPC layer and core slot loop remain unchanged.
- Immediate request ID response combined with asynchronous processing enables fast user feedback without sacrificing determinism.
- The engine comes with clear instructions and AI-assisted prompts that can generate the boilerplate for a new write flow (including policy and service stubs) in a very short time.

As a result, development teams can add new business functionality at high speed while the core system remains stable and auditable.

© 2026 Chain Solutions. All Rights Reserved.

4. Embeddability

The Chain Engine is designed to integrate smoothly into existing enterprise IT landscapes without requiring a rip-and-replace approach. It achieves this through a clean, trait-based extension architecture that allows organisations to plug in their own implementations for key infrastructure concerns.

Instead of hard-coding dependencies on specific identity systems, storage backends or secret managers, the engine exposes well-defined extension points. This enables companies to reuse their existing investments in identity and access management, document storage, logging platforms and secret management, while keeping the core engine unchanged.

The main extension points include:

- **Identity Provider (IdP) and RBAC** - connect to existing identity providers and role-based access control systems.
- **Document Storage** - abstract document and binary storage so it can be backed by local storage, MinIO, S3, Azure Blob or other solutions.
- **Secrets and Cryptographic Providers** - pluggable providers for secrets and cryptographic algorithms, making the engine crypto-agile.
- **Logging and Telemetry** - forward logs, audit events and metrics to the organisation’s preferred observability stack.

Because these extension points are implemented as traits, custom integrations can be developed independently of the core engine. In most cases, connecting an existing enterprise system requires only the implementation of a specific provider and a small amount of configuration.

In addition, many operational and security settings can be configured at runtime through a central configuration file. This includes slot timing, cryptographic providers, ledger compression and encryption, snapshot policy, audit log rotation behaviour, document storage settings and JWT token lifetimes. These configuration options allow organisations to adapt the engine’s behaviour to their specific environment and compliance requirements without modifying or recompiling the application.

This combination of trait-based extension points and runtime configuration makes it possible to adopt the Chain Engine incrementally while respecting existing infrastructure, security policies and operational practices.

© 2026 Chain Solutions. All Rights Reserved.

5. Performance and Scaling

The Chain Engine is built around a single-leader architecture with a deterministic (but configurable) 400 ms Proof-of-History slot. All writes are processed sequentially by the Leader Node, while reads are served from a fully in-memory Replica.

The Manager processes transactions one-by-one (FIFO queue), so a complex policy (e.g. 12 ms) temporarily delays the next transaction in the queue. This sequential model is a deliberate architectural choice: it guarantees perfect ordering and makes every transaction fully deterministic and auditable.

Measured Performance (April 2026)
On a modern laptop (Intel Core Ultra 7 155U, 14 cores, 32 GB RAM, PCIe 4.0 NVMe SSD) we performed parallel bulk tests creating 1000 companies with mandatory fields only (name + country code), including validation, normalization, Replica lookups and duplicate-name checks (low complexity).

- Single client test (no concurrency): ~30–35 TPS
- Parallel client test (CONCURRENCY 20–50): 250–280 TPS (≈100 transactions per 400 ms slot)

These numbers reflect relatively lightweight CRUD-style operations. Workloads with heavier business logic in the BLL will naturally reduce throughput. However, the impact is often less severe than a purely sequential model would suggest.

The Engine Manager processes transactions sequentially in FIFO order, meaning it starts them one after another. However, the execution programs are asynchronous which means Manager does not wait for a transaction to complete BLL execution before starting the next one. As a result, multiple transactions can be in flight within the BLL at the same time. A transaction with lighter business logic that was started later can therefore finish BLL processing before a heavier transaction that was started earlier. Its resulting event can reach the pending events queue and the Proof-of-History slot loop ahead of the earlier transaction. This pipelined execution behaviour helps maintain good overall throughput, even when the complexity of individual policies varies significantly.

© 2026 Chain Solutions. All Rights Reserved.

5.1 Scaling Strategies

Vertical Scaling (Recommended)
For the vast majority of workloads, vertical scaling is the simplest and most effective approach. Because write processing is single-threaded in the Engine Manager, scaling is sub-linear but still very effective in practice.

Scale Level	**vCPU**	**RAM**	**Expected TPS** (medium complexity)
Development	4	8 GB	40 – 80
Small Production	8	16 GB	80 – 150
Medium Production	16	32 GB	120 – 220
High Performance	32+	64 GB+	200 – 400

Advantages of vertical scaling:

- No architectural changes required
- Full auditability and single source of truth preserved
- Extremely simple deployment, monitoring and operations

Horizontal Scaling (Advanced use cases only)
Horizontal scaling patterns, such as functional sharding or multi-leader setups, are only relevant for extreme throughput requirements (typically well above 500–2,000 TPS) or true geo-distribution. These approaches increase operational complexity and are rarely needed for standard enterprise applications. Horizontal scaling of the Replica can be achieved using gossip, but requires a transition to a multi-binary deployment.

Documentation and code examples for multi-binary deployments are provided with the engine, including patterns for multicast UDP-based gossip and mempool usage.

In short: start with vertical scaling. Only move to horizontal patterns when you have a clear, quantified need that cannot be solved by adding more CPU and RAM to a single leader.

© 2026 Chain Solutions. All Rights Reserved.

Hardware Recommendations (Production)

Requirement	Minimum	Recommended	High Performance
CPU	4 vCPU	8–16 vCPU	32+ vCPU
RAM	8 GB	16–32 GB	64 GB+
Storage	NVMe SSD	Fast NVMe SSD	High-IOPS NVMe
Network	1 Gbps	10 Gbps	10+ Gbps

Best Practices

- Keep the hot application state (Replica) entirely in RAM. Place the ledger, audit logs and snapshots on fast local SSD storage. Older data can be safely archived to cold storage thanks to periodic snapshots and audit log rotation.
- Keep the Business Logic Library clean and efficient. Avoid unnecessary allocations, blocking I/O and heavy computations inside the critical write path.
- Offload heavy or slow operations (complex reports, external API calls, AI enrichment, etc.) to background workers where possible.
- Monitor ledger growth and rotate audit logs regularly.
- Start with vertical scaling. Only introduce horizontal scaling patterns when there is a proven need that cannot be met by adding resources to a single leader.

© 2026 Chain Solutions. All Rights Reserved.

6. Future vision: Chain AI Mesh

At Chain Solutions, true digital independence ultimately also extends to intelligence. That is why we are developing the **Chain AI Mesh**, a suite of specialized, interconnected large language models purpose-built for the Chain Engine.

It is important to note that the use of AI components within the Chain AI Mesh is entirely optional. Organisations can fully operate the Chain Engine without adopting any AI functionality. The engine itself remains lightweight and efficient on standard hardware, independent of AI. Companies can choose to adopt AI modules, such as development assistance or business intelligence, selectively and at their own pace, depending on their needs and available infrastructure.

Instead of relying on monolithic AI systems or external cloud providers, we have chosen a meshed architecture. This design strictly limits compute and energy consumption to what is necessary, while guaranteeing that any model with access to proprietary company data runs fully on-premise. All inter-model communication is governed by **differential privacy**: only LoRA adapter updates, aggregated metadata, and derived business metrics are ever shared externally. Raw event data never leaves the customer's premises.

Because sovereignty is a core principle, we do not offer critical AI components that process sensitive company data as AI-as-a-Service (AAAS). Only certain non-sensitive capabilities, such as development assistance, are made available in an on-demand AAAS model. All other models that work with actual business data are designed to run on-premise or in a federated setup under the customer's control.

The Chain AI Mesh is a multi-year vision that will be rolled out in phases, aligned with the growing adoption of the Chain Engine. The following models have already been identified for the core AI Mesh:

Name	LLM-type	Setup	Status
Chain AI Development	multi-LoRA Agentic	AAAS on demand (federated / on premise deployment possible)	In development
Chain AI Business Intelligence	multi-LoRa Traditional	On premise - federated	Planned
Chain AI Custom	multi-LoRA Agentic	On premise	Customer specific workflows
Chain AI Benchmark	Combi: Trad + Agentic	Centralized	Future
Chain AI Economics	Traditional	Centralized	Horizon
Chain AI Language	Traditional	Centralized	Buy(?)

© 2026 Chain Solutions. All Rights Reserved.

Vertical AI modules

Name	LLM-type	Setup	Status
Chain AI Accounting	(multi-LoRA) Agentic	On premise - federated	Planned
Chain AI Supply Chain	multi-LoRA Agentic	On premise - federated	Future
...			

6.1 Description of the AI-modules

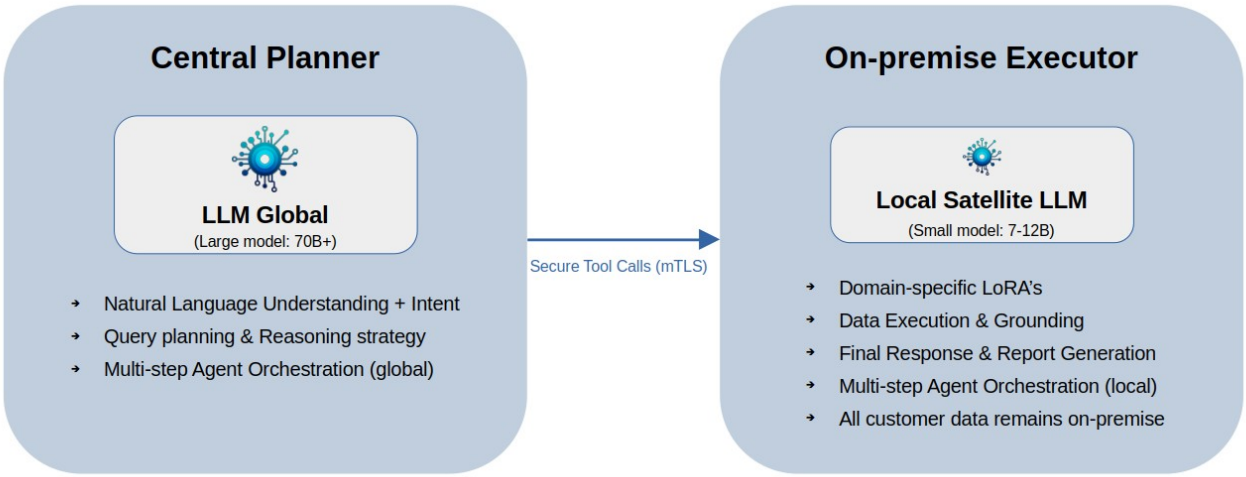
Chain AI Development
Chain AI Development is a sovereign, human-triggered AI developer assistant built on Qwen2.5-Coder-32B. It executes sophisticated agentic workflows specifically designed to design event structures, write flows, policies, and micro-services for the Business Logic Library.

By default, it is offered as a secure, pay-per-usage AI-as-a-Service (AAAS, pronounce as triple-A, S) on demand, making powerful AI-assisted development immediately accessible without any hardware investment for the vast majority of users. For organizations with intensive development cycles or those creating tailored commercial binaries, a fully local federated deployment option is available (minimum NVIDIA RTX 4090 with 24 GB VRAM or equivalent). The model shares only LoRA adapter updates with the central models and draws upon Chain AI Language for multilingual support. This will dramatically accelerate application development while keeping all intellectual property fully under the company's control.

Chain AI Business Intelligence
Chain AI Business Intelligence will be the model most frequently used by companies running the Chain AI Mesh. It combines a powerful central orchestrator with a lightweight on-premise data satellite, delivering sovereign, context-aware business intelligence directly from the immutable, event-sourced ledger while preserving the lightweight nature of the core Chain Engine.

Pre-trained on industrial Chain Engine templates with realistic simulated data, the system answers natural-language questions such as "What is our YTD turnover?" or "Show a breakdown of last month's top 10 customers by revenue." For example, when a user asks for revenue per product last month plus year-to-date figures and customer purchase history, the central orchestrator handles natural language understanding, query planning, multi-step reasoning and final report generation. It is powered by strong reasoning models such as Qwen2.5-72B or DeepSeek-R1. The on-premise BI Data Satellite executes all data access securely against the local in-memory Replica and ledger, applying specific LoRA adapters where needed for domain expertise. The system autonomously triggers agent workflows for analytical reports, forecasting, and regulatory compliance.

© 2026 Chain Solutions. All Rights Reserved.



Chain AI Business Intelligence - Hybrid Architecture

Asymmetric Learning Flow in the Hybrid Architecture

Because all user input first reaches the Central Orchestrator, the learning process is asymmetric. The central model primarily pushes improvements (base models and general LoRAs) to the on-premise Satellite. In return, the central model learns from local usage only through aggregated, differentially private signals, never through direct customer-specific LoRA updates. This design protects data sovereignty while still enabling ecosystem-wide learning over time.

Direction	Type of Update	How it Works	Direct / Indirect	Privacy-Preserving	Notes
Central → Local	Base model + general LoRAs	Improvements made centrally are distributed to the on-premise Satellite	Direct	Yes	Main flow
Central → Local	Improved planning prompts & strategies	The central planner becomes smarter, resulting in better execution plans for the Satellite	Indirect	Yes	Strong effect
Local → Central	Customer-specific LoRA deltas	**Not** sent back directly to the central model	-	-	Intentionally avoided
Local → Central	Aggregated patterns & success signals	Shared via differential privacy (e.g. which query types work well, effective report structures)	Indirect (delayed)	Yes	Present, but only in aggregated form

Chain AI Custom

Chain AI Custom enables organizations to create highly tailored agent workflows that reflect their unique business processes and requirements. These workflows can be developed internally or with support from Chain Solutions consultancy services. It draws intelligence from Chain AI Business Intelligence, Chain AI Benchmark, and Chain AI Language.

© 2026 Chain Solutions. All Rights Reserved.

Chain AI Economics – A Derived Ecosystem Product

Chain AI Economics represents a powerful long-term derivative of the Chain AI Mesh. As adoption of the Chain Engine grows - with the ambition that it becomes the new standard for enterprise IT within 5 to 10 years - the ecosystem will generate an unprecedented volume of ultra-clean, real-life, event-sourced business data. This model will provide deep macroeconomic insights and trend analysis by learning from the differentially private, aggregated signals shared by participating Chain AI Business Intelligence instances. Because the underlying data is tamper-evident, chronologically perfect, and free of the noise and reconciliation errors typical of traditional systems, Chain AI Economics has the potential to become a uniquely reliable source of truth for policymakers, economists, econometricians, journalists, and analysts. Unlike the other models in the Mesh, Chain AI Economics is not intended as a day-to-day tool for operating companies. It is a separate, high-value product developed and commercialized by Chain Solutions. Its value scales directly with the size of the network: the more organizations run the Chain Engine, the richer and more accurate the macroeconomic intelligence becomes.

Supporting Models

- Chain AI Benchmark supplies comparative market intelligence on request. It learns from differentially private insights shared across the ecosystem, enabling companies to benchmark market share, growth rates, and competitive positioning.
- Chain AI Language is a dedicated facilitation model focused on high-quality translations. Rather than developing this capability internally, we are evaluating the acquisition of a mature, production-ready model.

Chain AI Accounting

Chain AI Accounting will be the first vertical AI module built on top of the Chain AI Mesh. Designed as an agentic system, it enables organisations to automate and support core accounting processes directly from the immutable, event-sourced ledger.

The module assists with a range of operational and analytical tasks, including automated invoice matching, intelligent journaling of bank statements, reconciliation processes, and the generation of audit-ready financial reports. By combining the structured data of the Chain Engine with specialised agentic workflows, Chain AI Accounting delivers accurate, explainable, and fully traceable accounting intelligence while maintaining full data sovereignty.

The Path Forward

The Chain AI Mesh embodies the same core principles as the Chain Engine: full digital sovereignty, radical efficiency, privacy by design, and compliance without compromise. By combining on-premise execution for sensitive workloads with federated, privacy-preserving collaboration, it creates genuine collective intelligence that belongs to the enterprises that generate the data - never to a central cloud provider.

© 2026 Chain Solutions. All Rights Reserved.

This meshed, sovereign approach not only minimizes energy consumption and infrastructure cost, but also unlocks unprecedented development speed and strategic insight. From AI-assisted coding of new write flows to real-time business intelligence, industry benchmarking, and - at scale - macroeconomic intelligence, the Chain AI Mesh transforms the Chain Engine into a complete sovereign intelligence platform.

© 2026 Chain Solutions. All Rights Reserved.

7. About Chain Solutions

Chain Solutions develops private blockchain technology that enables organisations to regain control over their critical business data. The technology combines the benefits of blockchain with the performance, simplicity and embeddability of modern enterprise software.

Instead of offering Software-as-a-Service, Chain Solutions works for the engine exclusively with one-time licenses. This ensures that customers become full owners of their IT infrastructure and data, a core principle that we consider very important.

The company was founded by Geralde Passchier, who brings over 25 years of experience leading complex ICT transformation projects at financial institutions. During a sabbatical she started exploring blockchain technology on public networks. Shocked by the complexity and limitations she encountered, she took the project offline and began reimagining blockchain as a lightweight, internal data layer. The result is the Chain Engine: a pragmatic, high-performance blockchain engine designed specifically for enterprise use cases that demand sovereignty, auditability and development speed.

© 2026 Chain Solutions. All Rights Reserved.

Appendix A: Chain Engine and GDPR

Chain Engine is designed with data privacy principles at its core, particularly when handling personally identifiable information (PII) in enterprise applications. As a private application blockchain, it provides strong protections without relying on complex mechanisms such as zero-knowledge proofs.

PII is stored on the ledger only when operationally necessary. Multiple layers of safeguards are in place to support compliance with GDPR, CCPA and ISO 27001.

Key Privacy Features

The system is designed to comply with data privacy regulations by design:

- **Role-Based Access Control (RBAC) and Need-to-Know** - The engine integrates with existing user management systems. Access to data is strictly limited according to roles (aligned with ISO 27001 A.8.3). Field-level controls can be added where needed.
- **Encryption** - Data is encrypted at rest (AES-256) and in transit (TLS 1.3). The architecture is designed to support a future migration to post-quantum encryption.
- **Data Minimisation** - Only essential fields are stored on-chain.
- **Immutable Audit Trail** - All actions are logged immutably with restricted access. The audit log is encrypted and protected by continuous cryptographic hashing.

Right to Erasure (Art. 17 GDPR) and Storage Limitation (Art. 5(1)(e))

The append-only nature of the ledger prevents physical deletion of historical events. This preserves auditability and data integrity. However, the architecture supports functional erasure through the following mechanisms:

Logical Deletion:

IAn event such as `EntityExpired` or `ContactDeactivate` (including a timestamp and reason) can be implemented. During Replica state rebuilding and queries, expired entities are deleted / filtered out, making them invisible and inaccessible in all business flows (UI, APIs, reports).

Replica Snapshots:

The system supports configurable periodic snapshots of the Replica state (default: every 10,000 events). On restart or replay, only the snapshot + events after the latest snapshot are applied. This means pre-snapshot events containing expired PII are no longer reloaded into memory, effectively removing them from operational access.

© 2026 Chain Solutions. All Rights Reserved.

Audit Log Retention:

The audit.log and ledger retain historical records for justified purposes (e.g. fraud detection, legal obligations under Art. 17(3) GDPR, or other regulatory requirements). These files are encrypted at rest, access-restricted, and protected by continuous cryptographic hashing.

To balance long-term auditability with GDPR storage limitation requirements (Art. 5(1)(e)), the system supports periodic audit log rotation - for example annually or when a configurable size threshold is reached.

When rotation occurs:

- A new audit log file is started.
- The final cumulative hash of the previous log becomes the genesis hash for the new log, preserving the unbroken cryptographic chain of integrity.
- The old log file is sealed and moved to an encrypted archive.
- Once the required retention period expires (as defined in the organization’s data retention policy), the archived logs can be securely deleted without affecting the verifiability of newer records.

This mechanism enables progressive cleanup of historical audit data while maintaining full tamper-evidence and compliance with both audit and data minimization obligations.

Recommendations for Your DPIA

When performing a Data Protection Impact Assessment (DPIA) for deployments involving PII, include a dedicated section on erasure in append-only storage.

Identified Risk: Physical deletion is impossible in the append-only immutable ledger which raises a potential tension with the Right to Erasure.

Mitigations:

- Logical deletion via expiry events combined with Replica filtering and deletion makes the particular state is no longer queryable or usable.
- Periodic snapshots with replay-from-snapshot. This means pre-expiry data is not reloaded from configurable intervals, which makes the particular state no longer resides in the operational program.
- Restricted, encrypted audit.log during the legal retention in combination with periodic audit log rotation with chained hashing → enables secure deletion of aged archives after retention expiry, further reducing storage of potentially identifiable historical data.
- Optional enhancements: off-chain PII storage (hashes/references on-chain) or cryptographic erasure (destroy encryption keys on expiry).

© 2026 Chain Solutions. All Rights Reserved.

Residual Risk: Low; data is functionally inaccessible for processing; theoretical recovery limited to sealed audit archives under controlled access during retention periods only.

Conclusion: Proportional and effective compliance for private enterprise systems, aligning with EDPB and national DPA guidance on permissioned blockchains.

This approach balances strong auditability with privacy rights. Consult your Data Protection Officer or legal expert for deployment-specific reviews.

© 2026 Chain Solutions. All Rights Reserved.

Appendix B: Post-Quantum Readiness

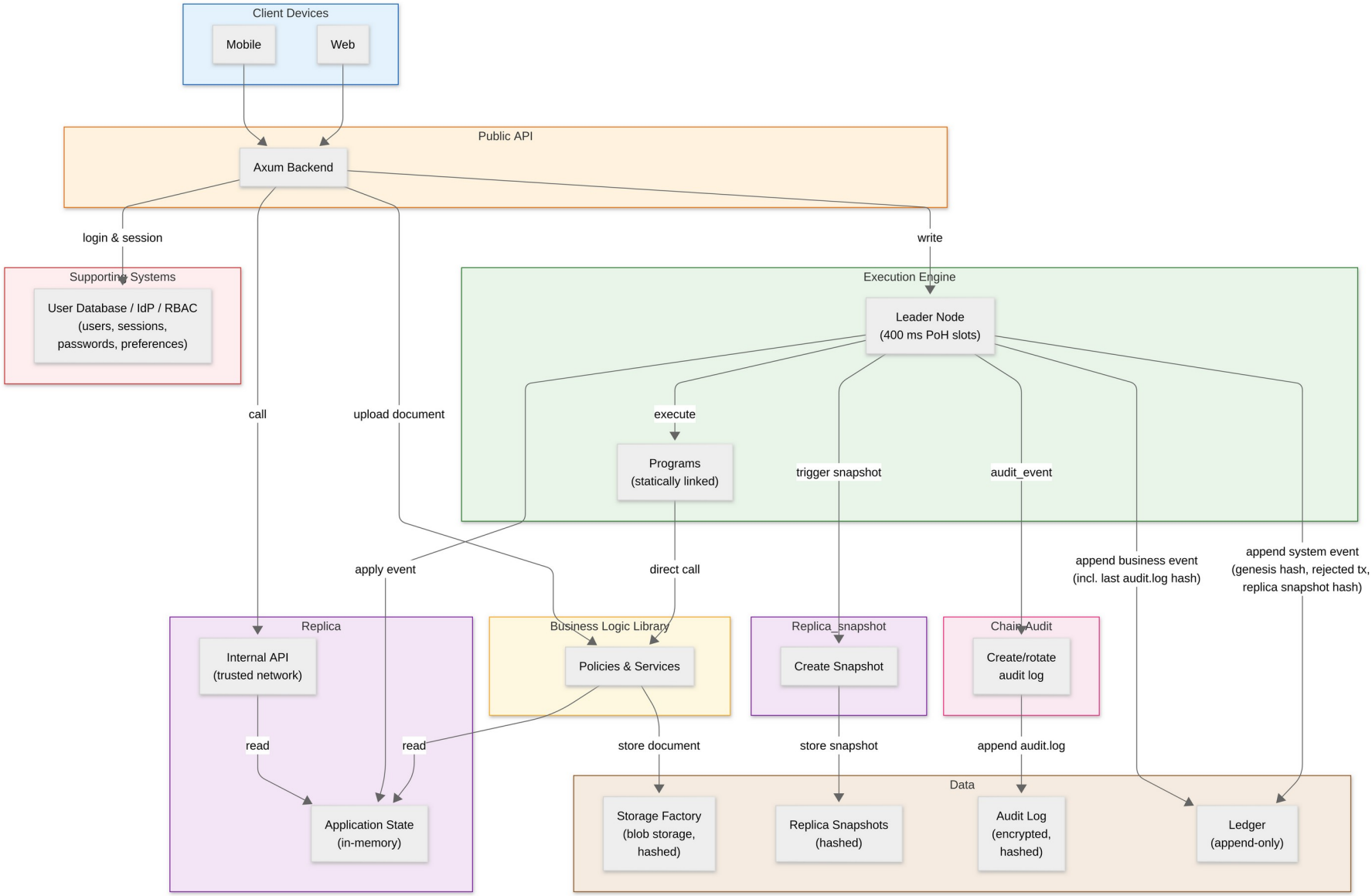
The Chain Engine currently uses industry-proven cryptographic primitives: Blake3 for high-performance hashing and AES-256 for encryption of data at rest. All cryptographic operations are implemented through clean, configurable traits and provider interfaces.

This abstraction layer ensures that organizations can seamlessly switch to post-quantum cryptographic algorithms, including quantum-resistant hashing functions, as soon as these standards reach sufficient maturity and receive broad adoption. No code changes or architectural rework are required, instead simply replace the cryptographic provider.

By building post-quantum readiness into the core engine today, Chain Solutions future-proofs sensitive enterprise data against emerging quantum computing threats while maintaining full backward compatibility with current standards (AES-256 and TLS 1.3).

© 2026 Chain Solutions. All Rights Reserved.

Appendix C: Chain Engine Architecture



© 2026 Chain Solutions. All Rights Reserved.