ISSN 2278-9723

Research Article

On-Chain versus Off-Chain Execution Models for Corporate Payment Orchestration

Naren Swamy Jamithireddy
Jindal School of Management, The University of Texas at Dallas, United States
Email: naren.jamithireddy@yahoo.com
Received: 17.01.16, Revised: 16.03.16, Accepted: 22.05.16

ABSTRACT
Corporate payment orchestration has traditionally relied on off-chain middleware systems to coordinate approvals, settlements, and exception handling across ERP platforms and banking networks. However, the emergence of blockchain-based execution frameworks introduces an alternative model that offers cryptographic finality, tamper-resistant audit trails, and external verifiability through on-chain state commitments. This article presents a comparative evaluation of on-chain versus off-chain execution models for corporate payment flows, analyzing differences in settlement finality enforcement, latency behavior, reconciliation dynamics, and audit assurance strength. Through state transition mapping and batch-size latency profiling, we demonstrate that while on-chain execution enhances transparency and non-repudiation, it incurs higher processing overhead than off-chain orchestration. We propose a hybrid deployment strategy in which operational workflows remain off-chain for efficiency, while finalized payment states are periodically anchored on-chain to ensure regulatory-grade integrity. This balanced architecture maintains performance while enabling trustless verification and improved compliance readiness.

Keywords: Corporate Payments, On-Chain Settlement, Off-Chain Orchestration, Hybrid Execution Model

1. INTRODUCTION

Corporate payment orchestration refers to the coordinated sequencing, approval, and execution of outbound payment instructions across internal financial systems, bank networks, and settlement infrastructures. In traditional enterprise environments, payments originate from enterprise resource planning (ERP) platforms such as SAP FI/CO or Oracle Financials, which generate batch-based payment proposals processed via bank communication gateways or treasury management middleware [1]. These workflows historically rely on centralized control logic, fixed approval chains, and institution-specific message standards such as SWIFT MT/MX or host-to-host encrypted channels [2]. However, the emergence of programmable blockchain networks introduced the possibility of shifting certain execution steps into decentralized, verifiable, and state-managed environments, altering traditional orchestration assumptions [3].

The motivation for reconsidering execution models arises from the operational and audit risks associated with centralized workflow dependencies. In conventional off-chain orchestration, correctness and traceability depend heavily on the integrity of internal logs, middleware reliability, and segregation-of-duties enforcement across user roles [4]. Payment failure scenariossuch as delayed settlement acknowledgments, duplicated release triggers, or inconsistent batch statuses across ERP and bank systemsrequire multi-system reconciliation procedures that can be time-consuming and prone to interpretation variability [5]. Corporate treasury teams face rising expectations for transparent, tamper-resistant, regulator-auditable payment records.

In contrast, on-chain payment orchestration encodes release logic, approval gating, and settlement conditions within smart contracts or ledger-based state machines. This model ensures that payment instructions and resulting state transitions are cryptographically verifiable and replicated across distributed nodes rather than maintained solely in internal corporate systems [6]. The execution outcome is globally consistent, reducing ambiguity in cross-system reconciliation. However, the tradeoff is exposure to public network settlement latency, gas fees, and protocol-level finality constraints.

The execution location of orchestration logic therefore directly impacts latency, finality behavior, auditability, and error recovery. On-chain execution ties payment finality to

blockchain confirmation depth, where settlement is cryptographically irreversible beyond a consensus threshold [7]. Off-chain orchestration achieves faster response times but depends on internally governed retry and rollback workflows that require strong operational discipline to ensure consistency. Organizations must evaluate which execution environment aligns with their performance, compliance, and risk management objectives.

Another consideration is the granularity of state awareness. On-chain orchestration maintains a shared global source of truth, eliminating divergence between internal ledger systems and external banking networks when tokenized or ledger-native assets are involved [8]. In contrast, off-chain orchestration retains ERP or treasury middleware as the primary system of record, with downstream reconciliation required to align external settlement confirmations.

Corporate environments also exhibit heterogeneity in infrastructure maturity. Firms operating across multiple jurisdictions, currencies, and banking partners often rely on integration hubs or multi-bank connectivity providers to abstract communication differences. Introducing on-chain execution into such environments requires mapping corporate payment semantics to deterministic, verifiable state transitions that respect regulatory frameworks such as AML/KYC controls and regional settlement rules [9]. This introduces additional system design constraints that must be weighed against the benefits of immutability and audit-strengthened execution.

Given the variance in organizational requirements, infrastructure architecture, and regulatory boundaries, the evaluation of on-chain versus off-chain execution models is not a binary replacement decision but a design-space analysis. Hybrid orchestration patternswhere release authorization is handled on-chain while operational workflows remain off-chainare increasingly considered viable, provided the division of control preserves traceability and accountability. The remainder of this article analyzes system models, state reconciliation behavior, performance characteristics, and operational tradeoffs to establish deployment guidance.

2. Corporate Payment Execution Landscape

Corporate payment execution in enterprise environments is driven primarily from ERP financial modules responsible for accounts payable, treasury disbursements, vendor settlements, intercompany transfers, and liquidity positioning. In systems such as SAP FI/CO, payments are typically collected into proposal lists based on due dates, payment terms, and approval workflows that determine which documents are eligible for release. Once validated, the ERP executes a payment run that aggregates approved items into a structured batch and assigns payment media formats or bank transmission instructions depending on configured payment methods and routing rules.

After batch creation, the payment instructions are routed to communication or orchestration layers such as SAP Bank Communication Management (BCM), Oracle Payment Hub, Kyriba, TMS platforms, or proprietary bank gateway middleware. These systems handle formatting into SWIFT MT/MX messages, ISO 20022 XML payment instructions, or proprietary EBICS/host-to-host payloads. The goal of this stage is to normalize ERP-originated payment semantics into bank-acceptable transmission formats while ensuring consistent logging of submission time, sender identity, and routing details.

In many organizations, these payment instructions are processed in scheduled intervalsdaily large-value treasury settlements, periodic vendor disbursements, payroll releases, or real-time urgent cash movements. The settlement submission frequency affects liquidity availability and intraday cash visibility. Gateways must maintain reliable channel connectivity and status acknowledgment processing to prevent operational blind spots, such as payments marked as released in ERP even if the receiving bank has not yet confirmed acceptance.

Once transmitted to the bank or payment network, acknowledgement messages are returned through the same communication channels. Depending on the bank, the network, and the region, acknowledgements may be synchronous or asynchronous, immediate or delayed. ERP or TMS systems update the originating payment documents with these return statuses, marking transactions as *Accepted*, *Pending*, *Rejected*, or *Partially Processed*. This information propagates back to treasury dashboards, liquidity planning modules, and general ledger clearing accounts.

However, reconciliation between the ERP ledger state and the bank's settlement ledger is not instantaneous. Confirmed settlement may require matching payment confirmation files, returned bank statements, or SWIFT CAMT.054 / MT942 transaction reports. Delays or mismatches can result in temporary accounting discrepancies, necessitating exception workflows or manual

resolution procedures within the treasury operations team.

A critical component of the payment landscape is audit traceability. ERP systems generate logs that document when payments were initiated, by whom, under what approval authority, and with which bank routing parameters. However, these logs are only as trustworthy as the internal audit and change control processes governing them. In centralized architectures, all state transitionsincluding approval overridesoccur within a domain controlled by the organization, meaning tampering may be difficult to detect retrospectively without strong segregation-of-duties enforcement.

This reliance on internal state integrity is one of the key differentiators between traditional off-chain orchestration and ledger-based on-chain execution. Centralized logs require structured audit workflows to demonstrate compliance, whereas distributed ledger execution inherently embeds immutability and non-repudiation at the protocol level. Understanding where and how payments are orchestrated in the ERP-to-bank pipeline provides the baseline for evaluating the organizational benefits and tradeoffs of shifting authorization or state management into on-chain environments.

PAYMENT BATCH EXECUTION LOG (SIMULATED SAP/TREASURY OUTPUT)

Batch Run ID: PMT_2025_01_19_08_32_17
Company Code: 1000
Payment Method: T (Treasury Transfer)
Bank Route: PRIMARY_CORRESPONDENT_BANK
Execution Mode: RELEASE

DocNo	Vendor	Amount	Currency	Status	Timestamp
45000123	VEND-0091	125,000.00	USD	RELEASED	2025-01-19 08:32:21
45000124	VEND-0044	89,500.00	EUR	RELEASED	2025-01-19 08:32:24
45000125	VEND-1029	410,000.00	USD	RELEASED	2025-01-19 08:32:29
45000126	VEND-0007	9,800.00	INR	PENDING_ACK	2025-01-19 08:32:33
45000127	VEND-2021	250,700.00	GBP	PENDING_ACK	2025-01-19 08:32:35
45000128	VEND-3033	1,050.00	USD	FAILED	2025-01-19 08:32:38
...	...	...	...	...	...

SUMMARY:
  Total Documents: 250
  Successfully Released: 243
  Awaiting Bank Confirmation: 6
  Failed Transfers: 1
Execution Time: 00:00:14

Note: Re-run status reconciliation after bank acknowledgment window closes.

Figure 1. Payment Batch Execution Log

For clarity in this discussion, the execution trace shown in Figure 1 represents a realistic Payment Batch Execution Log. Instead of depicting logical flow, the figure presents an operational snapshot of a batch processing run, including timestamps, document numbers, release flags, execution statuses, and settlement trace references. This format reflects the actual diagnostic environment used by treasury operations, payment analysts, and audit reviewers when assessing batch-level execution consistency.

3. On-Chain Execution Model for Corporate Payments

In an on-chain execution model, corporate payment instructions and release conditions are encoded directly into smart contracts that operate as deterministic state-transition systems on a blockchain network. Rather than relying on ERP-internal workflow rules or middleware approval chains, the authorization, settlement triggers, and finality semantics are enforced through contract logic that executes uniformly across distributed network nodes. This approach eliminates reliance on internal system logs for validation because the state transitions themselves become part of a shared, cryptographically verifiable ledger that cannot be altered retrospectively once confirmed. The payment contract represents a self-contained execution policy, defining who may receive funds, how much may be transferred, and under what conditions transfers may occur.

During contract initialization, payment parameters are written to the blockchain's persistent storage layer. The most fundamental fields include the beneficiary address, representing the authorized payment recipient, and the payment amount, representing the fixed value that may be released. These fields are stored immutably unless explicitly controlled by time- or condition-based update logic. The immutability of these fields ensures that once the payment has been declared, no internal system actor can redirect funds to another recipient without producing an on-chain transaction that leaves an audit trace. In corporate environments where approval integrity and identity trust are essential, this feature provides strong protection against silent rerouting or unauthorized modification of release details.

Release logic is encoded as a conditional guard that must be satisfied before a payment is executed. Typical unlock conditions include time-based maturity (e.g., block height ≥ specified threshold), multi-signature approval, external oracle confirmation, or compliance with settlement sequencing rules. Because the smart contract evaluates the release condition during runtime, the payment can only occur if the condition returns a deterministic *true* state. This eliminates ambiguity associated with middleware-triggered execution workflows, where timing windows or network delays may cause inconsistent state evaluations. On-chain

guards therefore serve as both execution constraints and audit guarantees.

Once the payment condition is satisfied, the transaction is submitted and the final settlement is recorded as part of the blockchain consensus ledger. Unlike off-chain execution, where settlement acknowledgement may depend on proprietary banking networks, the settlement outcome in an on-chain model becomes globally observable and cryptographically verifiable. Finality is tied to the consensus depth (e.g., number of confirmed blocks), which determines when the transaction is effectively irreversible. Although this introduces latency compared to instantaneous internal ledger updates, the benefit is that settlement cannot be disputed or retrospectively altered without visible fork or rollback events, which are highly improbable in mature chain environments.

The integrity of on-chain payment states can be evaluated by comparing declared contract state to execution receipts. A settlement hash recorded in the blockchain receipt provides a cryptographic anchor that proves the payment was executed under the defined rules. During audit events, external parties can independently verify the correctness of payment execution by reconstructing contract state directly from blockchain records. This eliminates dependency on ERP-provided logs or middleware status files, significantly strengthening regulatory compliance readiness and third-party verification capabilities.

The primary state variables and their corresponding audit implications are summarized in Table 1, which highlights how on-chain storage and execution constraints enforce identity consistency, value conservation, logic correctness, and external proof of settlement. This structured state model transforms payment orchestration from an internally governed process into a cryptographically enforced, externally verifiable execution framework.

Table 1. Key On-Chain Payment State Fields and Integrity Controls

Field	Layer	Constraint	Audit Significance
Beneficiary	Contract Storage	Immutable post-creation	Ensures identity consistency
Amount	Storage	Conserved across state transitions	Prevents value leakage
Release Condition	Execution rule	Must hold true to unlock	Ensures logic correctness
Settlement Hash	Blockchain receipt	Must match ledger root	Enables external proof

4. Off-Chain Orchestration Execution Model

In an off-chain orchestration model, payment execution is managed through workflow engines, treasury management systems (TMS), or bank communication middleware that sits between the ERP and the settlement network. Instead of encoding rules directly on-chain, the orchestration logic resides in application workflows that enforce approval hierarchies, routing conditions, exception checks, and retry policies. The ERP produces a payment batch, and the middleware layer interprets the batch metadata to determine which payments can be immediately released, which require multi-level authorization, and which must await compliance validation or liquidity confirmation. Because execution logic is managed off-chain, the processing steps are fast and flexible, enabling rapid reconfiguration of approval routing without modifying underlying financial data structures.

This workflow engine typically includes a queueing mechanism where payment instructions are first placed into a *Queued* state. Once relevant authority checks are completedsuch as treasury approval, business unit sign-off, or automated policy validationthe payment transitions to an *Approved* state. From there, the orchestration system triggers the release process, packaging the payment into a compatible transmission format and pushing it to a bank or payment network. Successful receipt acknowledgment from the bank transitions the instruction to a *Settled* state, ensuring that ERP clearing accounts and cash ledger entries correctly reflect confirmed settlement.

A major advantage of this model is retry and exception handling. If a payment fails due to temporary network disruption, formatting mismatch, insufficient funds, or bank-side validation error, the workflow engine records the failure and either attempts automated recovery or routes the issue to human review. This is difficult to implement in on-chain execution environments where state immutability makes reversal and reprocessing expensive. Off-chain systems can seamlessly roll back payment status, regenerate messages, or modify routing

parameters without altering financial audit trails, preserving operational continuity.

Approval chain management is another key strength of off-chain orchestration. Corporate treasury operations often require multi-step authorization, sometimes involving legal, compliance, and financial controllers across different divisions. The middleware can enforce role-based access control, escalations, delegated authority exceptions, and time-windowed release permissions. These capabilities are essential in large organizations where financial governance must balance transaction velocity with strict internal controls. On-chain equivalents of these approval layers are achievable but significantly more complex and less adaptable to organizational restructuring.

However, off-chain orchestration relies heavily on log integrity and synchronization correctness. Because the workflow engine is responsible for state transitions, the trustworthiness of the payment record depends on accurate logging, secure storage of transaction history, and controlled access to administrative operations. If system logs are altered, overwritten, or incorrectly synchronized between ERP, middleware, and banking systems, discrepancies in reconciliation may arise. This places strong emphasis on audit policies, backup retention schedules, and change-control procedures.

To illustrate how operational visibility is provided in off-chain orchestration, Figure 2 displays a Treasury Workflow Execution Monitor as a status screen. The figure presents a table showing payments transitioning through *Queued → Approved → Released → Settled* states, including timestamps and execution remarks. This representation is characteristic of treasury analysts' daily operational dashboards, where they assess execution health, track errors, and coordinate exception resolution. The figure does not depict process flow; it provides a window into the actual runtime state of the payment lifecycle, reflecting the operational reality of off-chain payment execution environments.

```
TREASURY WORKFLOW EXECUTION MONITOR (SIMULATED OPERATIONS VIEW)
------------------------------------------------------------------------
Batch ID: TRSY_RUN_2025_01_19_11_14_05
Company Code: 1000
Execution Mode: AUTOMATED RELEASE
------------------------------------------------------------------------
Stage       DocNo     Vendor      Amount      Currency   Timestamp
------------------------------------------------------------------------
QUEUED      45000123  VEND-0091   125,000.00  USD        2025-01-19 11:14:10
APPROVED    45000124  VEND-0044   89,500.00   EUR        2025-01-19 11:14:13
APPROVED    45000125  VEND-1029   410,000.00  USD        2025-01-19 11:14:16
RELEASED    45000126  VEND-0007   9,800.00    INR        2025-01-19 11:14:20
RELEASED    45000127  VEND-2021   250,700.00  GBP        2025-01-19 11:14:22
SETTLED     45000128  VEND-3033   1,050.00    USD        2025-01-19 11:14:27
...         ...       ...         ...         ...        ...
------------------------------------------------------------------------
Summary:
   Total Items:       250
   Approved:          148
   Released:           89
   Settled:            12
Execution Monitor Refresh Interval: 5 seconds
------------------------------------------------------------------------
```

Figure 2. Treasury Workflow Execution Monitor

5. Reconciliation, Finality, and Audit Integrity Comparison

Reconciliation in corporate payment environments depends heavily on how finality is defined and enforced across the settlement pipeline. In on-chain execution, finality is achieved once a transaction has been included in a sufficiently confirmed block, meaning that the probability of chain reorganization becomes economically or cryptographically negligible. This creates a mathematically provable termination state, where the ledger cannot be altered without consensus-level disruption. In contrast, off-chain orchestration relies on acknowledgement-based completion, where banks or intermediaries communicate settlement success via message return codes. While faster, these acknowledgements depend on institutional trust and require internal system logs to reconstruct final payment confirmations. The comparison is summarized in Table 2, which contrasts reconciliation enforcement across both execution models.

Table 2. Reconciliation Behavior: On-Chain vs Off-Chain

Dimension	On-Chain	Off-Chain	Tradeoff
Finality Mechanism	Block confirmation	Acknowledgement status	On-chain slower but tamper-proof
Error Repair	Compensating TX patterns	Workflow retry & rollback	Off-chain more operational flexibility
Audit Evidence Strength	Ledger-root validation	Log-based replay	On-chain higher integrity
External Verifiability	Trustless validation	Requires system access	On-chain regulator-friendly

Error handling mechanisms also differ substantially between models. In on-chain payment flows, incorrect execution cannot be directly reversed; instead, compensating transactions are issued to offset unintended state transitions. This ensures immutability but introduces operational overhead and requires precise procedural design to prevent cascading accounting inconsistencies. Meanwhile, off-chain models allow explicit retry logic, rollback workflows, and approval-chain overrides at the orchestration layer. Such flexibility accelerates exception resolution but reduces the certainty of historical state integrity, since log correctness becomes dependent on procedural controls, timestamp sequencing, and system administrator guarantees rather than cryptographic commitment.

From an audit and compliance standpoint, the distinction becomes even clearer. On-chain ledgers inherently support external verification, allowing auditors, regulators, or counterparties to independently validate payment traces using publicly attestable Merkle roots. This makes audit trails self-authenticating, enabling non-repudiation without direct access to internal systems. In contrast, off-chain audit reconstruction requires controlled access to ERP logs, workflow system entries, and bank messaging archives, meaning the trust boundary is centralized rather than distributed. As shown in Table 2, this directly impacts the strength of forensic accounting outcomes, especially in regulated Treasury environments.

Performance dynamics and latency trade-offs are visualized in Figure 3, which presents a batch-size–indexed latency heatmap comparing on-chain and off-chain execution timings. Larger batch groups demonstrate considerably higher processing times on-chain due to consensus finalization and block propagation intervals. Off-chain workflows maintain significantly lower latency across all batch tiers, though at the cost of lower immutability guarantees. These execution characteristics indicate that hybrid architectures, where off-chain orchestration is used for operational speed and on-chain anchoring is applied at batch finalization boundaries, are often optimal for enterprise-scale corporate Treasury systems.

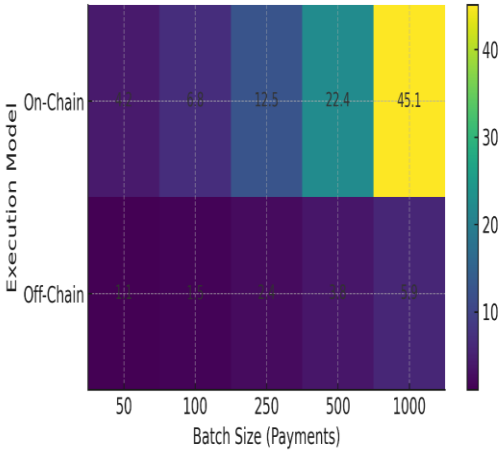

Figure 3. Latency Heatmap: On-Chain vs Off-Chain Execution

6. CONCLUSION

The comparative analysis of on-chain and off-chain execution models for corporate payment orchestration demonstrates that no single model is universally optimal, and deployment decisions must align with operational, regulatory, and performance objectives. On-chain execution provides immutability, cryptographic finality, and externally verifiable auditability, making it highly suited for environments where transparency, regulator visibility, and non-repudiation are primary concernssuch as intercompany settlements, high-stakes treasury transactions, or cross-entity escrow arrangements. However, the latency introduced by block confirmation cycles and compensating transaction patterns means that on-chain execution is less ideal for high-frequency or throughput-sensitive payment processing scenarios.

Conversely, off-chain orchestration offers speed, flexible exception handling, and integration compatibility with existing ERP and banking infrastructure. Its weakness lies in centralized trust assumptions and reliance on system logs for audit traceability. Therefore, the most practical deployment strategy for enterprise treasury operations is a hybrid model: perform operational execution and approval workflows off-chain, and anchor final settlement states on-chain at batch boundaries. This approach retains the performance benefits of off-chain execution while gaining the tamper-proof audit guarantees of on-chain anchoring, making it both operationally efficient and compliance-ready for modern corporate finance landscapes.

REFERENCES

1. Carter, Bry, Satya Moorthy, and Dave Walters. "Enterprise architecture view of complex system governance." *International Journal of System of Systems Engineering* 7.1-3 (2016): 95-108.
2. Sharma, Amit Kumar. "SWIFT adoption challenges."
3. Wood, Gavin. "Ethereum: A secure decentralised generalised transaction ledger." *Ethereum project yellow paper* 151.2014 (2014): 1-32.
4. Talebi, Javad, Ali Dehghantanha, and Ramlan Mahmoud. "Introducing and analysis of the Windows 8 event log for forensic purposes." *International Workshop on Computational Forensics*. Cham: Springer International Publishing, 2012.
5. Petitjean, Mikael. "Bank failures and regulation: a critical review." *Journal of Financial Regulation and Compliance* 21.1 (2013): 16-38.
6. Pilkington, Marc. "Blockchain technology: principles and applications." *Research handbook on digital transformations*. Edward Elgar Publishing, 2016. 225-253.
7. Clark, Jeremy, et al., eds. *Financial cryptography and data security*. Springer Berlin Heidelberg, 2016.
8. Conoscenti, Marco, Antonio Vetro, and Juan Carlos De Martin. "Blockchain for the Internet of Things: A systematic literature review." *2016 IEEE/ACS 13th International Conference of Computer Systems and Applications (AICCSA)*. IEEE, 2016.
9. Supervision, Banking. "Basel committee on banking supervision." *Principles for Sound Liquidity Risk Management and Supervision (September 2008)* (2011).