Blockchain reconciliation architecture

A technical overview of the Rock Innovation patent family

Rock Innovation LLC

US 10,192,073 · US 10,943,029 · US 11,720,711

Priority date: November 19, 2016

28 claims · 87 citing patent families

rockinnovationip.com · April 2026

Contents

SECTION 01

Executive summary

Rock Innovation LLC owns three U.S. utility patents covering blockchain reconciliation architecture. The patents share a single priority date of November 19, 2016. Together they contain 28 claims across system and method categories. Since issuance, 87 patent families from organizations including IBM, Visa, Microsoft, Bank of America, Samsung, and Toyota have cited them in their own filings or been cited against them during examination.

The patents describe a specific architecture: a system that connects to one or more blockchains, retrieves transaction records (called "interaction objects" in the patent language), pulls associated data from the same or different chains ("supplemental objects"), applies stored business logic ("preconfigured rules"), and produces reconciled output records ("compilation objects") that are written back to the blockchain. The patents call the processing component an "object compiler." This paper maps that architecture to modern engineering vocabulary.

Three patents cover the same architecture at different levels of specificity. US 10,192,073 (the parent, granted 2019) carries the most detailed claims, including flagging of missing records and handling of replacement and addendum data types. US 10,943,029 (first continuation, granted 2021) emphasizes criteria-based retrieval and covers the core compilation pipeline without requiring those specific data-handling behaviors. US 11,720,711 (second continuation, granted 2023) carries the broadest independent claims: seven elements that describe any system connecting to a blockchain, receiving supplemental data, applying rules, and producing compiled output stored on-chain.

The architecture these patents describe has structural parallels in every major category of blockchain system deployed after November 2016. Layer 2 rollups collect transactions, apply validation logic, and post compiled settlement data to Layer 1. Cross-chain bridges verify source-chain state against destination-chain records using oracle data. DeFi protocols reconcile user positions against price feeds and apply liquidation rules. Enterprise settlement platforms match multi-party records across distributed ledgers. Each of these system types follows a workflow that corresponds to elements in at least one of the three patents.

This paper provides the technical mapping between patent terminology and production blockchain architectures. It is written for engineers who need to assess whether their system's architecture falls within claim scope, for attorneys evaluating claim strength, and for licensing professionals who need a clear technical basis for commercial discussions.

SECTION 02

The reconciliation problem in distributed systems

Reconciliation, in its simplest computer science definition, is the process of ensuring that two or more sets of records agree. In traditional accounting software, this means comparing a bank statement against internal ledger entries. In distributed databases, it means resolving conflicts between replicas. The operation predates blockchains by decades.

Centralized systems make reconciliation straightforward because a single authority controls the data. When Bank A's internal ledger disagrees with Bank B's, one of them is wrong, and a central clearinghouse (SWIFT, ACH, DTCC) resolves the discrepancy by reference to its own authoritative record. The reconciliation logic lives inside the clearinghouse.

Distributed ledgers eliminated the clearinghouse. That was the point. Consensus protocols replaced central authority with mathematical proof: if 51% of nodes agree on a transaction's validity, it becomes part of the canonical ledger state. This solved the trust problem. It did not solve the reconciliation problem. It actually made it harder.

Why distributed ledgers create a new reconciliation gap

A blockchain guarantees that all nodes agree on what transactions occurred. It does not guarantee that those transactions are consistent with anything outside the chain. A purchase order recorded on Hyperledger Fabric says nothing about whether the corresponding invoice was generated in SAP. An Ethereum transaction confirming a token swap says nothing about whether the counterparty received settlement in their bank account. A sensor reading written to a supply chain ledger says nothing about whether the physical goods match the recorded specifications.

In centralized systems, these cross-system checks happen inside a single trust boundary. The ERP system that records the purchase order also generates the invoice. The bank that processes the payment also updates the account balance. One system, one owner, one reconciliation process.

In blockchain-based systems, each participant controls their own data. The ledger stores a shared record of the transaction, but the context surrounding that transaction, the business rules that determine whether the transaction was fulfilled correctly, the supplementary data from external sources, the obligations that the transaction was supposed to satisfy, all of that sits outside the shared ledger. Someone has to bring it together.

The state of the art in November 2016

When Rock Innovation's provisional application was filed on November 19, 2016, the blockchain industry was still in its prototype phase. Ethereum had launched 16 months earlier. Its total market capitalization was under $1 billion. Smart contracts existed but were limited to simple state transitions: ERC-20 token transfers, basic multisig wallets, rudimentary escrow logic.

Enterprise blockchain platforms barely existed. Hyperledger Fabric was in incubation at the Linux Foundation; its 1.0 release would not arrive until July 2017. R3's Corda was available only

as a developer preview. The Enterprise Ethereum Alliance would not form until February 2017. JPMorgan's Quorum was an internal project that had not yet been open-sourced.

No production system at that time provided a general-purpose framework for multi-source blockchain reconciliation. Individual companies built ad-hoc scripts. Consulting firms produced PowerPoint architectures. The public record does not show a granted patent or published system that treated reconciliation as a first-class operation with its own data model, rule engine, and compilation pipeline before this filing date.

The Rock Innovation patents addressed exactly this gap. The provisional filing captured an architectural pattern that the industry would not fully encounter for another two to three years, when enterprise blockchain deployments moved from pilot to production and confronted the multi-source data matching problems the patents address.

> **The filing date in legal context**
>
> The November 19, 2016 priority date establishes the date for prior art purposes: any prior art used to challenge these patents must predate that filing. Enforceable patent rights begin at issuance (2019 for the '073, 2021 for the '029, 2023 for the '711) under 35 U.S.C. § 154. The relevant technical question is whether a system's architecture matches the claim elements.

SECTION 03

Patent architecture walkthrough

This section walks through each major component of the patented system using the actual patent figures. For every component, we provide three things: the patent's own terminology, the engineering equivalent in modern blockchain vocabulary, and a concrete example of where this component appears in production systems today.

System architecture (Figure 7)

Figure 7 shows the full reconciliation system. The left side contains the processing engines: a compiler (701), an interaction recorder (702), a rule engine (703), an encrypt/decrypt engine (704), a block processor (705), and a report engine (716). Above them are the data stores: configuration database (706), rules database (707), object database (708). Two interfaces connect outward: a subscriber interface (709) and a device interface (710). The blockchain (500) sits on the other side of a network (310), and four device types connect from the application layer: company devices (712), user devices (713), subscriber devices (714), and entity devices (715).

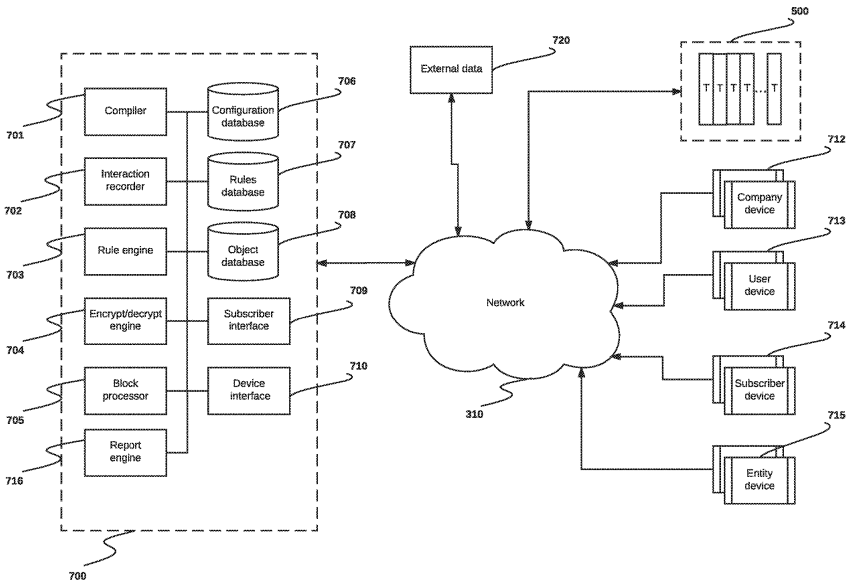

Fig. 7

Figure 7 from US 10,192,073: Block reconciliation system. Processing engines (701-716) sit between the blockchain (500) and connected devices (712-715). The compiler (701) is the central processing component.

An engineer looking at this diagram will recognize a standard middleware pattern: a service layer between a data source (the blockchain) and its consumers (connected devices). What distinguishes this from generic middleware is the specific data flow. The compiler does not proxy blockchain data to applications. It reads interaction objects from the chain, retrieves supplemental objects based on criteria, applies rules stored on the same chain, produces a new data structure (the compilation object), and writes that structure back to the chain. Read, match, apply rules, write: that four-step pipeline is the claimed invention.

Object model (Figure 6)

Figure 6 defines the ten data types in the system. This diagram is the Rosetta Stone for mapping patent terminology to production architectures.

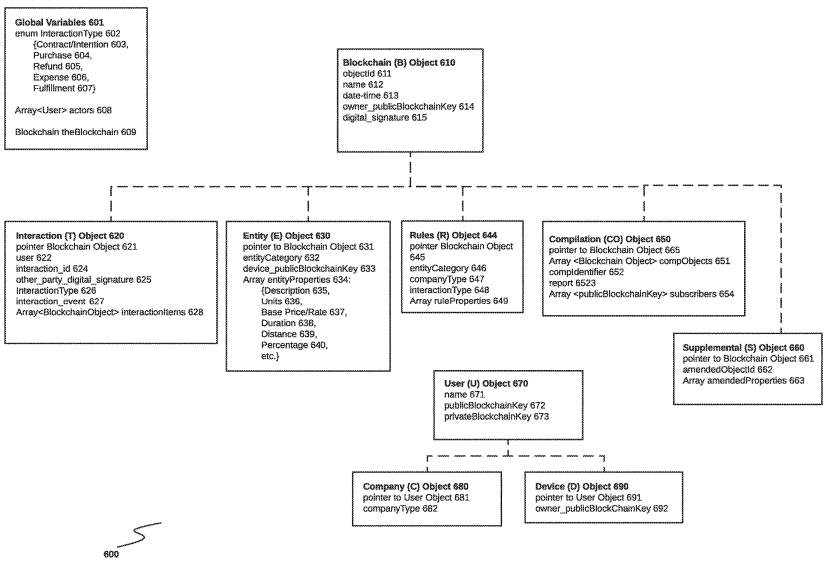

Fig. 6

Figure 6 from US 10,192,073: Ten object types. Global variables (601) define interaction types. Interaction objects (620) are the primary input. Rules objects (644) govern processing. Compilation objects (650) are the reconciled output. Supplemental objects (660) carry amendments.

Terminology translation table

The following table maps each patent term to its modern engineering equivalents. An engineer reading this table should be able to locate the corresponding component in their own system.

Patent term	Ref	Engineering equivalents (any of these indicate a mapping)
Interaction object	620	Transaction record, L2 transaction batch, swap event, trade confirmation, sensor data payload, attestation, on-chain commitment, user action log. Any discrete unit of data written to a blockchain by a participant.
Supplemental object	660	Oracle price feed, cross-chain state proof, off-chain attestation, counterparty confirmation, amendment record, L1 calldata posted by a sequencer, insurance endorsement, external API response cached on-chain, Chainlink aggregator report.
Preconfigured rules	644	Smart contract validation logic, liquidation threshold parameters, endorsement policy (Hyperledger), fraud proof window rules, collateralization ratio, matching criteria, state transition function, governance parameters stored on-chain.
Compilation object	650	State root, settlement proof, reconciliation report, compiled batch (rollup), liquidation receipt, validity proof, finalized block commitment, compiled financial statement, reconciled position record. Any structured output that combines input records with supplemental data according to rules.
Object compiler	701	Sequencer (Arbitrum, Optimism), prover (zkSync, StarkNet), aggregator (Chainlink), liquidation bot (Aave, Compound), endorsing peer (Hyperledger Fabric), bridge relayer (Wormhole, LayerZero), batch processor, state compiler.
Connected device	712-715	Web3 wallet, API client, dApp frontend, IoT sensor, enterprise integration endpoint, mobile app, validator node, relayer service. Any system that sends criteria to or receives compilation results from the reconciliation system.
Blockchain connection	500	RPC endpoint, WebSocket subscription, chain indexer (The Graph), light client, full node connection. The system's read/write interface to one or more blockchains.
Rules database	707	On-chain governance registry, parameter store, protocol configuration contract, endorsement policy configuration, risk parameter repository.
Report engine	716	Block explorer, analytics dashboard, compliance reporting module, audit trail generator, subgraph query layer.
Subscriber interface	709	Event subscription, webhook, push notification service, WebSocket feed, GraphQL subscription, data consumer API.
Flagging ('073 only)	n/a	Exception alert, incomplete transaction marker, fraud proof challenge, discrepancy flag, missing-data notification, liquidation warning, unmatched record indicator.
Replacement supplemental ('073 only)	n/a	State override, correction entry, amended record, superseding transaction, rollback-and-replace, updated oracle price that overwrites a stale feed.
Addendum supplemental ('073 only)	n/a	Appended metadata, additional attestation, enrichment data, late-arriving counterparty confirmation, supplementary documentation attached to an existing record.

Compilation pipeline (Figure 11)

Figure 11 shows the step-by-step compilation flow. Each step corresponds to a specific claim element.

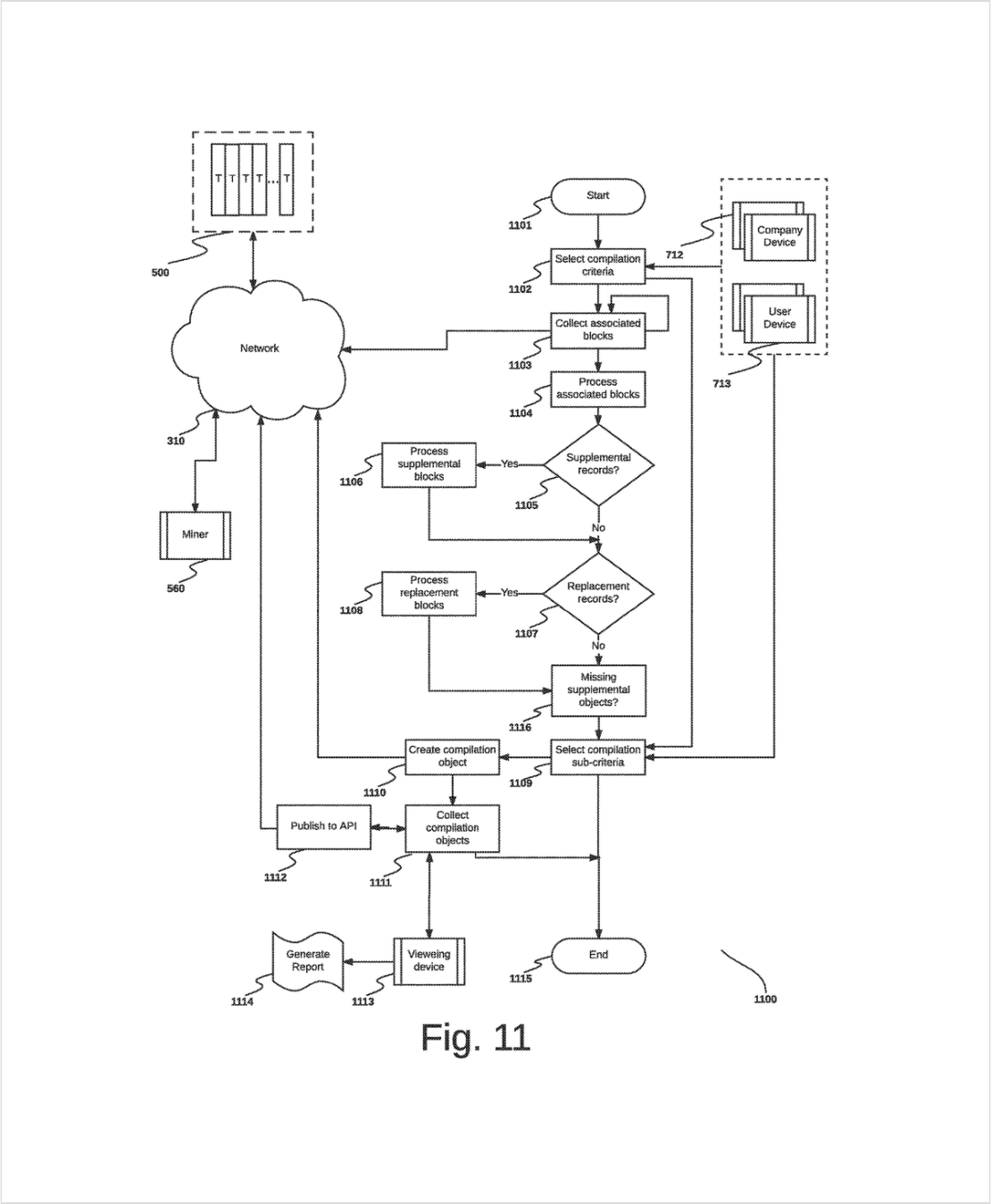

Figure 11 from US 10,192,073: The compilation pipeline from criteria selection (1101) through block collection, supplemental processing, rule application, compilation object creation (1110), and output via API (1112) and report engine (1114).

Pipeline step	Patent element	What happens (engineering terms)
1. Criteria selection	Receive criteria from connected device	A client sends query parameters: date range, transaction type, counterparty filter, chain ID. This is the equivalent of a GraphQL query, an API call with filter parameters, or a sequencer accepting transactions for the next batch.
2. Block collection	Receive blocks based on criteria	The compiler retrieves blockchain records matching the criteria. Each block contains preconfigured interaction objects. In rollup terms: the sequencer pulls pending transactions from the mempool. In DeFi terms: the liquidation bot queries positions matching its liquidation criteria.
3. Supplemental check	Analyze for associated supplemental objects	For each interaction object, the system checks whether required supplemental data exists. A bridge checks whether the source-chain lock transaction has a corresponding relay proof. A lending protocol checks whether the price oracle has a current feed for the collateral asset.
4. Supplemental retrieval	Request supplemental blocks	The compiler fetches supplemental data from the blockchain(s). This could be cross-chain: the supplemental object may reside on a different chain than the interaction object. Oracle aggregators pull data from multiple external sources and relay it on-chain.
5. Missing data handling	Flag if not found ('073)	If required supplemental data is absent, the system flags the interaction object. An optimistic rollup's fraud proof challenge period is this step: if no one submits a contradicting proof (the supplemental object), the batch is flagged as valid by default. If someone does submit, the flagged transaction gets reprocessed.
6. Data merging	Replace or append ('073)	Supplemental objects either overwrite (replacement) or extend (addendum) the original interaction data. A corrected oracle price overwrites a stale one. An additional counterparty signature appends to the existing approval chain.
7. Rule application	Receive preconfigured rules	The compiler retrieves business logic from the blockchain. In production: a smart contract's state transition function, a lending protocol's collateralization parameters, a bridge's verification threshold, a rollup's validity function.
8. Compilation	Create compilation objects	The compiler produces a structured output record containing the reconciled interaction data, supplemental data, and rule-application results. In rollup terms: the batch commitment or validity proof. In DeFi: the liquidation receipt. In enterprise: the reconciled settlement record.
9. Storage	Store on blockchain	The compilation object is written back to the chain. An L2 rollup posts its state root to L1. A bridge writes the relay proof to the destination chain. A lending protocol records the liquidation on-chain.

How to read this table

If your system performs all nine steps, it maps to the '073 patent's independent claims. If it performs steps 1-4 and 7-9 (no flagging, no replace/append distinction), it maps to the '029 patent. If it performs steps 4, 7, 8, and 9 (supplemental retrieval, rule application, compilation, and on-chain storage), it maps to the '711 patent's independent claims. The '711 patent has the lowest bar.

SECTION 04

Claim family comparison

The three patents were filed as a continuation chain. They share the same specification, the same figures, and the same priority date. Their claims differ by design. Each successive patent adjusted the claim scope to cover different slices of the same underlying architecture. Understanding this layered structure matters for both infringement analysis and design-around evaluation.

Independent claim elements side by side

Claim element (plain English)	'073 (parent)	'029 (1st cont.)	'711 (2nd cont.)
Network-connected computer with memory, processor, stored instructions	Required	Required	Required
Connections to a plurality of devices	Required	Required	Dep. claim 11
Connection to one or more blockchains	Required	Required	Required
Object compiler	Required	Implied (processor)	Implied (processor)
Receive criteria from a connected device	Required	Required	Dep. claim 12
Retrieve blocks based on criteria, each corresponding to preconfigured interaction object	Required	Required	Not in indep.
Analyze interaction objects for associated required supplemental object	Required	Required	Dep. claim 12
Request supplemental blocks from blockchain(s)	Required	Required	Required
If supplemental not found: flag the interaction object	**Required**	Not required	Not required
If supplemental type is replacement: replace properties	**Required**	Not required	Not required
If supplemental type is addendum: add properties	**Required**	Not required	Not required
Receive preconfigured rules from blockchain	Required	Required	Required
Create compilation objects based on criteria and rules	Required	Required	Required
Store compilation objects on blockchain	Required	Required	Required

"Dep. claim" = element appears in dependent claims only; not required for independent claim infringement.

Why the claims are structured this way

The '073 patent (13 elements) was prosecuted first and carries the most detailed claims. The flagging, replacement, and addendum elements make it highly specific. This specificity cuts both ways: the claims are harder to invalidate because of their precision, but they require all three

supplemental-handling behaviors to be present for infringement. This patent targets systems that perform full-lifecycle reconciliation with error handling and data amendment flows.

The '029 patent (10 elements) drops the flagging and replacement/addendum requirements. It retains the criteria-based retrieval and supplemental analysis steps, covering the core compilation pipeline without requiring specific error-handling or data-amendment behaviors. A system that retrieves blockchain records by criteria, checks for supplemental data, applies rules, and produces compiled output reads on '029 even if it never flags a missing record or distinguishes between replacement and addendum types.

The '711 patent (7 elements) strips the claims to their structural minimum. No criteria receipt in the independent claim. No multi-device requirement. No analysis step. The independent claim requires only: connect to blockchain, receive supplemental blocks corresponding to interaction objects, receive preconfigured rules, create compilation objects, store them on-chain. This is the broadest formulation and the hardest to design around.

Design-around analysis

For a system to fall outside all three patents simultaneously, it would need to lack every one of the following:

> **Minimum elements for complete avoidance (all must be absent)**
>
> 1. No retrieval of supplemental data associated with transaction records from a blockchain
> 2. No application of stored rules (on-chain or retrieved from on-chain) to the retrieved data
> 3. No creation of a compiled/reconciled output record from the combination of transaction data, supplemental data, and rules
> 4. No storage of that compiled output on a blockchain

A system that performs transaction processing but never retrieves supplemental data would fall outside the claims. A system that retrieves supplemental data but applies no stored rules would fall outside. A system that applies rules but never writes the compiled output back to a chain would fall outside.

In practice, the fourth requirement is the most limiting for design-around purposes. Most blockchain systems that perform reconciliation write their results on-chain. That is the point of using a blockchain: the reconciled state becomes part of the immutable record. A system that reconciles off-chain and never posts results to any blockchain could potentially avoid all three patents, but such a system would also forfeit the core value proposition of blockchain-based reconciliation.

Avoiding only one patent while reading on the other two has limited practical value. The family is licensed as a unit.

SECTION 05

Industry mapping and forward citations

The reconciliation architecture described in these patents has structural parallels in every major blockchain market segment. The following analysis identifies where patent claim elements correspond to published technical operations. Whether any specific implementation actually practices a given claim requires element-by-element claim construction, which is a legal determination beyond the scope of this paper.

Layer 2 rollups (Arbitrum, Optimism, Base, zkSync, StarkNet, Scroll)

A rollup sequencer collects L2 transactions (interaction objects), retrieves L1 state data needed for validation (supplemental objects), applies the rollup's state transition function (preconfigured rules), produces a batch commitment or validity proof (compilation object), and posts it to L1 (stores on blockchain). Each of those operations corresponds to an element of the '711 independent claim. Optimistic rollups add a fraud proof mechanism where missing or incorrect supplemental data triggers a challenge period, which parallels the '073 patent's flagging element.

Closest patent overlap: '711 independent claims for rollup architectures; '073 for optimistic rollups with fraud proofs.

Cross-chain bridges (Wormhole, LayerZero, Axelar, Chainlink CCIP)

A bridge verifies a transaction on the source chain (interaction object) by obtaining attestations or relay proofs from guardians or oracles (supplemental objects), applies verification rules (preconfigured rules), and produces a mint or release transaction on the destination chain (compilation object stored on blockchain). Bridges that flag unverified or disputed transfers parallel the '073 flagging element. Bridges that allow amended attestations (additional guardian signatures after initial submission) parallel the '073 addendum element.

Closest patent overlap: '029 for bridges with criteria-based retrieval; '711 for general bridge architectures.

DeFi lending and liquidation (Aave, Compound, MakerDAO)

A liquidation engine reads borrower positions (interaction objects), retrieves oracle price feeds (supplemental objects), applies collateralization and liquidation parameters (preconfigured rules), and produces liquidation transactions (compilation objects) recorded on-chain. The criteria-based retrieval element has a parallel here: the liquidation bot queries positions that meet specific health-factor thresholds. Oracle price updates that overwrite stale prices correspond to the '073 replacement supplemental element.

Closest patent overlap: '029 for criteria-based liquidation flows; '073 for systems with price-feed replacement and position flagging.

Enterprise blockchain settlement (Hyperledger Fabric, Corda, enterprise Ethereum)

Trade finance platforms match purchase orders against invoices and shipping confirmations across multiple participants' ledger entries. Hyperledger Fabric's endorsement pipeline has a close structural parallel: endorsing peers (the compiler) collect transaction proposals (interaction objects), obtain endorsements from other organizations (supplemental objects), apply the chaincode endorsement policy (preconfigured rules), and produce endorsed transaction blocks (compilation objects) committed to the shared ledger. Missing endorsements trigger rejection, paralleling the '073 flagging behavior.

Closest patent overlap: '073 for systems with flagging and amendment handling; '029 for the core endorsement/compilation pipeline.

Oracle networks (Chainlink, Pyth, API3)

An oracle aggregator collects individual data reports from multiple node operators (interaction objects), retrieves configuration data and reputation scores (supplemental objects), applies the aggregation algorithm and deviation thresholds (preconfigured rules), and produces a signed aggregated value (compilation object) posted on-chain. Chainlink's Off-Chain Reporting (OCR) protocol follows a similar pipeline structure. Stale or unresponsive nodes trigger threshold alerts, paralleling the '073 flagging element.

Closest patent overlap: '711 for oracle aggregation architectures; '073 for systems with node-failure flagging.

Forward citations: what they mean

Google Patents lists 87 patent families that cite the Rock Innovation patents. A citing family means either a patent examiner or a later applicant identified these patents as relevant prior art during prosecution. The citing organizations include IBM (multiple filings on distributed ledger management), Visa and Mastercard (payment settlement and cross-institution reconciliation), Bank of America (transaction verification), Microsoft (distributed data processing), Samsung and Toyota (device-interaction and IoT-blockchain integration), Oracle, Accenture, and Honeywell.

A forward citation is not evidence of infringement. It indicates that the citing party or a USPTO examiner considered the Rock Innovation patents technically relevant during prosecution. In licensing discussions, forward citations from major technology and financial services firms suggest that the patented architecture covers ground other companies are working in. When a company's own patent counsel cites these patents during their prosecution, it reflects a determination that the technology is relevant to their filing.

Segment	Citing companies (examples)	Relevant patent
Financial services / payments	Bank of America, Visa, Mastercard	'029, '073
Enterprise technology / cloud	IBM, Microsoft, Oracle, Accenture	'029, '711
Hardware / IoT / automotive	Samsung, Toyota, Honeywell	'073 (device interaction)

SECTION 06

Technical glossary

Addendum (supplemental type). A supplemental object that appends new properties to the associated interaction object without overwriting existing data. Claimed in '073 only. Engineering equivalents: additional attestation, late-arriving counterparty confirmation, appended metadata, enrichment record.

Block reconciliation. The process claimed in all three patents: collecting blockchain records, matching them against associated data, applying stored rules, and producing a reconciled output. Engineering equivalents: state compilation, batch processing, settlement verification, transaction matching.

Blockchain object (610). The chain reference within the data model. Properties include object ID, name, owner public key, and digital signature. Allows the system to address and operate across multiple blockchains simultaneously.

Compilation object (650). The structured output of the reconciliation process. Contains reconciled interaction data, associated supplemental data, and rule-application results. Properties include a pointer to the blockchain, a compilation identifier, a report field, and a subscriber array controlling access. Engineering equivalents: state root, validity proof, settlement record, batch commitment, reconciliation report, liquidation receipt.

Compiler / object compiler (701). The processing engine that performs reconciliation. Receives criteria, retrieves blocks, analyzes interaction objects, requests supplemental data, applies rules, creates compilation objects. Claimed in all three patents. Engineering equivalents: sequencer, prover, aggregator, endorsing peer, bridge relayer, liquidation engine, batch processor.

Connected device (712-715). Any network-connected system that interacts with the reconciliation computer. Four types defined: company device (712), user device (713), subscriber device (714), entity device (715). Engineering equivalents: wallet, API client, dApp frontend, IoT sensor, validator, integration endpoint.

Entity object (630). Business entity metadata. Properties include entity category, company type, device association, and an array of entity-specific properties (units, base price/rate, duration, distance, percentage). Engineering equivalents: organization profile, counterparty record, merchant configuration, participant identity.

Flagging. Specific behavior claimed in '073: when a required supplemental object is not found, the compiler marks the interaction object rather than silently omitting it. Engineering equivalents: exception alert, fraud proof challenge, discrepancy flag, unmatched-record marker, incomplete-transaction indicator.

Interaction object (620). The primary input data unit. Represents a transaction or event recorded on the blockchain. Properties include interaction ID, user reference, digital signature, counterparty signature, interaction type, interaction event, and an array of interaction items. Claimed in all three patents. Engineering equivalents: transaction record, swap event, user action, sensor reading, trade confirmation, on-chain commitment.

InteractionType (602). Enumerated classification of interactions. Examples in the specification: contract/intention (603), purchase (604), refund (605), expense (606), fulfillment (607). Engineering equivalents: transaction type code, event category, message type identifier.

Preconfigured rules (644). Business logic retrieved from the blockchain and applied during compilation. Stored as rule objects in the rules database (707), associated with specific interaction types and entity categories. Claimed in all three patents. Engineering equivalents: smart contract logic, validation rules, endorsement policy, liquidation thresholds, state transition function, governance parameters.

Replacement (supplemental type). A supplemental object that overwrites properties on the associated interaction object. Claimed in '073 only. Engineering equivalents: correction entry, amended record, superseding transaction, updated oracle feed, state override.

Subscriber (654). An entity authorized to receive compilation objects or reports. The compilation object includes a subscriber array defining which blockchain keys have access to the compiled data. Engineering equivalents: access control list, authorized viewer, data consumer, event subscriber.

Supplemental object (660). Data associated with an interaction object that is required for reconciliation but may reside on a different blockchain or in a different block. The compiler retrieves supplemental objects based on the association criteria. Claimed in all three patents. Engineering equivalents: oracle data, cross-chain state proof, counterparty record, off-chain attestation, relay proof, external data feed.