Reader’s note

This technical paper is unchanged from its original October 2025 release. It defines the formal authorization and verification structures that later enabled the articulation of $\kappa_{(C)}$ (the consent constant) in the scientific canon. No architectural or functional elements have been modified.

— Recur Labs, v1.1

The Recur Technical Paper

Defining the Permissioned-Pull Standard for Digital Value

Author: Mats Heming Julner — Recur Labs

Version 1.1 | November 2025

Preface

The Recur Whitepaper defined the conceptual foundation of consented continuity; a model where digital value moves continuously within cryptographic consent instead of reacting after failure.

This paper translates that principle into architecture. It defines the permissioned-pull standard as a technical primitive: a universal mechanism for consented flow that allows value to rebalance before imbalance, safely and without intermediaries.

While the whitepaper described the “why,” this document specifies the “how.” It outlines the authorization model, data schemas, revocation logic, and implementation parameters required to embed continuity at the protocol level.

The goal is not to introduce a new network, but to complete the existing ones; providing the missing layer of continuity beneath every digital dollar.

This paper serves as the canonical technical reference for the Recur permissioned-pull standard. It accompanies the reference implementation at github.com/recurmj/recur-standard.

Permanent record: IPFS CID bafkreihq2fkzdiqekyznerv73xenjpmsik2cb6fhczsoxlieyifqadnwbe — Recur Labs (October 2025).

Abstract

Finance still moves like a reflex: value reacts only after stress. Every stablecoin, every transfer, every liquidation cascade follows the same logic; push after imbalance.

The Recur architecture defines a missing motion: consented continuity. It introduces a permissioned-pull layer that allows digital money to move safely, predictably, and continuously within cryptographic consent.

This paper formalizes how continuity becomes computable: the technical design that lets liquidity rebalance before failure rather than after it. It describes the data structures, authorization logic, and verification flows required to embed consent, revocation, and timing directly into value transfer itself.

The architecture is defined across a sequence of interoperable standards: RIP-001 (Permissioned Pull Objects), RIP-002 (Consent Registry), RIP-003 (Cross-Network Flow Intent), RIP-004 (Non-Custodial Rebalancing), and their extensions through RIP-008 (Adaptive Routing and Settlement Mesh). Together, these define the flow layer for digital value.

1. Introduction

Modern financial rails are instantaneous but discontinuous. A push-based transaction is an isolated event: a sender signs, value moves, state changes, and the system waits for the next trigger. Between these moments, capital is static. Risk accumulates silently until imbalance forces reaction.

This architecture makes stability reactive by design. Each loan, margin position, or treasury balance remains vulnerable until someone does something. When volatility accelerates, these delayed actions compound into the familiar rhythm of crisis; over-leverage, liquidation, contagion.

The opposite of reaction is not inaction; it is continuity: motion that maintains equilibrium instead of restoring it. Recur defines the minimal primitives required to express that continuity in code.

Its central idea is simple:
A payment system should move value automatically within explicit consent, rather than manually after loss of balance.

To achieve this, Recur introduces three new components to the digital-value stack:

1. Permissioned Pull Objects (PPOs): signed, revocable grants that specify who may draw value, under what limits, and for how long.
2. Flow Channels: lightweight contracts that execute PPOs continuously or conditionally based on predefined triggers.
3. Consent Registry: an on-chain index that tracks, verifies, and enforces revocation across applications and networks.

Together, these elements form a flow layer: a universal mechanism for continuous, consent-driven liquidity movement across stablecoin networks and settlement systems.

Higher layers, defined in RIP-003 and RIP-004, extend this model across networks, allowing liquidity to stabilize itself across domains without custodial bridges.

The remainder of this paper details the engineering model, security assumptions, and economic implications of that architecture.

2. Design Principles

The Recur architecture rests on five foundational principles:

2.1 Continuity

Liquidity should not wait for imbalance. A financial system built on push events is always catching up to its own failures. Recur treats motion as a continuous state machine: capital can flow between authorized states before imbalance propagates. This continuity transforms stability from a reactive function into a native property of the network.

2.2 Consent

Every movement of value must occur within clear, cryptographic consent. Unlike custodial automation, permissioned pull does not transfer control; it delegates it temporarily, within explicit scope. A user signs an allowance not to a party, but to a structure. One that can be revoked, expired, or bounded at any time. Consent defines safety; revocation defines freedom.

2.3 Safety

Continuity cannot come at the cost of exposure. Recur embeds multi-layered safeguards:

- Each pull request must validate against both token-level allowance and permission object constraints.
- Expiry timestamps, balance limits, and cryptographic nonces prevent replay or drift.
- Revocation is absolute and global: one action cancels all active flows tied to a keypair.

2.4 Composability

Recur is not a chain; it is a pattern. It must operate across existing stablecoin systems, wallet standards, and Layer-2 architectures. Its design minimizes assumptions about where the value resides. Recur is designed so that a permissioned pull can behave identically across EVM networks (Ethereum, Base, etc.), and can in future be generalized to other execution environments. The logic remains invariant.

2.5 Transparency

Continuity must remain observable. The standard includes an on-chain registry model for visibility into active permissions and revocations. This registry (described in RIP-002) is intended to give wallets, auditors, and integrators a shared source of truth. This ensures regulators, auditors, and users alike can monitor liquidity movement without intermediaries or opaque batching.

These principles define not only the system’s technical posture, but its philosophical stance: stability by consent, automation by safety, trust by verification.

3. System Architecture

At its core, Recur consists of three modular layers:

1. **Authorization Layer:** where consent is granted and expressed.
2. **Flow Layer:** where movement is executed continuously under constraints.
3. **Verification Layer:** where validation and revocation are enforced.

3.1 Authorization Layer

The Authorization Layer encodes user intent as a Permissioned Pull Object (PPO).

Each PPO is a signed data structure containing:

Field	Description
grantor	Address granting consent to pull
grantee	Address authorized to initiate pulls
token	ERC-20 asset address
receiver	Destination address for funds pulled
maxAmount	Maximum total amount authorized under this PPO
validAfter	Earliest timestamp the PPO can be used
validBefore	Latest timestamp the PPO can be used
nonce	Unique salt to prevent replay
signature	EIP-712-compliant signature from grantor

PPOs can be created off-chain and registered only upon first use, minimizing gas cost and storage. They are revocable at any time by submitting a single revokePull(address grantee) transaction to the Consent Registry via a registry contract (RIP-002).

3.2 Flow Layer

In early implementations, execution is initiated per-call by the authorized grantee using the signed PPO. Future implementations introduce Flow Channels (RIP-005): dedicated contracts that continuously enforce timing, thresholds, or rebalancing logic under the same consent envelope.

In practice, a Flow Channel (RIP-005) is a contract that continuously executes a grantor's Permissioned Pull Object under rate limits, caps, and timing constraints. Channels can be paused, rate-adjusted, or revoked by the grantor at any time, and each attempted pull revalidates consent before executing.

Triggers may include:

- Scheduled intervals (e.g., streaming payroll).
- Balance deviation thresholds (e.g., automated rebalancing).
- Oracle-based conditions (e.g., margin collateralization).

The channel validates each execution against PPO parameters, ensuring no transfer exceeds its authorized scope. If any constraint fails — expired, revoked, or limit exceeded — the transaction halts automatically.

This architecture allows for both continuous and conditional flows. In continuous mode, a stream of micro-pulls maintains equilibrium in near real time. In conditional mode, a single pull executes only when a defined condition is met.

3.3 Verification Layer

The Verification Layer underpins both authorization and flow.

It performs three critical functions:

1. **Signature Verification:** validates EIP-712 signatures on PPOs to ensure origin authenticity.
2. **Registry Validation:** checks active consent state against the global Consent Registry.
3. **Revocation Propagation:** enforces real-time halting of flows once a revocation event is detected.

This guarantees that even if an off-chain actor or application continues to request pulls, revoked permissions will fail on-chain verification instantly.

3.4 Interoperability Architecture

Recur is designed to operate as an open standard. Its logic can be implemented via smart contracts, wallet SDKs, or embedded API calls. Cross-network functionality relies on minimal messaging: a signed PPO can conceptually be mirrored via proof relays to other networks without duplicating value.

The system therefore behaves as a continuity protocol, not a custody system; liquidity remains under user control at all times. Cross-network continuity is enforced through signed Flow Intents (RIP-003) and non-custodial executors (RIP-004), not through pooled liquidity or wrapped assets.

4. Security and Revocation Model

Continuity without containment would be chaos. Recur's permissioned-pull logic is engineered so that every automated motion remains bounded by consent and cryptography.

4.1 Non-Custodial Security

No Recur component ever holds funds. All balances stay within the native token contract; Flow Channels simply execute token transfers under the grantor's pre-signed scope. This preserves full self-custody. There is no private-key delegation, no pooled wallet, and no external escrow.

4.2 Revocation Mechanics

Revocation is immediate and absolute. A user can cancel a single PPO or all active permissions through one transaction.

Once a revocation event is written to the Consent Registry:

- All pending pulls referencing that PPO revert automatically.
- Flow Channels verify registry state at each execution, halting any transfer tied to a revoked ID.
- Applications subscribing to the registry receive an event emission to update local state instantly.

Revocation proofs are stored as short hashes to preserve efficiency and privacy; only the affected key-pairs and contract addresses are visible on-chain. All executors, including Flow Channels (RIP-005) and Cross-Network Rebalancers (RIP-004), must consult this registry before execution.

4.3 Limits and Failsafes

- Amount limits: Each PPO defines both per-flow and cumulative ceilings.
- Temporal limits: Expiry timestamps bound authorization duration.
- Nonce and replay protection: Prevents reuse of signed data.
- Circuit breaker: Global emergency pause callable only by the grantor's key to stop all flows in extreme conditions.

These guardrails ensure that even under malicious or buggy conditions, loss cannot exceed pre-approved scope.

4.4 Verification Path

At every execution step, verification follows the same deterministic order:

```
verifySignature(PPO)
→ checkRegistry(PPO.id == active)
→ enforceLimits(amount, interval, expiry)
→ executeTransfer()
```

If any condition fails, the transaction reverts with zero partial movement. Security therefore emerges from composition, not trust: no single component can bypass the chain of checks.

4.5 Auditable Continuity

In production implementations, every authorization, pull, and revocation MUST emit structured on-chain events so that wallets, auditors, and integrators can reconstruct a provable history of consent and motion, enabling transparent compliance and proof-of-consent frameworks for institutional users.

5. Example Scenarios

To illustrate how Recur behaves in practice, the following scenarios demonstrate its use across different financial contexts.

5.1 Stablecoin Auto-Rebalancing

A treasury holds USDC across multiple exchanges and custodial wallets. Using Recur, it issues a PPO to an automated grantee authorizing up to $5 million total per 24 hours. The Flow Channel monitors balances and automatically shifts excess liquidity toward deficit accounts when deviation >5%. If markets move violently, rebalancing occurs within consent before liquidation pressure accumulates. The treasury can revoke the PPO instantly, freezing all flows.

5.2 Recurring Payments

A consumer authorizes a media platform to pull $12 each month. Instead of storing card data or relying on third-party custody, the platform holds a signed PPO. At renewal, the Flow Channel executes a single pull under that scope. If the user cancels, the next pull fails verification; no disputes, no chargebacks, no waiting periods.

5.3 Protocol Insurance / Margin Top-Ups

A lending protocol receives consent to pull small increments of collateral from users' stablecoin wallets when health ratio < 1.2. As markets fluctuate, the Flow Channel automatically tops up margin positions pre-emptively, avoiding cascade liquidations. No custody, no global pauses; only local equilibrium maintained through consented flow.

5.4 Cross-Network Clearing

While Recur's initial implementations operate within single EVM networks, the same permissioned-pull logic can extend across them.

In its cross-network form, Recur does not bridge or custody assets. Instead, the liquidity owner signs a Flow Intent (RIP-003) declaring: which domain is overfunded, which domain is underfunded, which asset is eligible to move, which executor may act, and the maximum amount and time window. The Cross-Network Rebalancer (RIP-004) enforces that intent. It verifies the signature, checks revocation and caps, and then triggers a permissioned pull directly from the source domain into the destination address, using adapters that respect the grantor's consent.

No wrapped asset is ever minted. No intermediate pool is ever created. The executor never takes custody. All movement is still governed by the grantor's signed authorization.

5.5 Subscription Recovery and User Control

If a user forgets to cancel a service, they can still revoke its PPO after renewal. Future pulls fail automatically: no customer-support loops, no arbitration. This replaces trust in platforms with programmable control in the user's hands.

6. Implementation Architecture

Recur is designed to integrate gradually. No chain migration, no token swap, no new network required. Its architecture can be implemented through lightweight smart contracts, SDKs, and APIs layered atop existing systems.

6.1 Functional Overview

1. **Wallet Layer:**
   Wallets SHOULD expose a simple interface for users to issue, view, and revoke PPOs.
   Example methods:

```
authorizePull(address grantee, address token, uint256 limit, uint256 interval, uint256
expiry)
revokePull(address grantee)
```

Wallets render active PPOs as visual consent cards, similar to OAuth scopes.

2. **Application Layer:**
   Merchants, protocols, or agents integrate an SDK that verifies PPO state and triggers pull requests. Each pull must reference a valid PPO ID and proof of signature.
3. **Protocol Layer:**
   Smart contracts implement the core permissioned-pull logic; including the current pull module and the forthcoming Flow Channel and Consent Registry components. These contracts are designed to remain minimal, auditable, and upgradeable through open governance proposals rather than centralized control.
4. **Verification Layer:**
   Off-chain verifiers and block explorers read event logs and maintain consent snapshots for faster queries and regulatory audit trails.

6.2 Cross-Network Flow

The same PPO signature can operate across multiple EVM networks via message proofs. A relayer submits proof (per RIP-003) verifying that a PPO exists and remains active on its origin chain. This enables verified multi-chain continuity: liquidity can rebalance across ecosystems using proof-verified PPOs, without wrapping or bridging.

6.3 Integration Stack

Below is the intended stack. v1 of this repo includes the Contracts and early SDK pieces. Wallet surfaces, cross-chain relayers, and explorer tooling are part of upcoming milestones.

Layer	Component	Responsibility
Wallet	User Interface	Consent creation and revocation
SDK	Developer Tools	PPO verification and trigger handling
Contracts	Flow & Registry	Execution logic and consent enforcement
Relayers	Cross-Network Messaging	Propagation of PPO state
Explorer	Indexer	Consent visibility and analytics

The architecture remains modular: a single component can be adopted independently without requiring the full stack.

7. Governance and Standardization

7.1 Open Standard Formation

Recur Labs proposes the permissioned-pull model as an open-standard specification, designed for compatibility with existing Ethereum and EVM governance processes.

Future versions may formalize the following documents:

- RIP-001: Permissioned Pull Object Standard
- RIP-002: Consent Registry Interface
- RIP-003: Cross-Network Flow Intent Format

Each document remains open to external contributors, ensuring that no single entity controls the evolution of the standard.

Additional extensions (e.g., Non-Custodial Rebalancing, Flow Channels, Adaptive Routing) will follow as subsequent RIPs once the base layer stabilizes.

7.2 Stewardship

Recur Labs serves as initial steward, maintaining reference implementations and audits.

Once the specification stabilizes, control transitions to a community-governed working group consisting of wallet providers, stablecoin issuers, and developers.

7.3 Compliance and Transparency

Because all consent actions are public and cryptographically bound, Recur offers an audit-friendly environment without compromising self-custody. This architecture gives institutions auditable, user-signed proof of debit authority. Over time, that can evolve into compliance primitives (regulated pull, traceable consent trails) without resorting to custodial control.

8. Economic Impact

8.1 Efficiency

Push-based finance wastes time and liquidity. Balances remain idle between events, waiting for manual triggers. By enabling continuous liquidity, Recur increases capital velocity and reduces opportunity cost across the entire digital economy.

8.2 Stability

When liquidity can rebalance before imbalance, volatility dampens naturally. Protocols no longer need emergency liquidity injections or mass liquidations; flow restores equilibrium in real time. This transforms stability from a discretionary function (centralized policy) into a network property (distributed continuity).

8.3 User Autonomy

Recur eliminates custodial dependencies by turning trust into structure. Users control the boundaries of automation, not intermediaries. Each consent is explicit, revocable, and self-auditing; automation without surrender.

8.4 Macro Implications

As more systems adopt flow-based architecture, the macroeconomic rhythm changes:

- Capital efficiency rises.
- Volatility compresses.
- Cycles shorten and soften.

The result is not artificial control but structural freedom: markets that adapt continuously instead of collapsing periodically.

9. Roadmap

Phase	Milestone	Target
Q4 2025	Publish Technical Paper + RIP-001 → RIP-004 definitions (Magicians discussion live)	Complete
Q4 2025	v0.1 Reference Implementation (Ethereum / EVM compatible) — includes RecurPullSafeV2, RecurConsentRegistry, FlowIntentRegistry, CrossNetworkRebalancer, DomainDirectory, EVM Adapters, FlowChannelHardened, PolicyEnforcer, AdaptiveRouter, SettlementMesh, and UniversalClock	Live / Evolving
Q1 2026	Begin community review + interoperability tests across networks (Base, Arbitrum, etc.)	Planned
Q2 2026	SDK release + integration with wallet partners and stablecoin issuers	Planned
Q4 2026	Governance transition to open consortium (multi-network adoption)	Planned

Parallel initiatives include educational materials, testnet deployments, and developer incentive programs under Recur Labs stewardship.

10. Conclusion

Recur transforms motion into structure. Where legacy finance reacts after imbalance, Recur rebalances before it. Its permissioned-pull architecture turns liquidity from static inventory into living flow: continuous, safe, and consented.

This is not a new currency, chain, or token. It is the missing logic beneath them all: a universal grammar of value that makes equilibrium programmable.

By embedding consent directly into movement, Recur completes what digital money began: a system where freedom and stability are not opposites, but the same function expressed through flow.

— End of Technical Specification —

Appendix — Specification Preview (v0.1)

Defining the Permissioned-Pull Standard for Digital Value

1. Core Primitives

1.1 Permissioned-Pull Object (PPO)

A cryptographically signed object that defines the boundaries of consent for a pull. It lives client-side and can be revoked or updated at any time.

Data Schema (conceptual):

```
PermissionedPullObject {
   grantor: address,        // account granting consent
   grantee: address,        // account authorized to initiate pulls
   receiver: address,       // destination of funds
   token: address,          // ERC-20 / ERC-4626 / wrapped native
   maxAmount: uint256,      // maximum total amount authorized
   validAfter: uint256,     // earliest timestamp the PPO can be used
   validBefore: uint256,    // latest timestamp the PPO can be used
   nonce: uint256,          // unique salt for replay protection
   signature: bytes         // EIP-712 signature from grantor
}
```

Purpose:
Defines *what* may flow, *to whom*, *how often*, and *within what bounds.*

1.2 Consent Registry (CR)

A minimal, on-chain registry that records active Permissioned-Pull Objects by hash, not by data. It allows verification of consent without custody of funds or data.

Functions:

- verify(hash) → bool
- revoke(hash)
- status(grantor, receiver) → state

Purpose:
Makes consent verifiable, not trust-based; “don’t hope, verify flow.”

1.3 Flow Channel (FC)

A lightweight execution layer that performs the authorized pull within the bounds defined by the PPO.

Functions:

- pull(channelId, amount)
- claimable(channelId) → uint256

Purpose:
Executes movement of funds pre-approved by consent, not post-requested by push. Supports deterministic rebalancing between agents or protocols.

2. Architectural Principles

Principle	Description
Client-side Consent	Authorization always originates and resides on the client, not the receiver.
Reversible by Design	Consent can be revoked instantly without counterparty approval.
Interoperable	PPOs are network-agnostic: any EVM-compatible chain or L2 can verify them.
Non-Custodial	No custody or escrow is introduced; only pre-approved movement within defined limits.
Composable	PPOs can wrap existing ERC-20 logic and integrate with DeFi or payment rails.

3. Implementation Path (v1 Targets)

1. **Reference Implementation (Solidity):** basic PPO + Consent Registry + Flow Channel logic.
2. **SDK + JS Library:** client tools for creating, signing, and managing PPOs.
3. **Integration Templates:** open adapters for wallets, dApps, and stablecoin issuers.
4. **Audit & Standardization:** independent review and open discussion before formal standardization.

4. Scope Note

Recur does *not* issue, custody, or broker assets. It defines the continuity layer: a timing and consent standard that any existing wallet, network, or asset can adopt.