solpay

SOLPAY LITEPAPER

Programmable digital-dollar infrastructure
for people, businesses, and software

USDC powers payments	$SP powers participation

VERSION 0.1 | SANDBOX AND PRIVATE BETA | JULY 2026

The current release does not move real funds. Live capabilities remain gated pending provider, security, compliance, and operational readiness.

Abstract

Money can move globally at internet speed, but using digital dollars still asks too much of ordinary users and developers. Wallets, token accounts, chain confirmation, provider-specific APIs, and fragmented records turn a familiar action - paying or getting paid - into infrastructure work.

SolPay is a payment and account layer for USDC on Solana. It gives people and businesses familiar ways to fund an account, accept payments, send digital dollars, and track activity. It gives developers stable APIs for hosted checkout, payment links, transfers, webhooks, and software-controlled spending. Underneath, SolPay keeps provider integrations replaceable, verifies settlement evidence, and records every financial action in an auditable journal.

The current version delivers this contract as a complete sandbox. It is designed to prove the product experience and the safety model before live money movement is enabled.

1. The problem

Digital dollars combine a dollar-denominated unit with always-on, programmable settlement. That creates useful new possibilities: a business can accept a payment without banking hours, a marketplace can coordinate payouts in software, and an automated service can purchase an API call within a defined budget.

The underlying infrastructure, however, is still fragmented.

- Users must understand wallets, networks, token addresses, fees, and transaction finality.
- Developers must assemble checkout, provider APIs, chain monitoring, idempotency, webhooks, and reconciliation.
- Operators need one coherent view of payments, balances, transfers, fees, approvals, and exceptions.
- Software agents need limited authority, not an unrestricted wallet credential.
- A provider response alone is not enough to establish that money settled correctly.

The missing layer is not another wallet or token. It is a clear, reliable payment system that turns digital-dollar infrastructure into familiar financial actions.

2. The SolPay approach

SolPay makes USDC usable as account money and programmable as software.

For an end user, the product is intentionally simple:

1. Add money through an eligible, supported provider.
2. Receive USDC through checkout, payment links, or a connected wallet.
3. Send USDC to an approved person, business, or wallet.
4. Withdraw through supported methods when the relevant capability is available.
5. See a clear balance and activity history throughout the process.

For a developer, SolPay exposes the same system as composable primitives:

- payment creation and hosted checkout;

- fixed or customer-entered payment links;
- wallets, balances, transfers, and approvals;
- scoped API credentials and signed webhooks;
- policy-controlled accounts for software agents;
- immutable payment, fee, journal, audit, and reconciliation records.

Customers interact with SolPay rather than a chain-specific or provider-specific workflow. SolPay owns the payment lifecycle while external providers and Solana supply the underlying movement and settlement evidence.

3. Why USDC on Solana

SolPay begins with USDC on Solana because it offers a focused foundation for digital-dollar payments: a dollar-denominated asset, fast settlement, fine-grained amounts, and a network designed for high-throughput applications.

The initial focus is deliberate. SolPay does not present every token and network as interchangeable. Each asset and chain adds new settlement, liquidity, provider, operational, and compliance assumptions. A single-currency, single-network starting point makes behavior easier to understand and verify while leaving provider interfaces open to carefully evaluated future expansion.

USDC amounts are represented internally as integer atomic units with six decimal places. Financial calculations never use floating-point arithmetic.

4. Product primitives

Accounts and wallets

An organization can create projects and wallets for its own operations, customers, or software agents. Environments, credentials, and records are separated so test activity can never address live resources. On-chain balances remain authoritative for funds; SolPay's records explain activity and obligations.

Payments and hosted checkout

A merchant creates a payment with an amount, description, expiry, and destination. SolPay returns a hosted checkout that hides blockchain mechanics from the payer. A payment can become paid only after valid provider or chain evidence has been checked and the domain state machine permits the transition.

Payment links

Businesses can publish reusable links with a fixed amount or bounded customer-entered amount. The link uses the same payment lifecycle, receipt, journal, and webhook contracts as an API-created checkout.

Transfers and approvals

Transfers support recipient allowlists, transaction limits, and approval states. High-value activity can require a second authorized operator before provider submission. Every money-changing request requires an idempotency key, preventing a retry from becoming a duplicate transfer.

Software-controlled spending

SolPay treats an agent as a constrained financial principal, not as a copy of an owner's authority. Policies can limit each transaction and each day's total, restrict recipients, and require human approval. The policy input, version, decision, and resulting activity form an immutable audit trail. Unrestricted agent withdrawal is outside the product model.

Events and reconciliation

Signed webhooks notify merchant systems of committed state changes. Stable event identifiers, timestamp protection, retries, and replay handling make delivery safe to consume more than once. Provider notifications are reconciled against independent evidence before they authorize financial success.

5. System design

SolPay is built as a set of explicit trust boundaries:

```
User or merchant system
          |
          v
   SolPay UI and API
          |
          +--> domain rules and state transitions
          |            |
          |            +--> application records, journal, audit, outbox
          |            |
          |            +--> provider interfaces --> wallet/payment providers
          |
          +--------------------------------------> independent Solana evidence
```

The browser never receives provider or storage secrets and never talks directly to infrastructure providers. Storage operations pass through one repository boundary; provider operations pass through typed provider interfaces. This keeps product rules consistent and makes sandbox and production adapters implement the same contract.

Related financial changes are atomic. A valid payment transition, journal entries, audit facts, and outbound event are committed together. Journal entries are immutable and balanced; corrections use compensating entries rather than rewriting history.

6. Safety model

SolPay's safety model is based on explicit authorization, verifiable state, and recoverable operations.

Idempotency by default

Every money-changing operation is bound to an idempotency key and request fingerprint. Replaying the same request returns the original result. Reusing a key for different parameters fails.

State transitions, not status assignment

Clients cannot mark a payment paid. Status changes pass through a domain transition function, and terminal success requires checked provider or chain evidence.

Double-entry records

Payment and fee effects are represented by balanced journal entries. History is append-only, so refunds and corrections remain explainable.

Least authority

Credentials are scoped by organization, project, environment, and permission. Agent credentials identify one agent and expose only agent operations. Administrative and customer authority are separate domains.

Fail-closed capabilities

Live providers, onramps, offramps, payouts, and other sensitive features are controlled by environment-specific server flags. A missing credential, entitlement, verification, or operational gate disables the capability; hiding a button is never the control.

Test/live isolation

Test and live data do not share projects, records, credentials, endpoints, or provider configuration. Normal development and CI never require a production secret or submit a mainnet transaction.

7. Current release

The new SolPay version is a functional sandbox and private-beta foundation. It includes:

- a responsive marketing site and account experience;
- organization and project context with server-side role checks;
- test API credentials with scoped access;
- simulated Solana USDC wallets and activity;
- payment creation, hosted checkout, receipts, and refunds;
- fixed and variable payment links;
- immutable audit and double-entry journal records;
- balances, allowlisted transfers, and dual-control approvals;
- signed webhook delivery, retries, logs, and replay;
- constrained agent accounts with approved and denied policy decisions;
- a gated Crossmint staging onramp path for Solana Devnet USDC;
- API documentation and an OpenAPI contract.

No real funds move in the default build. The sandbox is not a mock front end: it exercises the domain rules, provider boundaries, persistence model, idempotency behavior, journal, audit trail, and event contract intended for production adapters.

8. Path to live service

SolPay will activate live capabilities in stages rather than treating deployment as proof of readiness.

Stage 1 - Sandbox and private beta

Validate checkout, payment links, transfers, agent policies, API ergonomics, tenant isolation, accessibility, failure handling, and operational workflows without real funds.

Stage 2 - Hosted staging

Exercise provider credentials, signed inbound and outbound webhooks, Solana Devnet USDC, reconciliation, rate limiting, monitoring, and incident procedures in an isolated hosted environment.

Stage 3 - Gated live payments

Enable a capability only after exact provider support, production credentials, account entitlement, compliance review, limits, fee disclosure, independent settlement verification, reconciliation, security review, and recovery procedures are documented and tested.

Stage 4 - Controlled expansion

Evaluate additional geographies, payment methods, treasury tools, agent use cases, and networks individually. Customer offramps, production agent spending, and x402 settlement remain disabled until their distinct requirements are satisfied.

9. The role of $SP

USDC powers payments. **$SP is intended to power participation and utility across the SolPay network.** The two assets have different jobs: USDC remains the payment and settlement asset, while $SP is designed as an optional access, loyalty, and service-utility layer.

Holding or using $SP will not be required to open a basic SolPay account, receive USDC, or use the standard product. Any $SP program introduced for mainnet beta will be opt-in, independently gated, and subject to published terms, technical review, and applicable law.

Access and account utility

The proposed utility model connects $SP participation to product capabilities rather than replacing conventional pricing. Final thresholds and benefits have not yet been set.

Access level	Proposed $SP relationship	Illustrative utility
Free	No stake required	Standard SolPay access
Plus	Stake $SP	Lower eligible fees and higher account limits
Business	Stake a higher amount of $SP	API access, team wallets, webhooks, and lower eligible processing fees
Agent	Stake or allocate $SP per agent	Agent wallets, programmable budgets, and eligible x402 access and limits

Access will continue to depend on geography, identity or business verification, risk controls, provider availability, and plan entitlements. Staking $SP cannot override a compliance restriction, authorize unsupported money movement, or give an agent unrestricted withdrawal authority.

Credits and lower service costs

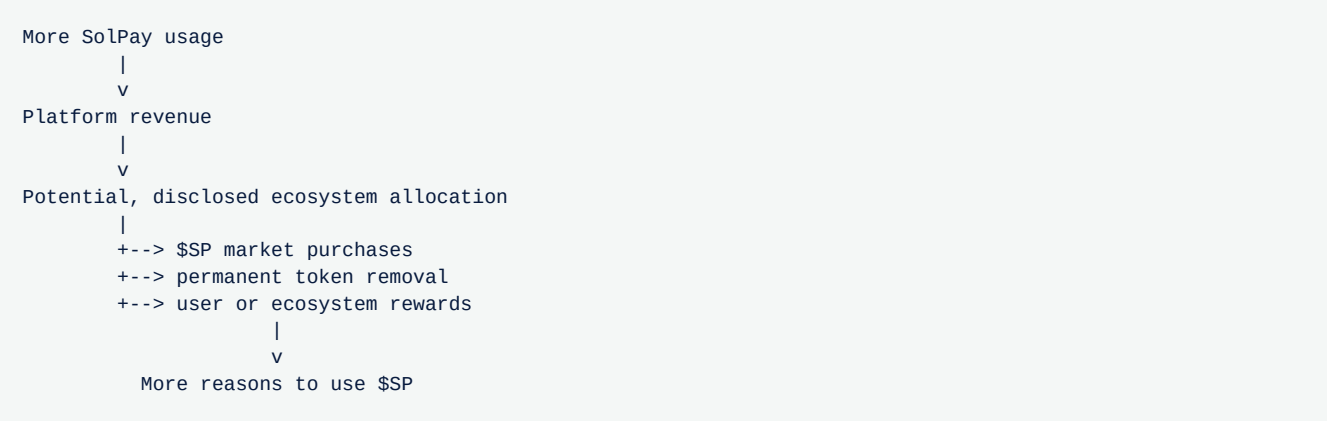
SolPay is evaluating two ways for $SP to reduce the cost of using the platform:

- qualifying $SP participation may earn non-cash SolPay service credits;
- users may be able to exchange or permanently remove $SP from circulation in return for defined service credits;
- credits may offset eligible SolPay fees, subject to expiry, usage, and jurisdiction rules;
- qualifying businesses may unlock lower eligible processing rates through transparent, versioned fee rules.

Service credits would apply only to disclosed SolPay charges. They would not conceal provider or network costs, represent a claim on customer funds, or guarantee that a transaction is free. Any credit earning, redemption, or burn mechanism will be published before activation and recorded in an auditable ledger.

A usage-aligned utility loop

The intended model aligns $SP utility with real SolPay usage:



This is a proposed utility loop, not a promise of price support, revenue sharing, profit, or investment return. SolPay may choose not to conduct purchases, burns, or rewards. Any such program would require a defined policy covering source of funds, limits, timing, custody, disclosures, conflicts, accounting, and legal treatment. Product revenue always belongs first to the safe operation of the platform and its obligations.

Mainnet Founders program

For a limited mainnet beta, SolPay may introduce a Founders program that recognizes early, constructive participation. Eligible activity could earn non-transferable Founders Points, reduced eligible fees, priority access to gated features, or eligibility for discretionary $SP rewards.

Points would measure participation, not ownership. They would have no cash value, would not guarantee an allocation or reward, and could not bypass product, risk, or compliance requirements. Final eligibility, anti-abuse rules, program dates, and rewards - if any - will be published separately before the program begins.

Activation gates

No staking, credit-redemption, buyback, burn, or $SP reward mechanism is active in the current sandbox. Before any of these features is introduced, SolPay intends to complete:

1. legal, regulatory, tax, and accounting review;
2. a published supply, treasury, custody, and authority model;
3. smart-contract and application security review where applicable;
4. auditable rules for staking, credits, fees, points, purchases, burns, and rewards;
5. jurisdiction, eligibility, sanctions, and anti-abuse controls;
6. clear risk disclosures and user terms; and
7. staged testing with monitoring, reconciliation, incident response, and a disable path.

10. Business model

SolPay is designed around transparent subscriptions, usage entitlements, and transaction fees. The sandbox models these rules without charging real money.

Potential revenue streams include:

- subscription plans based on projects, team access, agents, analytics, approvals, and support;
- disclosed processing fees on successfully verified payments;
- metered agent and machine-payment infrastructure;
- premium treasury controls and enterprise integrations;
- provider-enabled funding or withdrawal services where commercial and legal terms permit.

Fee calculations use versioned rules and integer arithmetic. Their inputs and outputs are retained so gross amount, provider or network cost, SolPay fee, and merchant net can be reproduced. No fee is earned on an unverified payment, and no provider markup is assumed without explicit authorization.

Illustrative prices and rates in internal product planning are not contractual offers. Production pricing will state included usage, billing period, transaction fees, overages, and material restrictions before a customer commits.

11. Scope and principles

SolPay's launch scope is intentionally narrow: digital-dollar accounts, USDC payments, hosted checkout, payment links, balances, transfers, developer tooling, supported funding, and controlled software spending.

The initial release does not include storefronts, catalogs, marketplace discovery, escrow, custom smart contracts, card programs, active $SP staking or reward mechanics, investment returns, or unrestricted automated withdrawals. The proposed $SP utility layer is optional, separately gated, and is not required to use the core product.

The product is guided by five principles:

1. **Make money understandable.** Users should see familiar actions and clear records, not infrastructure jargon.
2. **Verify before declaring success.** Application state alone cannot prove settlement.
3. **Make retries safe.** Networks fail; repeated requests must not repeat money movement.
4. **Give software bounded authority.** Automation should operate within explicit, inspectable rules.
5. **Earn live access.** Capabilities remain off until their technical, operational, commercial, and compliance assumptions are proven.

12. Vision

Software is becoming an active participant in commerce. Businesses increasingly need to collect, route, approve, and reconcile small amounts continuously. People still expect money products to be legible, predictable, and recoverable.

SolPay's goal is to connect those two realities: familiar digital-dollar accounts for people and businesses, and safe payment primitives for software. The long-term opportunity is a financial layer where a checkout, a marketplace payout, a software budget, or a paid API request can use the same clear rules and auditable settlement system.

Money should move with the clarity, speed, and control of software.

Important notice

This litepaper describes a product under development and is provided for informational purposes only. References to $SP, staking, credits, points, purchases, burns, or rewards describe possible future utility and are not commitments to launch those features. Nothing in this document is an offer to sell securities, a promise of token value or investment returns, legal or tax advice, or a representation that any regulated financial service is currently available. SolPay is a financial technology platform, not a bank. Digital assets, USDC, Solana, and third-party providers involve technical, market, liquidity, operational, and regulatory risks. Feature availability may depend on geography, eligibility, identity or business verification, provider support, and applicable law.