MACHINE
BRIDGE

FOUNDATION RELEASE / v1.0

Connect. Activate. Prove.

A wallet-native participation layer built around readable consent, public receipts, and limited non-transferable Points.

DOCUMENT	VERSION	DATE
Public Whitepaper	**1.0**	**September 2026**

> **Important:** Machine Points have no cash value and do not represent a token, token allocation, airdrop entitlement, financial return, or guaranteed future benefit. The optional Boost is a public transaction and network gas is paid separately.

Contents

Abstract 3

1. The problem 4

2. Product scope 5

3. Design principles 6

4. Wallet identity and verification 7

5. Machine Points 8

6. Public Proof 9

7. Optional Boost 10

8. Contract model 11

9. Application architecture 13

10. Data model 14

11. State recovery and idempotency 16

12. Security model 17

13. Privacy 19

14. Transparency and Activity 20

15. Economics 21

16. Governance and operations 22

17. Accessibility and user protection 23

18. Roadmap directions 24

19. Risks 25

20. No Token or Airdrop Commitment 27

21. Conclusion 28

Abstract

Machine Bridge is a wallet-native participation product built around a narrow and verifiable user journey. A user connects a self-custody wallet, confirms control with a readable and expiring message, receives a one-time base record of 100 non-transferable Machine Points, and may choose whether to create a Public Proof or submit one optional exact-value Boost transaction. A successful Boost adds 100 Points, producing a maximum of 200 Points per wallet.

The product separates events that interfaces often blur together: opening a wallet connection, proving wallet control, submitting a transaction, obtaining a successful receipt, reconciling that receipt, and updating an application record. A connected wallet is not automatically a verified wallet. A submitted transaction is not automatically successful. A successful transaction is not automatically an application credit until the expected sender, contract, method, value, event, block, and contract state have been checked.

Machine Points are participation records. They are not a token, currency, investment, ownership interest, debt instrument, revenue share, governance right, claim on treasury assets, or promise of redemption. The project may study future ecosystem programs, but this Whitepaper does not promise that a token, airdrop, distribution, snapshot, eligibility rule, allocation, conversion rate, return, or benefit will exist.

1. The problem

Wallet-connected products often compress too many meanings into one button. “Connect” can be presented as if it creates an account, proves identity, accepts terms, grants a reward, joins a community, and completes an on-chain action. In reality, a browser wallet connection normally exposes an address and basic permissions. It does not prove that the address should receive a durable application record, and it does not create a public transaction.

Paid participation flows introduce other ambiguities. An interface may display success as soon as a transaction hash appears, even though the transaction can still revert. A backend may credit the same hash twice. A wrong amount may be accepted through a raw transfer. A stale frontend may point to the wrong contract. A user may refresh during confirmation and then submit a duplicate payment. Statistics may be filled with placeholder activity that appears real.

Machine Bridge addresses these issues with a deterministic model:

1. wallet ownership is confirmed with a human-readable, replay-resistant message;
2. the base record is granted once to the verified wallet;
3. a Public Proof can be recorded through a zero-value contract call, with network gas paid separately;
4. one optional Boost of exactly 0.01 BOT can add 100 Points;
5. every public action is recognized only after receipt and state verification;
6. the interface uses empty and unavailable states instead of invented data.

2. Product scope

The foundation release contains:

- a public landing page;
- self-custody wallet connection;
- supported-network validation;
- readable ownership verification;
- a one-time 100-point base record;
- an optional zero-value Public Proof;
- an optional exact 0.01 BOT Boost;
- a 200-point maximum;
- an Account page;
- a public Activity page;
- a readable Whitepaper and downloadable PDF;
- Privacy and Terms pages;
- transaction recovery and reconciliation;
- real empty, loading, pending, success, reverted, stale, and unavailable states.

The foundation release is not:

- a live cross-chain asset bridge;
- a decentralized exchange;
- a lending market;
- a yield or staking product;
- a token issuance;
- an NFT sale;
- a prediction market;
- a guaranteed rewards program;
- an investment product;
- financial, legal, or tax advice.

The word “Bridge” describes the movement from a wallet connection to an inspectable participation profile. It does not represent a promise to transfer arbitrary assets across chains.

3. Design principles

3.1 Self-custody first

Machine Bridge does not request or store private keys or seed phrases. Wallet software remains the user's signing and transaction boundary. Every message and transaction must be approved in the wallet.

3.2 Readable consent

A user should understand the purpose, value, destination, expected result, gas responsibility, one-time limit, and Points effect before a wallet request opens. Opaque signatures and ambiguous paid calls are not acceptable.

3.3 Deterministic rules

The foundation rules are deliberately narrow:

```
Ownership verification: 100 Points once
Public Proof:           zero value, once, no Points change
Optional Boost:         exactly 0.01 BOT once
Boost credit:           +100 after verified success
Maximum:                200 Points
```

3.4 Proof before celebration

A hash means submission, not success. The interface waits for a successful receipt and verifies the expected transaction and state before it displays confirmation or updates Boost Points.

3.5 Honest empty states

No fake wallet, transaction, aggregate, activity, partner, audit, or user data is used to fill space. Unavailable data is labeled unavailable rather than displayed as zero.

3.6 Template fidelity

The project-owner template is the final visual and engineering base. Machine Bridge content and controls must be mapped into all major template sections and component families with at least 90/100 measured fidelity.

4. Wallet identity and verification

A wallet address is a user-controlled identifier, but a browser connection is not enough for a durable Points record. The ownership flow therefore uses a readable message with:

- application domain;
- origin URI;
- normalized wallet address;
- purpose statement;
- statement version;
- cryptographically random nonce;
- issued-at time;
- expiry time;
- optional request identifier.

The server verifies the signature, exact domain and origin, address, time window, nonce, and replay state. The nonce is consumed atomically with the base record. Concurrent requests cannot create duplicate base Points.

Ownership verification costs no BOT because it is a signature rather than a transaction. It should not request token approval, blind data, or a payment.

A wallet can disconnect, reconnect, or use another browser. Its base record remains bound to the normalized wallet address. Switching wallets must reload all profile and contract state rather than mixing records.

5. Machine Points

Machine Points are application-level participation records.

A verified wallet receives:

```
Base Points: 100
Source: ownership verification
Transaction hash: none
```

A confirmed Boost may create:

```
Boost Points: 100
Source: verified public transaction
Transaction hash: required
```

The maximum is 200. Machine Points cannot be:

- transferred;
- sold;
- withdrawn;
- staked;
- redeemed for BOT;
- exchanged at a promised rate;
- treated as cash or stored value;
- used as proof of token eligibility;
- used as proof of investment ownership.

The Points ledger must use database constraints and idempotency keys. No administrator interface should offer arbitrary positive or negative adjustments in the foundation release.

6. Public Proof

The Public Proof is an optional, one-time, zero-value contract call. It allows a verified wallet to create a public participation receipt without submitting the 0.01 BOT Boost amount.

The review surface identifies:

- wallet;
- configured contract;
- value: 0 BOT;
- network gas responsibility;
- one-time limit;
- Points change: none;
- expected `ProofRecorded` event.

Before opening the wallet, the application checks that the contract exists, the method is available, the contract is not paused, the wallet has not already recorded a proof, and simulation succeeds.

After submission, the interface displays a pending state and transaction hash. Confirmation requires a successful receipt, expected contract address, expected method, expected zero value, correct event, matching wallet, valid block range, and `proofRecorded(wallet) == true`.

The Public Proof does not imply identity verification beyond wallet control, legal status, reputation, financial value, or future eligibility.

7. Optional Boost

A verified wallet may submit one Boost transaction of exactly 0.01 BOT. The Boost is optional. The contract rejects a different amount and a repeated Boost. The value is forwarded to the approved Treasury in the same transaction. If forwarding fails, the transaction reverts and the Boost remains unused.

The application adds 100 Points only after verifying:

- the receipt succeeded;
- the configured internal environment is correct;
- the sender is the verified wallet;
- the destination is the configured contract;
- the decoded method is the approved Boost method;
- the value is exactly 0.01 BOT;
- the expected event exists and identifies the same wallet;
- contract state reports the wallet as boosted;
- the configured confirmation depth is reached;
- the hash has not already generated a credit.

A direct Boost may also create the Public Proof in the same transaction. A user choosing the paid path should not be forced to submit a separate free proof first.

8. Contract model

The reference Registry is intentionally small.

State:

```
immutable Treasury
proofRecorded[wallet]
boosted[wallet]
totalProofs
totalBoosts
totalBoostValue
paused
owner and pending owner
```

Methods:

```
recordProof()
boost() payable
pause()
unpause()
transferOwnership()
acceptOwnership()
```

Invariants:

- one proof per wallet;
- one Boost per wallet;
- exact Boost fee;
- no arbitrary amount input;
- direct Boost records a missing proof;
- successful Boost immediately forwards value;
- failed forwarding rolls back all state;
- direct payments and unknown calls revert;
- contract has no Points-minting function;
- pause controls block new actions;
- ownership transfer uses two steps.

The immutable Treasury reduces administrative change risk but increases deployment risk. The address must be independently checked and explicitly approved before deployment.

9. Application architecture

A production implementation normally contains:

1. **Public web application:** template-based marketing, Activation Terminal, Account, Activity, Whitepaper, Privacy, and Terms.
2. **Wallet layer:** connection, network guard, readable signatures, simulation, transaction submission, receipt polling, and pending recovery.
3. **Profile API:** nonce issuance, signature verification, profile reads, transaction registration, and reconciliation.
4. **Database:** wallet profiles, nonce records, Points ledger, transaction records, reconciliation attempts, and security events.
5. **Contract:** one Public Proof and one exact-value Boost per wallet.
6. **Chain reader/indexer:** event retrieval from the deployment block, finality policy, pagination, retries, stale markers, and deduplication.
7. **Operations:** logs, alerts, migration controls, backups, incident response, deployment evidence, and rollback.

No client-exposed variable may contain a deployer key, database secret, session secret, or private provider credential.

10. Data model

A wallet profile stores:

- normalized wallet address;
- ownership verification time and version;
- base Points;
- Boost Points;
- total generated from those fields;
- creation and update times.

A nonce record stores:

- wallet;
- nonce hash;
- domain and origin;
- statement version;
- issue and expiry;
- consume time.

A Points ledger stores:

- wallet;
- source;
- exact amount;
- unique idempotency key;
- optional transaction hash;
- creation time.

An on-chain transaction record stores:

- hash;
- wallet;
- action;
- environment;
- chain identifier internally;
- contract;

- value;
- submitted time;
- receipt status;
- block;
- event and state verification flags;
- reconciliation time.

Public activity should expose only the minimum useful public fields and should not add unrelated off-chain identifiers.

11. State recovery and idempotency

After submission, the frontend persists only non-sensitive pending data:

```
action
wallet
transaction hash
internal chain identifier
submitted time
```

On refresh, it validates that the connected wallet and environment match before resuming. It never silently sends a replacement transaction.

Reconciliation is idempotent. The same hash and wallet cannot create a second Points entry. Multiple workers, browser retries, webhook retries, and manual retry actions must converge on one result.

Important state distinctions:

- submitted;
- pending;
- confirmed on-chain;
- reconciled;
- reverted;
- timeout;
- provider unavailable;
- confirmed but reconciliation delayed.

A timeout is not automatically a failure. A provider error is not automatically a zero balance. A delayed reconciliation is not permission to resubmit the payment.

12. Security model

12.1 Wallet threats

Wallet extensions and mobile wallets can contain defects, be compromised, or present confusing prompts. Machine Bridge reduces risk through explicit review screens and limited contract methods, but cannot secure the user's device.

12.2 Signature replay

Short expiry, random nonces, exact origin/domain binding, atomic consumption, and server-side verification reduce replay risk. The signed message should not authorize unrelated actions.

12.3 Transaction substitution

The frontend simulates the exact configured method. The backend verifies sender, destination, method, value, event and state. A transfer to another address or a call to another method does not create Boost Points.

12.4 Duplicate credit

Database unique constraints, transaction-hash uniqueness, wallet-level uniqueness and idempotency keys prevent duplicate base or Boost credits.

12.5 Provider failure

Reads can be delayed, inconsistent or unavailable. The application uses retries, bounded timeouts, confirmation depth, stale markers and an unavailable state. It should compare independent sources during operations when practical.

12.6 Contract administration

The owner can pause/unpause and start a two-step ownership transfer. The owner cannot mint Points or withdraw arbitrary user balances from the Registry. Treasury selection remains a critical operational decision.

12.7 Frontend supply chain

Dependencies, wallet libraries, analytics, remote scripts and deployment accounts can be compromised. The project should pin dependencies, minimize third-party scripts, use CSP, protect deployment access, scan dependencies, review diffs and retain rollback capability.

13. Privacy

Wallet addresses and public transactions are public identifiers. They can be correlated with other activity. Users should choose a wallet consistent with their privacy preferences.

The application should minimize off-chain collection. It does not need a legal name, home address, phone number, private key, seed phrase or arbitrary wallet history for the foundation flow.

Operational logs may use short retention, hashed IP prefixes and hashed user-agent data for abuse prevention. Privacy disclosures should describe providers, cookies, analytics, retention, rights, public-chain permanence and contact procedures accurately.

Deleting an application profile cannot delete a public transaction from the underlying ledger.

14. Transparency and Activity

The public Activity page may display:

- action;
- shortened wallet;
- amount;
- confirmed time;
- transaction link;
- block;
- sync status.

Aggregates must be derived from the configured deployment block and identified as delayed or unavailable when reads fail. No placeholder activity, fabricated “live” feed, fake partners, fake audits or fake user counts are acceptable.

The Account page distinguishes:

```
Ownership verification: application record
Public Proof: public transaction
Boost: public transaction
```

This source labeling prevents users from assuming that every Point exists as a transferable on-chain asset.

15. Economics

The foundation release has one optional paid action:

```
Amount: exactly 0.01 BOT
Frequency: once per wallet
Destination: approved immutable Treasury
Gas: paid separately
Refund: not automatically available after confirmation
Points effect: +100 after verification
Maximum total Points: 200
```

The 0.01 BOT amount is not an investment price, token sale price, exchange rate, deposit balance, yield position, or claim on Treasury assets.

The Treasury may use received value for lawful project operations, development, infrastructure, security, community programs or other purposes described in the final Terms. Specific use claims must not be made unless approved and operationally true.

16. Governance and operations

The foundation release is operated by the project owner rather than by token governance.

Operational responsibilities include:

- approving Owner and Treasury addresses;
- maintaining secure deployment credentials;
- reviewing contract source and dependencies;
- running tests and independent security review;
- controlling Vercel and database access;
- applying migrations and backups;
- monitoring RPC, contract and reconciliation health;
- responding to incidents;
- keeping public copy accurate;
- reviewing Privacy and Terms;
- documenting deployments, upgrades to surrounding applications and rollbacks.

A future governance model is not promised.

17. Accessibility and user protection

The interface should support keyboard operation, visible focus, semantic headings, labels, error association, reduced motion, readable contrast, responsive text, mobile wallet return paths and screen-reader announcements for transaction status.

Paid consent must remain understandable without relying only on color. The exact amount, destination, gas responsibility and expected Points effect should appear as text.

18. Roadmap directions

Phase A - Foundation

Wallet verification, 100 base Points, optional Public Proof, optional exact 0.01 BOT Boost, Account, Activity, Whitepaper, Privacy, Terms, test deployment, security review and Vercel release.

Phase B - Proof vocabulary

Standards-based attestations, clearer source metadata, portable receipt formats and improved activity indexing.

Phase C - Community programs

Clearly governed opt-in participation programs, transparent rules and evidence sources, privacy controls and reviewed dispute processes.

Phase D - Interoperability research

Account-abstraction support, proof portability, independent security/privacy review and compatibility research.

These are directions, not commitments. They may change, be delayed or be cancelled. They do not promise a token, distribution, eligibility or financial benefit.

19. Risks

Smart-contract risk

A contract defect can cause reversion, incorrect state or misdirected value. Testing and review reduce but do not eliminate risk.

Treasury risk

Boost value is forwarded to the configured Treasury. A wrong immutable Treasury is a serious deployment error and must be approved and verified before deployment.

Wallet risk

Users may approve the wrong prompt, use compromised software or lose wallet access. Machine Bridge cannot recover a wallet or reverse public transactions.

Application and database risk

Base Points depend on application verification and database integrity. Backups, constraints, access controls, migrations and reconciliation are required.

Provider risk

RPC, database, wallet, hosting and rate-limit services can be unavailable or inconsistent. UI states must distinguish delay from failure.

Privacy risk

Public wallet and transaction data can be correlated. Users should choose a wallet consistent with their privacy preferences.

Regulatory and tax risk

Points programs and native-asset payments may be treated differently across jurisdictions. Appropriate legal, regulatory, accounting and tax advice may be required.

Expectation risk

Users may speculate about future rewards despite disclosures. The product and community must not imply a guaranteed token or airdrop.

Template and intellectual-property risk

High-fidelity template adaptation must respect the template's license and avoid copying the reference project's protected brand, data, content and assets.

20. No Token or Airdrop Commitment

Machine Points do not represent:

- a token or token allocation;
- a claim on a future token;
- an airdrop or snapshot entitlement;
- a conversion formula;
- cash or stored value;
- equity, debt, revenue share or governance rights;
- interest, yield or profit;
- ownership of Treasury assets;
- a guaranteed service or product benefit.

The project may evaluate future ecosystem programs. Evaluation does not guarantee that a program will launch or that a wallet will qualify. Any future decision would require separate rules, legal analysis, security design, implementation, public terms and communication. This Whitepaper is not that decision.

Users should not submit the optional Boost based on an expectation of financial return or future distribution.

21. Conclusion

Machine Bridge begins with a limited, inspectable model:

```
One self-custody wallet.
One verified base record.
100 Machine Points.
One optional public proof.
One optional exact 0.01 BOT Boost.
A maximum of 200 Points.
No guaranteed token, airdrop, return, or future benefit.
```

The foundation's value is not complexity. It is the separation of connection, consent, proof, transaction, receipt and application state. Preserving those boundaries allows future participation experiments without misleading users about what already exists.