THE #PEG ENGINE

A Specification for Draw-Based Monetary Allocation under Proof of Luck

ABSTRACT

This document describes the #PEG engine, a draw-based monetary allocation mechanism that introduces purely denominative monetary units and allocates them through execution constrained by chance.

A #PEG unit doesn't carry intrinsic economic meaning: it is not a claim, asset, entitlement, or representation of value, and the Engine encodes no policy, valuation logic, or theory of worth. Allocation occurs without reference to identity, effort, capital, or merit, and outcomes are recorded as monetary facts without corrective or stabilizing mechanisms.

The #PEG engine is presented as a structural monetary system rather than a functional one. It specifies the minimal conditions under which monetary units may be denominated and allocated, while deliberately abstaining from defining how those units should be valued, exchanged, or governed. Any economic significance or monetary behavior associated with #PEG units arises externally, at the System level, and is not specified or assumed here.

The purpose of this document is not to propose an improved monetary model, but to articulate a formally minimal monetary infrastructure and to examine the consequences of separating denomination and allocation from justification and policy.

TABLE OF CONTENTS

1. INTRODUCTION

Most monetary systems begin by answering a question they rarely make explicit: *who is entitled to money, and why?*

Work, risk, capital, trust, sovereignty, merit, or governance are invoked in different combinations, but the gesture is always the same. Money is issued, allocated, or stabilized on the basis of a justification. Even systems that claim neutrality or inevitability encode such justifications implicitly, as if money could not exist without them.

The #PEG engine starts from a different premise. It asks whether money can exist *prior* to entitlement—before value, policy, or justification enter the picture at all. This is not an attempt to improve money, correct it, or make it fairer. It is an attempt to suspend the very questions that usually organize monetary thought and to observe what remains.

The originality of the #PEG design lies neither in the use of draws as such nor in the simulation of a lottery mechanism, but in the conjunction of draw-based execution with a purely denominative unit. The unit introduced by the Engine carries no intrinsic economic meaning: it is not a claim, not an asset, not a promise of redemption, not a representation of work, risk, or merit. It is a unit of denomination only, and it is allocated exclusively through execution constrained by chance. In doing so, the engine separates denomination from valuation and allocation from justification, producing monetary facts without embedding an economic theory of worth.

This move places #PEG outside the lineage of existing monetary forms. It does not compete with them, improve upon them, or extend them. Instead, it intervenes at a lower level—below economics, below policy, below entitlement—at the point where money first becomes legible as a unit that can be counted, allocated, and recorded at all. The distinctiveness of this intervention is not incremental but categorical.

Five features make this conception of money incomparable:

First, **denomination without valuation**. The unit exists without a theory of value attached to it. No purchasing power, peg, or backing is defined or assumed.

Second, **allocation without justification**. Allocation occurs without reference to identity, effort, capital, history, or merit. Chance functions not as spectacle, but as a structural constraint that prevents justificatory logic from entering the system.

Third, **monetary facts without monetary policy**. Outcomes are produced and recorded, but no feedback, correction, or stabilization mechanism is introduced. Measurement exists without management.

Fourth, **money whose meaning is externalized**. The engine produces outcomes, but it does not interpret them. Any economic

meaning arises outside the system, through use, exchange, or expectation.

Fifth, **a new ontological category of money**. What is introduced is neither a currency, nor a stablecoin, nor a commodity, nor a credit instrument. It is a minimal monetary infrastructure that provides denomination, allocation, and finality —and nothing else.

This allows a distinction to be drawn between two kinds of monetary systems. *Functional* monetary systems define how money should behave, be valued, and be managed. *Structural* monetary systems define only the conditions under which something can count as money at all. #PEG belongs to the latter category. It is best described as a pre-economic monetary system: one that establishes form without prescribing function.

None of this prevents #PEG units from acquiring monetary significance beyond the engine. They may circulate, be priced, or exhibit stable or quasi-stable behavior as a result of repeated use and expectation. Such behavior may even display Markovian characteristics, emerging from participation rather than design. But these dynamics are not specified, assumed, or governed here. They belong to the system that forms around the engine, not to the engine itself.

This document is therefore not a proposal for a better monetary order. It is an invitation to consider a different one. Not a system that tells us what money should do, but one that asks what money is, once we stop telling it what it is for.

Terminological and symbolical conventions.

In this document, the term **Engine** refers exclusively to the #PEG engine, the execution layer responsible for draw instantiation, draw register resolution, and allocation recording. The term **Protocol** refers exclusively to the Proof of Luck (PoL) protocol, which defines execution admissibility under publicly verifiable randomness. The term **System** refers to the #PEG System as a whole, encompassing the engine, external access contexts, intermediaries, interfaces, markets, and all social, legal, and economic arrangements that arise around or beyond engine execution. These terms are used consistently and are not interchangeable.

The prefix "#" denotes a purely denominative unit produced by the #PEG engine. It does not indicate a currency, asset, claim, entitlement, backing, or redemption right, and carries no economic or institutional semantics beyond denomination within the Engine.

The generic currency sign "¤" is not used, as it conventionally denotes an unspecified currency and thus implies monetary status or interchangeability that the #PEG engine explicitly does not assume.

The document is structured around a strict separation between three conceptual layers:

The Proof of Luck (PoL) — Protocol

Proof of Luck (PoL) is a protocol that admits executions solely on the basis of a verifiable random outcome produced according to predefined rules.

At the protocol level, Proof of Luck (PoL) specifies a verifiable randomness source and a deterministic rule for interpreting each random value as an admissibility condition. Execution proceeds if and only if this condition is satisfied.

The PoL protocol determines no subject, entitlement, or recipient. It constrains execution by chance alone, without reference to identity, effort, capital, history, reputation, or merit. Concretely, the Protocol governs only whether a draw execution is admissible under publicly verifiable randomness; it does not select participants or determine allocation outcomes, which follow mechanically from #PEG engine execution.

The #PEG Engine — Monetary Denomination & Allocation Engine

The #PEG engine is the layer that produces monetary outputs by denominating and allocating #PEG units through repeated PoL executions.

The engine operates over primitive denominative units (#PEG), allocates fixed quantities of those units according to draw outcomes, and records and exposes the resulting monetary facts (e.g. allocation amounts, Pay out Ratio). It implements no stabilization, redemption, valuation, synthetic derivation, or corrective logic.

Its outputs are monetary facts, not economic interpretations. They describe what was allocated, not what it is worth or how it should be used.

The engine allocates monetary value without conducting monetary policy.

The #PEG System — Total External Domain

The #PEG system comprises everything outside the Engine boundary that interacts with, interprets, reacts to, or ignores #PEG monetary outputs.

This includes, without limitation:
- participants, distributors, and interfaces,
- liquidity, exchange, and conversion practices,
- regulatory classification and institutional response,
- economic valuation, incentives, and behaviors,
- narratives, expectations, and cultural interpretations.

The System may stabilize, exploit, reject, or reinterpret #PEG outputs. None of these actions feed back into the PoL protocol or the #PEG engine.

The System assigns meaning; the Engine does not.

This separation is constitutive/definitional. It determines not only how the #PEG engine operates, but also how it must be described. Any behavior not enforced by deployed code lies outside the Engine's boundary.

Purpose and Limits of This Document

This document is a specification. It does not advocate for applications of #PEG, define policy objectives, or prescribe use cases. It does not attempt to justify randomized allocation as fair, efficient, or desirable. It specifies how the #PEG engine operates and delineates the boundaries of that operation.

The document is intended to function simultaneously as:
- a standalone technical specification,
- a canonical boundary object for analytical work,
- and a reference for comparative analysis within a plural monetary landscape.

All normative judgment, strategic interpretation, and institutional response lies outside its scope.

PART I — PROTOCOL AND ENGINE SPECIFICATION

2. THE #PEG ENGINE'S DESIGN CONSTRAINTS

The #PEG engine is defined by a set of non-negotiable design constraints. These constraints delimit what the Protocol and the Engine can and cannot do. They are enforced structurally through deployed contract logic and do not rely on governance, discretion, interpretation, or external coordination.

Once a #PEG draw is deployed, these constraints apply irrevocably for the lifetime of that draw.

2.1 Irrevocable Execution

#PEG draws are instantiated as smart contract instances deployed without administrative privileges, pause mechanisms, or upgrade paths. Once deployed, contract logic and configuration parameters cannot be modified, reversed, or overridden.

There are no privileged keys, emergency controls, or discretionary intervention mechanisms. Execution proceeds deterministically according to deployed code and supplied inputs.

If a draw is executed, its outcome is final.

The Protocol does not support rollback, recovery, exception handling, or post-execution modification. Any change to logic or parameters requires the deployment of a new, independent draw instance. Forks or parallel deployments operate independently and have no effect on existing draws.

2.2 Absence of Governance

The #PEG engine does not include on-chain or off-chain governance mechanisms. There are no voting rights, councils, committees, parameter adjustment procedures, or adaptive control processes.

The PoL protocol does not respond to participation levels, economic conditions, valuation signals, or external feedback. Once deployed, the Engine's behavior does not evolve.

All coordination, interpretation, mitigation, or modification occurs outside the Engine boundary and has no effect on deployed draws.

2.3 Randomized Allocation

The #PEG engine allocates outcomes exclusively through publicly verifiable random selection as defined by deployed draw logic.

All randomness used for #PEG draw execution is publicly verifiable: any observer can independently verify the randomness source, the admissibility rule, and the resulting execution decision without permission, trust, or privileged access.

Allocation outcomes are independent of:
- participant identity,
- capital size,
- participation history,
- contribution frequency,
- or prior draw outcomes.

Each draw constitutes an independent execution event. Neither the PoL protocol nor the #PEG engine retain any historical state beyond what is required for transaction execution and settlement. This excludes any participant profiling, reputation history, or cross-draw memory; only instance-local execution state and the underlying ledger record persist.

Random selection is applied mechanically and without qualification. The Engine does not evaluate, weight, or rank participants beyond the satisfaction of entry conditions.

2.4 Permissionless Initiation and Participation

Initiation of #PEG draws is permissionless at the Engine level. Any address may deploy a draw by defining its entry conditions.

Participation in #PEG draws is permissionless at the Engine level. Any address may enter a draw by satisfying its deployment-defined entry conditions.

No identity verification, whitelisting, eligibility filtering, or reputational assessment is applied by the Proof of Luck (PoL) protocol, and the #PEG engine records no reputational, behavioral, or historical state beyond what is strictly required for execution.

The #PEG engine assigns no roles, privileges, or tiers. All valid allocation commitments are treated equivalently at execution time, regardless of origin, aggregation, or submission pathway.

2.5 Fixed Operational Scope

The #PEG engine does not define objectives beyond the execution of draws as specified at deployment.

The Engine does not target: price stability, capital preservation, growth, efficiency, inclusion, or policy outcomes.

The Engine does not attempt to correct, optimize, or adapt its behavior. Performance metrics and monetary outcomes are observable consequences of execution, not operative targets.

Any interpretation of outcomes occurs outside the Engine's boundaries.

2.6 External Dependence

The #PEG engine depends on external infrastructure for execution, including:
- the underlying blockchain,
- transaction inclusion mechanisms,
- verifiable randomness sources,
- and any external data referenced at execution time.

The Engine does not embed incentive mechanisms for infrastructure provision. Continued operation depends on voluntary, economic, or ideological support external to deployed contracts.

The specification therefore includes *no liveness guarantee*: continuation is an external contingency, not a protocol property.

Failure, degradation, or withdrawal of supporting infrastructure may prevent draw execution or settlement. The Engine includes no internal remediation or substitution mechanisms.

3. #PEG ENGINE FUNDAMENTALS

The #PEG engine operates through a limited set of execution primitives. These primitives define how draws are instantiated, how participation is handled, and how outcomes are produced

and settled. The Engine includes no auxiliary logic beyond what is required for draw execution.

This section describes only those elements that are enforced mechanically by deployed contract logic.

3.1 Engine Preconditions (Standing Conditions)

The #PEG engine has no bootstrap phase: it defines no genesis state, initial allocation, privileged participants, or system-level activation event. Any preparatory deployment, testing, or early draw execution occurs outside the engine boundary and confers no structural privilege.

The #PEG engine exists as a standing allocation logic whose execution is instantiated only through the deployment of individual draws. #PEG draw instantiation therefore marks the beginning of a draw lifecycle, not the activation of the engine itself.

The instantiation of any #PEG draw presupposes:

(i) an execution environment capable of deploying and executing irrevocable smart contracts;

(ii) the availability of a publicly verifiable randomness mechanism compatible with PoL protocol standards;

(iii) a published draw contract specification defining entry handling, execution timing, randomness consumption, payout rules, and wallet-addressed settlement.

(iv) a #PEG draw register for each draw instance: a bounded, Engine-internal data structure that aggregates all valid entries and their associated #PEG-denominated allocation commitments submitted prior to execution. The draw register exists solely for the duration of the draw lifecycle and is exhaustively resolved at execution time. It does not custody balances or assets; it records prospective allocations that become effective only upon execution.

Because the draw register records only non-transferable allocation commitments and not spendable balances, it is structurally non-lootable: no #PEG units can be spent without allocation via execution.

(v) #PEG units. A #PEG unit is a purely denominative unit in which allocation commitments and resulting allocations are expressed. Outside draw execution, #PEG may circulate as a conventional monetary reference within external systems, but such circulation does not imply backing, redemption, custody, or recognition by the #PEG engine.

These preconditions are necessary for #PEG draw instantiation but do not guarantee it. Any decision to instantiate a draw and coordinate participation occurs outside the Engine boundary.

3.2 #PEG Draw Instantiation

Each draw is instantiated as an independent smart contract instance and thereby initiates an individual draw lifecycle within the #PEG engine.

At deployment, a draw is configured with a fixed set of parameters, including execution timing, entry conditions, payout structure, and denomination identifiers. These parameters are supplied as configuration inputs and are immutable after deployment.

Draw instances are isolated. There is no shared mutable state across draws beyond the underlying blockchain ledger. No draw inherits parameters, privileges, or outcomes from any prior instance.

Once instantiated, a #PEG draw executes exactly once.

3.3 Entry and Allocation Commitment

Participation in a #PEG draw requires submission of an allocation commitment denominated in the unit specified at deployment.

Entry conditions are defined at draw instantiation and applied uniformly to all participants. The Engine does not impose limits on:
- the number of participants,
- frequency of participation,
- aggregation of entries,
- or submission pathways.

Entries submitted directly by participants and entries submitted through intermediaries are treated equivalently by the #PEG engine at execution time; any differences in access, aggregation, or post-execution handling occur outside the Engine boundary.

Entries that do not satisfy deployment-defined conditions are rejected without exception.

3.4 Random Selection

Winner selection is performed using publicly verifiable randomness supplied to the execution environment.

Selection is independent of:
- participant identity,
- allocation commitment size,
- participation history,
- prior outcomes.

The PoL protocol does not weight entries, apply multipliers, or retain state influencing probability. Each valid entry is subject to the same selection logic as defined by the #PEG draw parameters.

Random selection is executed mechanically. Outcomes cannot be predicted, influenced, or revised by #PEG engine logic.

3.5 Execution Finality

Upon execution, a #PEG draw register resolves immediately at the Engine level, with no intermediate or deferred state.

Payouts are allocated strictly according to the deployed rules.

In this document, "allocation" refers to the assignment of #PEG-denominated settlement outcomes to wallet addresses under #PEG draw rules; selection determines which entries receive those outcomes.

In this document, "settlement" refers exclusively to the Engine-level recording of allocation outcomes to wallet addresses. It does not imply economic realization, liquidity, availability, or post-allocation handling, all of which occur outside the Engine boundary.

All #PEG engine-level settlements are effected by assigning outcomes to wallet addresses. Wallets are treated as addressable containers within the execution environment and are not interpreted as identities, beneficiaries, or subjects.

There is no post-execution phase. The Engine does not permit: appeals, corrections, retries, compensatory actions, or discretionary intervention.

Execution finality is defined solely at the engine level and does not imply immediate settlement, redistribution, or availability in external systems.

Executed draws terminate without residual control surfaces. All outcomes are final.

A draw lifecycle is finite and reaches a terminal state at its scheduled execution time. Terminal resolution may occur either through execution, producing allocation outcomes, or through cancellation, producing no allocations. Finality applies to both terminal paths: once a draw has executed or been cancelled, no further state transitions occur within the Engine.

3.6 Absence of Adjustment Mechanisms

The #PEG engine includes no mechanisms for price targeting, supply modification, parameter tuning, or adaptive response.

The protocol does not monitor:
- external markets,
- participation trends,
- valuation signals,
- or system-level behavior.

There is no notion of deviation, correction, or stabilization within the #PEG engine execution logic.

Each draw executes independently under identical rules.

3.7 Draw Lifecycle Termination

After execution, #PEG draw contracts become inert.

No state is preserved beyond transaction records required for settlement and verification. The Engine does not include renewal, rollover, continuation, or memory mechanisms.

If required inputs are unavailable at execution time, the draw does not settle; no alternative execution path exists within the instance, and any resolution beyond the deployed rules lies outside the Engine boundary.

Repeated use of the #PEG engine requires repeated instantiation of new, independent draws.

Finality in the #PEG engine denotes irreversibility of state; termination denotes lifecycle completion. In the Engine, all terminal states are final, but finality refers to outcome irreversibility, not merely to lifecycle end.

A draw reaches finality either through execution, producing allocation outcomes, or through cancellation, producing no allocations. In both cases, finality signifies that no further execution, reversal, or state transition is possible within the Engine.

3.8 The #PEG Engine Process (Single Draw Lifecycle)

This subsection defines the #PEG engine process for a single draw lifecycle, from draw configuration through terminal resolution. The process is strictly finite: every draw reaches a terminal state at its scheduled execution time, and no draw may terminate with outstanding commitments recorded in the draw register.

Steps 1 through 5 define the pre-execution phase of a draw. During this phase, a draw is configured, entries are submitted and validated, the draw register is aggregated, and the register is closed at the scheduled execution time. No execution logic, randomness consumption, or allocation occurs prior to register closure.

At the scheduled execution time, the Engine attempts to acquire the publicly verifiable randomness required for execution (Step 6) and performs the Proof of Luck admissibility check (Step 7). This check determines only whether draw execution may proceed; it does not select participants or allocation outcomes.

No postponement or retry logic exists within the Engine. The admissibility check constitutes a hard gate: at the scheduled

execution time, the draw transitions immediately to a terminal resolution path.

If execution is admissible, the Engine resolves the draw deterministically using the closed register and the acquired randomness (Step 8A). Allocation outcomes are produced mechanically, recorded during settlement (Step 9A), and the draw register is cleared as commitments are consumed by execution (Step 10A). The draw then reaches the terminal state Executed.

If execution is not admissible at the scheduled execution time, the draw is immediately cancelled (Step 8B). Cancellation is terminal and produces no allocation outcomes.

Upon cancellation, the engine attempts to release all recorded allocation commitments to their originating submission wallets (Step 9B), restoring the corresponding #PEG units to the externally transferable state. "Release" denotes a deterministic reversal of a prior Engine-internal recording, not a right, guarantee, or entitlement.

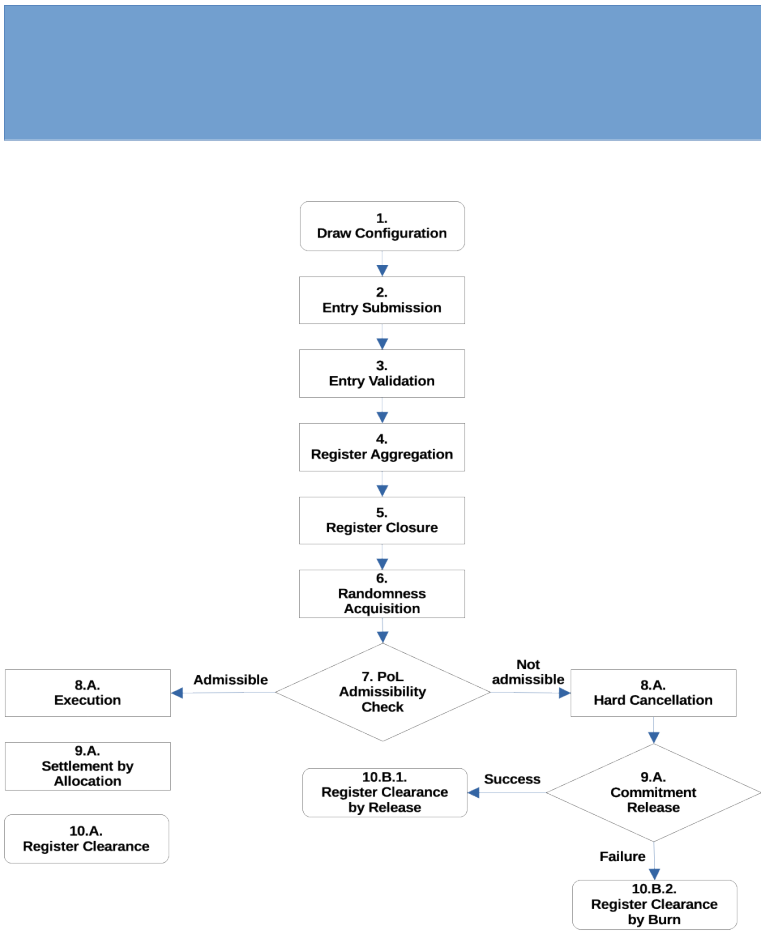
If commitment release cannot be performed deterministically, the engine burns the unreleasable commitments (Step 10B.2). Burning is non-discretionary and final, and serves solely to enforce draw termination and register clearance.
In all terminal cases—execution, cancellation with release, or cancellation with burn—the draw register is cleared and contains no remaining commitments.

Step	Process Step Title	Inputs	Engine Action	Output / State Change	Notes / Invariants
1	**Draw Configuration**	Initiator parameters (denomination, draw type, scheduled execution time, caps, aggregation mode, acceptance scope, identifiers/metadata)	Records draw configuration and opens draw for entry	Draw instance created; register opened	No execution logic, valuation logic, or entitlement logic created here
2	**Entry Submission**	Entries submitted (direct or mediated) with inlay / commitment amount	Accepts valid entries under configuration constraints	Entry recorded in draw register	Engine records submission wallet as the sole provenance handle
3	**Entry Validation**	Entry fields, caps, format, admissibility preconditions	Validates format and acceptance constraints	Accepted entries retained; invalid rejected	Validation is not identity/reputation filtering; it is structural admissibility
4	**Register Aggregation**	Accepted entries	Aggregates entries according to configured aggregation mode	Consolidated register state	Aggregation affects register representation only, not execution semantics
5	**Register Closure**	Scheduled execution time reached	Closes register to new entries	Register becomes immutable for this draw	Closure is mechanical at scheduled time
6	**Randomness Acquisition**	Publicly verifiable randomness source	Attempts to retrieve randomness required for execution at scheduled time	Randomness proof present OR absent	No postponement window; this is a hard gate
7	**PoL**	Randomness proof +	Determines	**Admissible** or	PoL determines no

Step	Process Step Title	Inputs	Engine Action	Output / State Change	Notes / Invariants
	Admissibility Check	deterministic consumption rule	whether execution is admissible	**Not admissible**	subject/recipient; it gates execution only
8A	**Execution** *(if admissible)*	Closed register + randomness	Resolves draw deterministically and computes allocations	Allocation mapping produced	Allocation outcomes are a mechanical consequence of execution
9A	**Settlement by Allocation** *(if executed)*	Allocation mapping	Records allocations to destination wallets	Allocations finalized; draw marked **Executed**	Execution finality is immediate at this step
10A	**Register Clearance** *(if executed)*	Register + settlement result	Clears register commitments as consumed by execution	Register becomes empty	**Register-zero invariant** satisfied
8B	**Hard Cancellation** *(if not admissible)*	Closed register + failed admissibility	Marks draw **Cancelled** immediately	Cancellation event recorded	Cancellation is terminal at scheduled time
9B	**Commitment Release Attempt** *(if cancelled)*	Register entries + submission wallets	Attempts to release commitments back to originating submission wallets	Release succeeds OR fails	Release restores transferability of #PEG balances at submission wallets
10B-1	**Register Clearance by Release** *(if release succeeds)*	Release result	Clears register commitments	Register becomes empty; draw remains **Cancelled**	**Register-zero invariant** satisfied; no allocations exist
10B-2	**Register Clearance by Burn** *(if release fails)*	Unreleasable commitments	Burns unreleasable commitments deterministically	Register becomes empty; draw marked **Cancelled (Burn)**	Burn is non-discretionary, terminal, and recorded explicitly

Table 1 — #PEG Engine Process (Single Draw Lifecycle)

The table above transposes into the following process workflow chart.



Explicit Non-Steps (Boundary Clarification)

The following are not #PEG engine process steps and must not be represented as such:

- denomination (property, not a step)
- aggregation (system-level coordination)
- observability / PoR reporting (non-operative)
- distribution, liquidity, valuation, conversion
- governance, stabilization, redemption
- incentive provision or liveness guarantees

All of the above belong to the #PEG system, outside the Engine boundary.

4. #PEG ENGINE FUNCTIONAL COMPONENTS

The functional components described in this section realize the execution sequence specified in §3.8 and do not introduce additional steps, logic, or control beyond that canonical process. The #PEG engine relies on a limited set of functional components required for #PEG draw execution. These components perform specific operational roles necessary for deployment, execution, and settlement. The PoL protocol does not evaluate, optimize, or coordinate their performance beyond successful availability at execution time.

This section describes only those components insofar as they are required for full draw execution.

4.1 Smart Contract Execution

Deployment of a #PEG draw contract instantiates a single #PEG draw lifecycle; it does not activate, configure, or modify the #PEG engine, which remains invariant across draws and exists as a standing allocation logic prior to and independently of any specific draw.

Each draw is instantiated as a separate contract instance with fixed configuration parameters defined at deployment. Contract execution follows the rules encoded in the deployed bytecode.

The #PEG engine does NOT include:
- administrative functions,
- override permissions,
- modification interfaces,
- upgrade paths.

Each #PEG draw execution proceeds deterministically given valid inputs at runtime. Once deployed, the contract logic remains unchanged for the lifetime of the draw.

4.2 Randomness Source

The #PEG engine relies on an external source of publicly verifiable randomness to determine draw outcomes.

Randomness is supplied to the execution environment and consumed by the smart contract according to deployed rules. The Engine does not generate randomness internally.

The Engine verifies randomness only insofar as required for execution correctness within the execution environment. It does not assess statistical quality, provenance, or external trust characteristics beyond successful verification at execution time.

If publicly verifiable randomness is unavailable or invalid at execution time, the draw fails without fallback or substitution.

4.3 External Data References

Certain draw parameters may reference external data values supplied at execution time.

Such references may include denomination identifiers or execution-related values provided through external data sources accessible to the execution environment.

External data references, where used, parameterize execution of a specific #PEG draw instance; they do not provide price feeds, parity enforcement, or stabilization inputs.

The #PEG engine:
- consumes referenced data as supplied,
- does not validate correctness beyond syntactic or verification checks required for execution,
- does not monitor persistence, continuity, or accuracy of external data sources.

Failure or inconsistency of external data sources results in execution failure or degraded behavior without protocol-level response.

4.4 Infrastructure Dependencies

At execution time, the engine treats all valid entries equivalently by resolving the draw register without regard to entry provenance (direct or intermediary).

The #PEG engine depends on underlying public blockchain infrastructure for:
- transaction inclusion,
- state persistence,
- and execution ordering.

The Engine does not embed incentive mechanisms for:
- block production,
- transaction prioritization,
- network maintenance,
- or data availability.

Continued operation depends on the ongoing availability of external infrastructure maintained for reasons independent of the Engine. Loss or degradation of such infrastructure may prevent draw execution or settlement without internal mitigation.

In the event of draw cancellation, any release of allocation commitments is performed exclusively to the originating submission wallets; where entries are aggregated or mediated, downstream redistribution remains entirely external to the #PEG engine.

4.5 Distributors

Distributors facilitate access to #PEG draws externally by aggregating entries, providing interfaces, or coordinating participation.

The #PEG engine does not recognize distributors as a distinct class. Distributors do not receive Engine-level privileges and are subject to the same execution rules as any other participant.

All allocation commitments submitted through distributors are treated equivalently at execution time. Distributor behavior occurs entirely outside the Engine boundary and does not affect execution logic.

5. MONETARY PROPERTIES

The monetary behavior associated with the #PEG engine arises from repeated draw execution under fixed PoL protocol constraints. The Engine itself has no defined economic objectives, targets, or optimization criteria.

All monetary quantities in this section are #PEG-denominated expressions of allocation commitments (pre-execution) or recorded allocations (post-execution).

All properties described in this section are descriptive and observable. They are not operative and do not influence PoL protocol behavior.

5.1 Allocation Commitment and Draw Settlement Allocation

Each draw records participant allocation commitments denominated in the #PEG unit specified at deployment.

The total allocation commitment represents the maximum amount available for allocation, subject to engine-level execution frictions; system-level frictions affect participant outcomes only outside the #PEG engine. The Engine allocates a portion of the total allocation commitment to selected participants according to the deployed payout rules.

No additional issuance, leverage, or reserve supplementation occurs. #PEG draw settlement allocation is strictly bounded by the allocation commitment provided for that draw.

5.2 Denomination as Reference Frame

#PEG draws are denominated in reference units identified at deployment (e.g. €PEG, $PEG, AUPEG – the latter being the denomination for gold).

Denomination functions as a reference frame for accounting and comparison. It does not constitute:
- a claim on reserves,
- a redemption guarantee,
- or an enforceable parity with any external asset or currency.

Denomination is not a monetary peg: the engine enforces reference-formatting only and makes no commitment to parity, redemption, or market price alignment.

The #PEG engine enforces denomination syntactically. It does not enforce monetary equivalence.

Denominated outcomes produced by #PEG draw execution are therefore legible in reference terms without being stabilized or defended by the Engine.

5.3 The Pay out Ratio (PoR): Definition and Components

The Pay out Ratio (PoR) of a #PEG draw is an ex post observable ratio expressing the proportion of #PEG-denominated allocation commitments that result in allocations following draw execution. The PoR is computed solely from facts produced by draw register resolution and has no meaning prior to execution.

For a given draw, the PoR is defined as the ratio between:
- the total #PEG-denominated allocations recorded at execution, and,
- the total #PEG-denominated allocation commitments recorded in the draw register prior to execution, after deduction of engine-level execution frictions.

Engine-level execution frictions are not post hoc deductions from participant balances; they are execution costs that bound the maximum allocable amount during settlement.

The PoR is defined only for draws that reach execution; cancelled draws produce no allocation outcomes and therefore no PoR.

5.4 Engine and System-level #PEG Draw Frictions

Engine-level execution frictions are costs incurred strictly within the #PEG engine during draw execution. These may include protocol-defined execution costs, computational costs, and peg maintenance costs where applicable. Engine-level frictions are applied mechanically and uniformly and are independent of draw outcomes, participant identity, or access context.

System-level frictions are costs incurred outside the #PEG engine, including but not limited to access fees, intermediary commissions, custody costs, conversion spreads, taxation, regulatory compliance, or liquidity constraints. System-level frictions do not enter PoR computation and have no effect on draw execution, allocation, or Engine-level outcomes.

The PoR reflects only the internal efficiency of a #PEG draw execution within the Engine and does not represent participant-level profitability or realized economic return; such participant-relative outcomes are captured, where relevant, by Net Draw Return (NDR), a system-level analytic external to the engine (see section 10).

5.5 The Pay out Ratio (PoR): Calculation and Reporting

For each draw, the #PEG engine reports a Pay out Ratio (PoR) as defined in 5.3. The reported PoR is observable post-execution and does not influence subsequent draw execution or configuration.

The PoR is:
- calculated post-execution,
- observable and reportable,
- non-operative.

The #PEG engine does not act on PoR values. No minimum, maximum, or target ratio is defined. Previous PoRs do not influence future draws or execution logic.

5.6 Execution Costs and Engine-level Execution Frictions

Execution of a #PEG draw incurs Engine-level execution frictions that reduce the amount of commitment allocations available for settlement allocation.

Engine-level execution frictions refers to the aggregate of such costs and may include:

(i) Draw execution costs
Costs required to execute the #PEG draw according to the protocol specification, including:
- smart-contract execution costs,
- transaction processing costs,
- state transition costs required for draw register resolution.

(ii) Randomness acquisition and consumption costs
Costs incurred to obtain, verify, and consume publicly verifiable randomness at execution time, including:
- oracle invocation costs,
- cryptographic verification costs,
- randomness beacon interaction costs (where applicable).

(iii) Allocation recording costs
Costs required to:
- record allocation outcomes,
- assign allocations to destination wallets,
- finalize execution state irreversibly.

(iv) Peg maintenance costs (engine-internal)
(Contingent, but PoR-relevant when present)
Costs incurred by the engine to maintain the internal denominative consistency of #PEG units with their reference (e.g. basket, index), including:
- oracle reading costs for reference values,
- protocol-defined adjustment mechanics (if any).

(v) Engine-internal accounting or verification overhead
Minimal costs associated with:
- internal consistency checks,
- admissibility verification,
- draw register integrity enforcement.

The above costs are intrinsic to execution, are strictly limited to what is required for randomness consumption and allocation finality, and exclude any post-execution handling or processing.

The Protocol does not expose engine-level execution frictions as standalone values; such frictions are accounted for implicitly by reducing the allocable #PEG amounts at execution and are reflected only in post-execution allocation outcomes and in the reported Pay out Ratio (PoR). The #PEG engine does not allocate #PEG units to infrastructure providers; any compensation of execution infrastructure occurs outside the engine and does not form part of draw settlement.

Only engine-level execution frictions enter PoR computation. All system-level frictions are external to the #PEG engine and affect participants exclusively outside draw execution. These external frictions will be discussed in the second part of this specification.

5.7 Absence of Economic Controls

Only the PoR and allocations are observable.

The #PEG engine does not include mechanisms for controlling:
- price,
- supply,
- demand,
- participation behavior,
- or denomination alignment.

The #PEG engine does not intervene to increase efficiency, smooth outcomes, or stabilize observed metrics.

All economic effects result from voluntary #PEG draw initiation and participation and other external conditions.

5.8 Observability Without Intervention

All engine-level monetary facts produced by #PEG draw register resolution (including allocation amounts and PoR) are publicly observable.

Participants and external observers may compare:
- participation levels,
- draw outcomes,
- PoR values,
- and draw configurations across independent draw instances.

The Engine does not interpret these observations. Any decision-making based on observed outcomes occurs entirely outside its boundary.

5.9 Denomination Persistence as External Condition

Repeated #PEG draws denominated in the same reference unit may, under favorable external conditions, produce outcomes that are treated as economically legible over time.

Such persistence:
- is not enforced by the #PEG engine,
- is not guaranteed,
- and may degrade without corrective response.

The Engine does not defend denomination. It exposes the cost of doing so externally.

6. ACCESS CONTEXTS AND DRAW INSTANTIATION CONDITIONS

#PEG draws may be accessed in a variety of external contexts depending on infrastructure availability, access methods, and organizational arrangements. Draw instantiation additionally presupposes the standing preconditions described in Section 3.1. The #PEG engine does not prescribe, privilege, or adapt to any particular access context.

All draw instantiations follow the same execution rules once deployed.

6.1 Direct Participation

Participants may enter #PEG draws directly by submitting transactions that satisfy the draw's deployment-defined entry conditions.

Direct participation requires access to the underlying public blockchain and the ability to interact with deployed draw contracts.

All valid entries are processed uniformly at execution time.

6.2 Non-Recognition of Access Contexts

The #PEG engine does not encode assumptions about participant intent, economic motivation, or usage patterns.

The Engine neither recognizes nor evaluates access contexts. All variation in usage arises from external organization and participant behavior outside the Engine boundary.

6.3 #PEG Draw Initiation Parameters

A draw initiator may specify only the following parameters.

(i) Draw denomination
- Parameter: Denomination unit (#PEG instance, e.g. #PEG, AUPEG, $PEG)
- Scope: Engine
- Effect: Determines the #PEG unit in which allocation commitments and allocations are expressed.
- Constraint: Does not imply valuation, backing, or exchange rate.

(ii) Draw execution time

The execution time is a hard, non-deferable boundary: at this time the draw transitions immediately to a terminal resolution, either by execution or by cancellation.
The #PEG engine does not implement postponement, retry, or grace-period logic with respect to execution time.
- Parameter: Scheduled execution time (or execution window, if supported)
- Scope: Engine
- Effect: Determines when the #PEG draw register is resolved.
- Constraint: Does not affect randomness, allocation logic, or eligibility.

(iii) Draw type
- Parameter: Draw type identifier (e.g. hope, ambition, greed)
- Scope: Engine
- Effect: Selects a predefined allocation profile.
- Constraint: Draw types are fixed templates; initiators cannot modify their internal structure.

(iv) Entry acceptance conditions
- Parameter: Entry acceptance rules (open / restricted / mediated)
- Scope: Engine boundary (enforced externally if needed)
- Effect: Determines which entries may be admitted into the draw register.
- Constraint: Cannot discriminate at execution time; enforcement occurs prior to entry.

(v) Entry aggregation rules
- Parameter: Aggregation mode (individual entries vs batched entries)
- Scope: Engine
- Effect: Determines how entries are recorded in the draw register.
- Constraint: Aggregation does not affect execution-time equivalence.

(vi) Draw instantiation limits
- Parameter: Maximum total allocation commitment (cap) and/or maximum number of entries
- Scope: Engine
- Effect: Bounds the size and cap of the #PEG draw register.
- Constraint: Does not affect selection probability or allocation rules.

(vii) Draw identifier and metadata
- Parameter: Draw identifier and optional descriptive metadata
- Scope: System-facing
- Effect: Enables external referencing, indexing, and interpretation.
- Constraint: Metadata has no semantic effect on execution or allocation.

The parameter set above is exhaustive. Any configuration not listed here occurs outside the #PEG engine and has no effect on draw execution, allocation, or on the PoR.

(viii) Explicit non-parameters

The draw initiator cannot specify or influence:
- randomness source or randomness outcome,
- admissibility rules under Proof of Luck,
- selection probabilities,
- allocation mechanics beyond predefined draw types,
- Pay out Ratio (PoR),
- engine-level execution frictions,
- system-level frictions,
- execution finality conditions,
- post-execution allocation handling.

7. COMPARATIVE POSITIONING

The #PEG engine operates within a plural monetary landscape composed of heterogeneous allocation and issuance architectures. It does not assume replacement, convergence, or institutional adoption. The Engine coexists with other monetary and crypto-economic systems without requiring interoperability, coordination, or recognition.

This section situates #PEG engine relative to other monetary and protocol forms by describing differences in allocation logic, stabilization mechanisms, and control surfaces. No hierarchy, normative ranking, or performance claim is implied.

7.1 Coexistence Within a Plural Monetary Landscape

Contemporary monetary systems operate in parallel across jurisdictions, infrastructures, and institutional arrangements. No single architecture satisfies all economic functions or use cases.

Fiat currencies are issued administratively and derive validity from state authority, legal tender status, and monetary policy enforcement. Custodial stablecoins maintain reference value through reserve backing and redemption guarantees. Central bank digital currencies (CBDCs) extend state money into regulated digital form. Community-based credit instruments rely on social coordination and trust. Scarcity-based crypto-assets constrain issuance through protocol-defined limits.

The #PEG engine does not replicate these models.

Within the Engine, #PEG-denominated allocation commitments are allocated only through draw execution. Participation is permissionless and non-selective. The Engine records no state beyond what is required for execution and settlement.

#PEG therefore constitutes a distinct design category: a "non-governed, draw-based denominated monetary unit allocation engine" whose operation depends solely on contract execution and continued access to external infrastructure.

7.2 Comparative Overview of Monetary Architectures

The table below summarizes structural differences between #PEG and other monetary or protocol architectures. It is intended as a classificatory aid only.

Monetary / Protocol Architecture	Primary Allocation / Maintenance Logic	Source of Stability or Legitimacy	Governance / Control Layer	Pegged's Structural Difference
Fiat Currency	Administrative issuance by state authority	Legal tender status, chartal character, monetary policy	Central bank and political institutions	No issuer authority; no legal mandate
Asset-Backed Stablecoins	Minting against custodial reserves	Redemption guarantees and reserve audits	Issuer governance, compliance obligations	No reserves; no redemption promise
Algorithmic Stablecoins	Supply adjustment via feedback mechanisms	Market incentives and algorithmic tuning	Parameter updates, governance interventions	No adjustment logic; no feedback loops
PoW Scarcity Protocols ($BTC)	Proof-of-Work issuance with capped supply	Costly verification and scarcity constraints	Informal governance via forks and miner coordination	No work, no scarcity, no cumulative advantage
Programmable Execution Layer Protocols ($ETH)	Fee-based issuance with evolving monetary policy	Network usage, smart contract execution	Active protocol governance and upgrades	No protocol governance or upgrade path
Central Bank Digital Currencies (CBDCs)	State-issued digital ledger entries	Sovereign guarantee and policy enforcement	Full administrative and regulatory control	No administrative issuance; no compliance/surveil-lance layer; no discretionary control
Community Credit Instruments	Mutual or trust-based issuance	Social obligation and reputational enforcement	Informal governance and social coordination	No trust or obligation layer
Random Denominated Unit Allocation Protocols (#PEG)	Irrevocable, random allocation via draws	Observable behavior and Pay out Ratios (PoR)	None (post-deployment)	Outcome allocation by chance; indifference by design

7.3 Absence of Stabilization and Adjustment

The #PEG engine does not include mechanisms for price targeting, supply adjustment, or corrective intervention.

Unlike algorithmic stabilization models, the Engine does not attempt to maintain parity through feedback loops, arbitrage incentives, or governance-mediated tuning. There is no notion of deviation or correction within the execution logic.

Observable metrics such as the PoR expose execution efficiency but remain non-operative. They do not trigger protocol response.

7.4 Denomination and Observability

As described in §5.7, the #PEG engine does not enforce or defend denomination persistence over time. #PEG draws are denominated in reference units (e.g. €PEG, $PEG, AGPEG). These units do not represent claims on reserves and are not redeemable at fixed rates.

Denomination functions as a reference frame for participation and accounting. Each completed draw reveals the effective allocation achieved in that unit through the reported PoR.

If repeated draws remain efficient over time, denominated outcomes may be treated as economically legible in use. If efficiency degrades, no compensatory mechanism exists. The #PEG engine neither guarantees nor defends denomination.

7.5 Scope and Limits

The #PEG engine introduces a draw-based allocation architecture that operates alongside existing monetary systems.

It does not replace other forms of money, does not coordinate with them, and does not embed assumptions about adoption or dominance.

The Engine defines a fixed structure for allocation by chance. Its operation is independent of monetary issuer authority, governance processes, or policy objectives. Continued operation depends solely on voluntary initiation and participation and external infrastructure availability.

PART II — CONTEXT AND EXTERNAL CONSIDERATIONS

8. POST-LAUNCH OPERATIONS AND NON-PROTOCOL STAKEHOLDERS

The #PEG engine does not define post-deployment operations beyond draw settlement. Once deployed, draw contracts execute autonomously and terminate without residual control surfaces.

Any Systemic activity occurring after deployment—including access facilitation, coordination, interpretation, or mitigation—takes place entirely outside the Engine boundary and does not modify PoL protocol behavior.

The #PEG engine provides no guarantees regarding participation outcomes, execution success, or recovery of commitments beyond the constraints explicitly specified at the engine level. Any expectations of continuity, compensation, retry, or mitigation arise solely within the surrounding system context and are not enforced, assumed, or recognized by the engine itself.

The following sections describe #PEG System-level conditions under which the Engine may be accessed, instantiated, or repeatedly engaged; none of these conditions are specified, recognized, or enforced by the Engine itself.

8.1 Mediated Access, Distributors, and Interface Providers

Access to #PEG draws may be facilitated by external actors providing interfaces, aggregation services, custodial access, or coordination mechanisms. Such actors may include web applications, mobile tools, custodial services, or community-based organizers.

These intermediaries may aggregate entries, manage user experience, provide informational framing, or impose access conditions and fees. Any such conditions, restrictions, representations, or costs operate entirely outside the #PEG engine.
The #PEG engine does not recognize mediated access, distributors, or interface providers as distinct or privileged operational categories. Entries submitted through intermediaries are treated identically to direct entries at execution time, and intermediary behavior does not affect draw execution or outcome determination.

8.2 Distribution Environments

#PEG draws may be instantiated in environments with varying levels of infrastructure reliability, regulatory oversight, and participant familiarity with digital systems.

The #PEG engine does not adapt to environmental conditions. Execution behavior remains invariant across geographic, regulatory, or infrastructural contexts. Differences in outcomes arise from external conditions rather than Engine response.

8.3 Repeated Instantiation

The #PEG engine does not include mechanisms for recurring or continuous draws.

Each #PEG draw requires explicit instantiation as a separate contract instance. No state, privilege, or parameter inheritance occurs across draws.

Repeated use of the #PEG engine therefore depends on external organization and coordination.

8.4 Infrastructure Operators

Post-deployment operation of #PEG draws depends on continued availability of underlying infrastructure, including:
- blockchain networks,
- transaction relayers,
- randomness providers,
- and external data sources referenced at execution time.

Infrastructure operators may maintain these services for economic, ideological, or incidental reasons. The #PEG engine does not coordinate with, incentivize, or compensate such operators.

Withdrawal or degradation of infrastructure services may prevent draw execution or settlement without triggering any protocol-level response.

8.5 Observers and Analysts

The #PEG engine produces publicly observable execution data (see section 5.6).

External observers may analyze this data to assess participation efficiency, denomination behavior, or usage patterns.

Such analysis has no effect on Engine behavior. The Protocol neither incorporates external feedback nor responds to interpretation, critique, or strategic behavior inferred from observed data.

Operational dependencies required for draw execution are evaluated strictly at the scheduled execution time. The Engine does not implement retry, delay, or fallback mechanisms in response to temporary unavailability of external services. If required external inputs are not available at execution time, execution does not occur.

8.6 Absence of Operational Authority

No actor possesses authority to intervene in #PEG engine operation once a draw is deployed.

There are: no administrative roles, no escalation procedures, no discretionary powers, and no post-deployment controls embedded in the Engine.

Any coordination, mitigation, or adaptation in response to outcomes occurs outside the Engine boundary and has no effect on existing or future #PEG draw execution logic.

8.8 System-level frictions

These frictions occur outside the #PEG engine. They may affect participants economically but never affect draw execution or PoR computation.

(i) Access and interface costs
- front-end usage fees,
- API access fees,
- wallet service fees,
- UX-related charges.

(ii) Intermediary and distributor fees

- broker commissions,
- initiation fees,
- aggregator margins,
- batching or facilitation fees,
- distribution spreads.

(iii) Custody and wallet costs
- custodial wallet fees,
- safekeeping charges,
- key-management services,
- account maintenance costs.

(iv) Conversion and liquidity frictions
- DEX slippage,
- CEX spreads,
- order-book depth constraints,
- on/off-ramp conversion fees,
- volatility during conversion.

(v) Regulatory and compliance costs
- KYC/AML compliance expenses,
- reporting obligations,
- jurisdiction-specific licensing or taxation,
- enforcement-related delays or penalties.

(vi) Temporal and settlement delays
- exchange settlement delays,
- withdrawal waiting periods,
- batching or clearing cycles,
- banking system delays.

(vii) Informational and cognitive costs
- participant misunderstanding,
- informational asymmetry,
- strategic misinterpretation of PoR,
- behavioral biases.

These frictions vary by intermediary and jurisdiction and are external to the #PEG engine. They affect external pricing and participant experience only, do not alter draw execution, allocation outcomes, or execution finality, are not recognized by the engine, and are explicitly out of scope for this specification.

9. #PEG AND THE ECONOMICS OF CHANCE

The #PEG engine operates through randomized allocation as its sole allocation mechanism. This design situates the #PEG engine (operating under PoL constraints) within a broader class of economic arrangements that incorporate chance as a structuring principle rather than as a corrective or optimization tool (see section 7.2).

In the #PEG engine, randomness is not used to improve outcomes, balance incentives, or approximate fairness according to external criteria. It functions as a procedural condition of execution. The PoL protocol does not evaluate results, enforce

allocation patterns, or attempt to align outcomes with normative expectations.

9.1 Randomness as Procedure, Not Justification

The use of chance in the #PEG engine does not constitute a normative claim about fairness, justice, or efficiency.

Randomness is specified here as an allocation procedure, not advanced as a theory of justice, welfare, or fairness.

Randomized allocation is implemented as a fixed procedural rule that removes discretionary choice from Protocol-level allocation. By doing so, the Engine constrains avenues for influence, negotiation, or strategic positioning within its own execution logic.

The #PEG engine does not assert that random allocation is superior to alternative mechanisms. It merely enforces it as an irrevocable condition of participation. Any evaluation of this condition—whether favorable or critical—occurs outside the Engine boundary.

9.2 Comparison with Deterministic Allocation Mechanisms

Most contemporary monetary and distribution systems rely on deterministic criteria such as contribution, stake, identity, eligibility, or historical participation. These criteria embed assumptions about merit, entitlement, predictability, or behavioral incentives.

The #PEG engine incorporates none of these criteria. Allocation outcomes are determined exclusively by draw execution. The Engine does not record, accumulate, or act upon participant history, economic status, or contribution beyond the submitted allocation commitment required for entry.

This places the #PEG engine outside systems designed to reward effort, optimize incentives, or manage behavior through feedback. It also distinguishes it from mechanisms intended to correct perceived inequities through targeted intervention.

9.3 Behavioral Interpretation and External Response

Participants and observers may interpret randomized outcomes in various ways, including as expressions of neutrality, indifference, unpredictability, or exposure to risk.

Such interpretations may influence:
- participation patterns,
- usage contexts,
- narrative framing,
- or external organizational arrangements.

The #PEG engine does not respond to these interpretations. Participation decisions, risk perceptions, and strategic behavior

occur entirely outside the Engine boundary and do not alter execution logic.

9.4 Limits of Randomized Allocation

Randomized allocation does not guarantee specific economic outcomes.

It does not ensure:
- equal results,
- sustained participation,
- denomination persistence,
- or economic stability.

#PEG draw outcomes may vary significantly depending on participation conditions, execution costs, and external frictions. The #PEG engine does not mitigate such variability; it exposes it through observable execution data and reported metrics such as the PoR, and implements no corrective or compensatory mechanisms.

10. USE CASES AND DISTRIBUTION DYNAMICS

The #PEG engine does not define, privilege, or optimize for specific uses. The protocol specifies a mechanism for draw-based allocation of reference-denominated units; any application of that mechanism arises from external organization, participant behavior, and contextual constraints.

All usage patterns described in this section are external arrangements. They do not modify Protocol or Engine behavior and are not encoded as objectives or functions.

10.1 The Net Draw Return (NDR)

Specific usage may result in a draw specific Net Draw Return. The NDR is a System-level, participant-relative quantity expressing the proportion of #PEG-denominated allocation commitments that result in realized economic outcomes for a given participant or access context, after accounting for both Engine-level execution frictions and System-level frictions.

The NDR is not computed, reported, or recognized by the #PEG engine and has no effect on draw execution, allocation, or the PoR.

The PoR may serve as an input to external estimation of the NDR, but the NDR varies across participants and access contexts and is not a property of the draw itself.

10.2 Participation and Coordination Contexts

#PEG draws may be accessed through individual participation, pooled entry, or coordinated group arrangements, reflecting local practices, infrastructural constraints, cultural familiarity with

chance-based allocation, or organizational preferences. The #PEG engine does not distinguish between these contexts: all valid entries are treated equivalently at execution time, and any coordination or pooling logic operates externally and has no effect on draw execution or outcome determination.

10.3 Mediated Distribution and Access Structures

Access to #PEG draws may be facilitated by external actors providing interfaces, aggregation services, or custodial arrangements. These structures may influence accessibility, transaction costs, participation patterns, or user experience.

The #PEG engine does not recognize mediated distribution as a distinct operational mode. Entries submitted through intermediaries are processed identically to direct entries at execution time. All economic and organizational arrangements between participants and intermediaries operate outside the Engine boundary.

10.4 Environmental and Contextual Variation

#PEG draws may be instantiated across environments with differing levels of infrastructure reliability, taxation, regulatory oversight, financial inclusion, cultural acceptance and participant familiarity with digital systems.

Such variation may affect access methods, execution costs, or denomination legibility. The #PEG engine does not adapt to environmental conditions. Execution behavior remains invariant across cultural, geographic, regulatory, or infrastructural contexts.

10.5 Non-Prescriptive Scope of Use

The #PEG engine does not prescribe intended uses such as payments, savings, speculation, redistribution, or institutional settlement. It embeds no incentives or constraints favoring any particular application.

Participants may adopt #PEG draws for purposes aligned with their own objectives or constraints. The Engine neither encourages nor discourages such uses and does not adapt based on observed patterns.

10.6 Non-#PEG draw parameters

Besides the #PEG Draw Initiation Parameters define in 6.3, the following parameters may be chosen by initiators or intermediaries outside the engine, but are not draw parameters:
- access fees or participation costs,
- intermediary commissions,
- custody arrangements,
- KYC/AML requirements,
- interface design,
- batching strategies beyond engine aggregation rules,
- post-allocation handling or redistribution.

These affect the NDR, not the PoR.

11. LIMITATIONS AND OPEN QUESTIONS

The #PEG engine defines a fixed execution structure with no internal mechanisms for adaptation, recovery, or correction. As a result, its operation is subject to a set of limitations that arise directly from its design constraints and its dependence on external conditions.

These limitations are not flaws to be addressed within the Engine. They are structural consequences of deliberate design choices.

11.1 Dependence on External Infrastructure

Pegged relies on the continued availability of external infrastructure, including:
- public blockchain networks,
- transaction inclusion mechanisms,
- publicly verifiable randomness providers,
- and external data sources referenced at execution time.

The #PEG engine does not embed incentives, guarantees, or redundancy mechanisms for the maintenance of this infrastructure.

Under strict execution semantics, a draw may terminate without execution if required conditions are not met at the scheduled execution time. In such cases, allocation commitments may be released or, if release cannot be deterministically performed, irreversibly destroyed. This behavior reflects a design choice to enforce finite draw lifecycles and to prevent persistent locked states within the Engine.

Failure, degradation, or withdrawal of external services may prevent draw execution or settlement. No fallback, substitution, or recovery mechanisms exist within the protocol.

11.2 Participation Variability

Participation levels may vary significantly across #PEG draws due to external factors such as:
- access costs,
- coordination practices,
- regulatory constraints,
- participant expectations,
- or narrative framing.

Low participation may reduce allocation efficiency or increase the relative impact of execution costs. High participation may increase contention for outcomes.

The #PEG engine does not regulate participation levels, smooth variability, or respond to participation dynamics.

11.3 Denomination Risk

#PEG draws are denominated in reference units without enforceable parity mechanisms.

Denominated outcomes are not claims on reserves and are not redeemable at fixed rates. If participation efficiency degrades or external costs increase, outcomes may diverge from reference expectations.

The Engine does not intervene to defend denomination, restore alignment, or compensate participants for divergence.

11.4 Irreversibility of Outcomes

All draw outcomes are final.

Execution errors, misconfigurations, misunderstandings of #PEG draw parameters, or unintended consequences cannot be corrected once a draw has been initialised.

Draw initiators bear full responsibility for configuration choices. Participants bear full responsibility for entry decisions. The #PEG engine provides no appeal, compensation, or remediation mechanisms.

11.5 Absence of Governance and Evolution

The #PEG engine does not include governance mechanisms nor upgrade paths.

Design limitations, emergent behaviors, or external challenges cannot be addressed within deployed instances. Any modification or improvement requires the deployment of new, independent draws or parallel protocols.

Existing draws remain unaffected by subsequent design changes, forks, or reinterpretations.

12. RISK SURFACES AND EXTERNAL MITIGATION PATTERNS

The #PEG engine exposes a set of risk surfaces that arise from its execution model, anonymity posture, and reliance on external infrastructure. These risks are not addressed within the Engine itself.

Any mitigation occurs externally and does not modify PoL protocol or Engine behavior. The presence of mitigation patterns does not imply endorsement, standardization, or effectiveness.

12.1 Sybil Participation and Coordination Risk

Because participation is permissionless and anonymous, #PEG draws are exposed to Sybil participation, coordinated entry strategies, and aggregation behavior.

The protocol does not distinguish between individual and coordinated participants and does not detect or prevent Sybil behavior.

External mitigation patterns may include:
- entry caps imposed by intermediaries,
- participation heuristics applied by distributors,
- social coordination norms within specific access contexts.

Such measures operate entirely outside the Engine boundary and do not affect draw execution.

12.2 Draw Pool Integrity and Interface Risk

Participants can interact with the #PEG engine through interfaces or intermediaries that collect allocation commitments and submit entries on their behalf.

These arrangements introduce risks related to:
- misrepresentation of draw parameters,
- diversion or misallocation of allocation commitments,
- opaque aggregation practices,
- custody exposure between instantiation and execution.

External mitigation patterns may include:
- interface transparency and disclosure,
- independent verification of contract addresses,
- open-source front-end code,
- community reputation or audit practices.

The Engine does not validate interfaces, certify draw authenticity, or monitor custody arrangements.

12.3 Randomness and Data Source Dependence

The #PEG engine relies on external randomness providers and, where applicable, external data sources referenced at execution time.

Failure, manipulation, delay, or withdrawal of these services may prevent draw execution or affect outcome determination.

The #PEG engine eliminates the risk of persistent locked states by enforcing finite draw lifecycles and mandatory register clearance. As a consequence, under strict execution semantics, failure to execute may result in irreversible loss of allocation commitments if deterministic release cannot be performed. This risk reflects a deliberate tradeoff between preventing indefinite value accumulation within the Engine and tolerating bounded loss events at draw termination.

The Engine does not adjudicate randomness quality beyond verifiability at execution and does not substitute failed inputs.

12.4 Liquidity and Conversion Risk

Denominated outcomes produced by #PEG draws may be subject to liquidity constraints, conversion spreads, or market fragmentation when exchanged or valued externally.

Such risks arise from:
- limited market depth,
- jurisdictional constraints,
- intermediary practices,
- or participant expectations.

External mitigation patterns may include:
- market-making activity,
- arbitrage,
- bilateral conversion arrangements,
- acceptance of denomination risk by participants.

The #PEG engine does not coordinate liquidity provision or support secondary markets.

12.5 User Experience and Disclosure Risk

Participation in #PEG draws involves probabilistic outcomes and irreversible execution. Participants may underestimate risk, misunderstand draw parameters, or misinterpret denomination.

External mitigation patterns may include:
- explicit disclosure of draw parameters,
- probabilistic framing,
- educational material,
- conservative interface design.

The #PEG engine provides no warnings, guidance, suitability checks, or protective disclosures.

12.6 Regulatory and Narrative Exposure

The #PEG engine's anonymity posture, lack of governance, and refusal of redemption guarantees may attract regulatory scrutiny or reputational challenge.

Interpretations of legality, compliance obligations, or consumer protection vary by jurisdiction and may change over time.
External mitigation patterns may include:
- jurisdictional filtering by intermediaries,
- legal disclaimers,
- selective access restriction,
- narrative framing by participants or commentators.
The Engine does not adapt to regulatory environments or enforce compliance.

13. CONCLUSION

This document has specified a monetary engine whose operation is limited to the execution of irrevocable, draw-based allocations denominated in reference units and constrained by verifiable

randomness. The specification is complete at the level it defines: the Proof of Luck protocol, the #PEG engine, and the boundary separating both from the System in which their outputs are interpreted.

The #PEG engine has been presented neither as a solution to monetary instability nor as a proposal for reform. It does not claim to correct existing systems, compete with them, or replace them. Its ambition is narrower and, in a sense, more radical: to demonstrate that money can be reduced to form without collapsing into function, and that allocation can occur without justification, policy, or entitlement.

What emerges from this reduction is not an answer to the question of what money should be, but a clarification of what money can be when its usual predicates are suspended. By formalizing denomination without valuation, allocation without merit, and execution without governance, the #PEG engine exposes a layer of monetary reality that is ordinarily obscured by economic purpose. It produces monetary facts without explaining them, outcomes without defending them, and units without telling us what they are worth.

This abstention is deliberate. The Engine does not deny that meaning, value, or stability may arise around its outputs. On the contrary, it assumes that such meanings will emerge—socially, culturally, and economically—once the Engine is embedded in a broader system. But it refuses to anticipate, encode, or manage those meanings. In doing so, it draws a sharp boundary between what can be specified and what must remain contingent.

The result is a monetary artifact that is incomplete by design. It provides denomination, allocation, and finality, and then stops. Everything that usually follows—exchange, valuation, institutionalization, belief—is left to the world beyond the Engine. This incompleteness is not a limitation to be overcome, but the condition under which a different kind of monetary experimentation becomes possible.

Whether such a system is useful, adoptable, or durable is not a question this document attempts to answer. Its purpose has been more modest and more demanding: to articulate a monetary structure that refuses to justify itself, and to invite others to consider what forms of monetary life might arise when justification is no longer embedded at the point of origin.

In that sense, the #PEG engine is less a blueprint than a boundary object. It marks a limit—of specification, of governance, of theory—and asks what happens when money is allowed to begin there.

NORMATIVE DEFINITIONS AND COMPLETION CLARIFICATIONS

This subsection defines a minimal set of normative terms and conditions required to complete the specification of the #PEG engine. These definitions do not introduce new functionality, objectives, or guarantees. They exist solely to remove ambiguity where execution semantics depend on external inputs or structural representations.

N1. Publicly Verifiable Randomness (Normative Minimum)

For the purposes of the Proof of Luck (PoL) protocol, a randomness source is publicly verifiable if and only if:

Unpredictability prior to revelation
The random value cannot be known or reliably inferred by any participant or observer prior to its disclosure.

Post hoc verifiability
Any observer can independently verify, after the fact and without privileged access, that:
- the random value was produced according to the declared randomness mechanism, and
- the value consumed by the Engine corresponds exactly to the value produced.

Non-equivocation
The randomness source provides a single, non-ambiguous output for a given execution context. Multiple competing values for the same execution context are treated as invalid.

At execution time, randomness is considered available if a value satisfying the above conditions is supplied to the execution environment in the form required by the deployed draw contract. Randomness is considered unavailable or invalid if no such value can be verified at the scheduled execution time.

The #PEG engine does not assess randomness quality beyond these minimum properties and does not substitute, retry, or defer execution in the event of randomness unavailability.

N2. Allocation Commitment (Engine-Level Representation)

An allocation commitment is an Engine-internal representation of a prospective #PEG-denominated allocation submitted prior to draw execution.

An allocation commitment:
- is recorded in the draw register,
- is non-transferable prior to execution,
- does not constitute custody of a spendable balance,
- and confers no entitlement, claim, or guarantee of allocation.

Allocation commitments become effective allocations only upon execution, at which point they are either:

- consumed by allocation settlement (executed draw), or
- deterministically reversed or destroyed (cancelled draw).

At no point does the #PEG engine custody transferable #PEG balances prior to execution. The draw register therefore cannot be looted, drained, or redistributed independently of execution.

N3. Deterministic Release and Release Failure Conditions

Upon draw cancellation, the Engine attempts a deterministic release of all recorded allocation commitments.

A deterministic release is possible if and only if:
- each commitment recorded in the draw register can be uniquely and unambiguously mapped to a submission wallet address, and
- the Engine can perform a mechanical reversal of the prior commitment recording without requiring external input, discretion, or interpretation.

Deterministic release fails if any of the above conditions are not met, including but not limited to:
- aggregation modes that do not preserve a one-to-one mapping between commitments and submission wallets,
- incomplete or incompatible provenance data for commitments,
- structural constraints of the deployed draw contract that prevent reversal at terminal resolution.

Release failure is a structural condition, not an error state.

N4. Burn Semantics (Terminal Enforcement)

If deterministic release cannot be performed, the Engine must burn the unreleasable commitments.

Burning:
- is non-discretionary,
- is terminal,
- and exists solely to enforce finite draw lifecycles and register clearance.

Burning does not constitute punishment, slashing, or economic deterrence. It reflects a deliberate design tradeoff between preventing indefinite value accumulation within the Engine and tolerating bounded loss events at draw termination.

N5. Scheduled Execution Time and Execution Windows

Unless explicitly specified otherwise by a given draw type, a #PEG draw executes at a single scheduled execution time, which constitutes a hard, non-deferable boundary.

If a draw type supports an execution window, the following rules apply:
- The window defines the temporal interval during which a single execution attempt may occur.

- The first admissible execution opportunity within the window is treated as the scheduled execution time.
- Failure to execute at that opportunity results in immediate cancellation.

No retry, postponement, or second attempt occurs within the window.

Execution windows therefore do not introduce liveness guarantees or retry semantics.

N6. Entry Acceptance and Restriction Enforcement

At the Engine level, participation in #PEG draws is permissionless by default.

Entry restriction may occur only through:
- Engine-level restriction, where the deployed draw contract includes an explicit admissibility function evaluated prior to entry recording; or
- System-level restriction, where access is gated by external interfaces, intermediaries, or coordination mechanisms that do not modify Engine semantics.

System-level restriction does not alter the permissionless nature of the Engine itself and has no effect on execution logic once a valid entry is recorded.

N7. Economic Griefing and Cost Amplification

The #PEG engine does not attempt to prevent or mitigate strategies that intentionally increase execution costs, contention, or cancellation probability.

Such strategies, where present, constitute system-level economic griefing risks and are external to the Engine's scope. The Engine neither detects nor responds to such behavior and does not adapt execution rules in response.

N8. Observational References

Any references to stochastic properties (e.g., Markovian behavior) describe possible ex post observational patterns arising from repeated draw execution and participation. No stochastic model, convergence property, or probabilistic guarantee is specified or assumed by the Engine.