Lux Credit Protocol Specification: Omnichain Collateralized Credit Issuance, ~~Transport, and Settlement~~

Zach Kelling

Lux Industries

2024

Lux Credit Protocol Specification: Omnichain Collateralized Credit Issuance, Transport, and Settlement

Zach Kelling

2024

Abstract

We present the canonical specification for the Lux Credit Protocol, an omnichain system for collateralized credit issuance, cross-chain transport, and atomic settlement. The protocol issues overcollateralized, in-kind redeemable credit tokens (e.g., LETH backed by ETH) subject to hard loan-to-value constraints, with strategy yield accruing to protocol surplus rather than to credit holders. This document establishes the formal asset taxonomy, canonical message schema, system invariants, solvency state machine, failure behavior, and trust assumptions. It serves as the authoritative reference for auditors, counterparties, and implementors. The protocol decomposes into four auditable subsystems—collateral custody, credit issuance, transport/finality, and surplus allocation—each subject to distinct failure classes and verification requirements.

1 Introduction

Cross-chain liquidity systems occupy a contested design space between bridges, lending protocols, and synthetic asset platforms. Each framing carries implicit assumptions about user rights, failure modes, and regulatory posture that, if left undefined, produce semantic leakage: code that compiles and deploys but whose economic and legal meaning drifts across contexts.

The Lux Credit Protocol resolves this by committing to a single, precise classification: it is an *omnichain collateralized credit and settlement protocol*. It is not a bridge. It is not a vault. It is not a stablecoin. It is not a liquid staking token. Each of these labels implies properties the system does not possess or obligations it does not accept.

Under this framing, the protocol comprises exactly four subsystems:

1. **Collateral custody.** Vault-based asset custody with strategy deployment and cryptographic backing attestation.
2. **Credit issuance.** Minting of overcollateralized, in-kind redeemable credit tokens subject to hard LTV caps.
3. **Transport and finality.** Omnichain message relay with per-chain finality thresholds and threshold signature verification.
4. **Surplus allocation.** Protocol equity accrual from strategy yield, structurally separated from credit holder claims.

Each subsystem is audited against different failure classes. This decomposition determines which invariants attach to which code paths, which failure modes trigger which recovery procedures, and which guarantees are offered to which participants.

Scope. This document freezes the protocol semantics. It defines terminology, asset classes, message formats, invariants, the solvency state machine, failure behavior, trust assumptions, and the boundary conditions with the ATS compliance layer. Line-by-line router review proceeds from this specification, not the other way around.

2 Asset Taxonomy

Three mutually exclusive asset classes exist within the protocol. Conflating classes is a protocol violation enforceable at both the contract and legal layers.

2.1 Class 1: Credit Assets

Definition 2.1 (Credit Asset)**.** *A credit asset is an overcollateralized, in-kind redeemable liability of the protocol, minted against vault-backed base assets subject to a hard LTV cap.*

Instances. LETH (backed by ETH), LBTC (backed by BTC).

Requirement 2.1 (Credit Asset Properties)**.** *A credit asset* MUST *satisfy all of the following:*

1. *Minted only against recognized, attested backing.*
2. *Redeemable 1:1 in kind:* $\texttt{burn}(x) \Rightarrow \texttt{release}(x)$ *of the underlying base asset.*
3. *Hard issuance cap at or below the configured LTV ratio.*
4. *No direct claim on strategy yield.*
5. *No governance, profit-share, or security rights.*

Invariant 2.1 (Class Separation)**.**

$$\texttt{creditAsset} \neq \texttt{yieldShare} \neq \texttt{securityToken}$$

Enforced technically and legally via separate token contracts, separate registries, separate accounting buckets, and separate disclosures. A single contract or accounting path MUST NOT *conflate two classes.*

Canonical phrasing. For regulatory filings, disclosures, and counterparty agreements:

> *LETH is a protocol-issued, fully collateralized, in-kind redeemable credit token backed by ETH vault assets and minted subject to a hard maximum loan-to-value ratio of 90%, with strategy yield accruing to protocol surplus rather than to LETH holders.*

2.2 Class 2: Yield and Equity Instruments

Definition 2.2 (Yield/Equity Instrument)**.** *A yield/equity instrument represents a claim on protocol surplus, not on redemption principal backing.*

Instances. xLUX, protocol equity accounting unit.

Requirement 2.2 (Yield Instrument Properties)**.**

1. *Absorbs upside from strategy earnings.*
2. *May absorb losses after buffer depletion, depending on instrument design.*
3. *Never used as collateral equivalence for credit redemptions unless explicitly approved by governance action.*

The structural separation between credit assets and yield instruments is the primary mechanism by which the protocol avoids characterization as a security or investment contract with respect to credit holders.

2.3 Class 3: Security Assets

Definition 2.3 (Security Asset)**.** *A security asset is issued and controlled under the ATS/compliance layer and is not fungible with protocol credit liabilities.*

Instances. Tokenized equity, fund interests, real-world assets.

Requirement 2.3 (Security Asset Properties)**.**

1. *Compliance-enforced transferability (whitelist, lockup, KYC).*
2. *Jurisdictional controls per offering type.*
3. *Separate wrapping policy when bridged into Lux (Section 7).*

3 Canonical Bridge Message Schema

The message layer is frozen at this specification version. No additional message types may be introduced without a protocol upgrade.

3.1 Message Types

Type	Semantics
`DEPOSIT_V1`	Lock base asset; mint credit on destination
`BURN_V1`	Burn credit token on source chain
`RELEASE_V1`	Release base asset on destination chain
`BACKING_V1`	Backing attestation from vault oracle

3.2 Common Envelope

Every message MUST include the following 13 fields:

Field	Type	Description
`version`	`uint8`	Schema version
`messageType`	`uint8`	Enum (4 types)
`srcChainId`	`uint256`	Source chain
`dstChainId`	`uint256`	Destination chain
`routerAddress`	`address`	Sending router
`assetId`	`bytes32`	Canonical asset ID
`nonce`	`uint256`	Per-router sequence
`amount`	`uint256`	Transfer amount (wei)
`recipient`	`bytes32`	Destination address
`expiry`	`uint256`	Expiration timestamp
`originTxHash`	`bytes32`	Source event hash
`payloadHash`	`bytes32`	Typed payload hash
`domainSeparator`	`bytes32`	Deployment domain

The `domainSeparator` is computed as:

$$\texttt{keccak256}(\texttt{abi.encode}(\mathcal{N},\ \mathcal{V},\ c,\ r)) \qquad (1)$$

where $\mathcal{N}$ is the protocol name, $\mathcal{V}$ the protocol version, c the chain ID, and r the router address. This closes five replay classes: chain replay, deployment replay, migration replay, same-signature context reuse, and duplicated source event processing.

3.3 Receive-Path Verification

Requirement 3.1 (Minimum Verification Rules)**.** *Every receive path MUST verify all of the following before executing any state change:*

1. *`block.timestamp` $\leq$ `expiry`*
2. *`dstChainId` $=$ `block.chainid`*
3. *`routerAddress` $=$ `address(this)`*
4. *`nonce` unused for (`srcChainId`, `routerAddress`, `assetId`)*
5. *`amount` > 0*
6. *`recipient` $\neq 0$*
7. *`assetId` recognized and enabled*
8. *Message hash signed by $\geq t$ of n authorized signers*

4 System Invariants

Let B denote total backing value, L total outstanding credit liabilities, and $E = B - L$ protocol equity. All values are denominated in the base asset's native unit.

4.1 Core Issuance Invariant

Invariant 4.1 (Issuance Cap)**.**

$$L \leq \lambda \cdot B \qquad (2)$$

where $\lambda =$ `issuanceLtvCap`. For LETH with $\lambda = 0.9$: $L \leq 0.9 \cdot B$. Checked on every mint; reverts on violation.

4.2 Redemption Invariant

Invariant 4.2 (In-Kind Redemption)**.**

$$\texttt{burn}(x\ LETH) \Rightarrow \texttt{release}(x\ ETH) \qquad (3)$$

subject only to finality delay (Section 6.4), solvency state constraints (Section 5), and applicable compliance requirements.

4.3 Parity Invariant

Invariant 4.3 (Unit Parity)**.** *Redeemable principal per unit of credit token equals exactly one unit of the base asset at all times, independent of surplus level or strategy performance.*

4.4 Equity Threshold

Invariant 4.4 (Minimum Equity Ratio)**.** *Under maximum LTV λ:*

$$\frac{E}{B} \geq 1 - \lambda \quad \Leftrightarrow \quad \frac{L}{B} \leq \lambda \qquad (4)$$

For $\lambda = 0.9$: equity ratio $\geq 10\%$, leverage ratio $\leq 90\%$.

5 Solvency State Machine

The protocol defines exactly four solvency states. All transitions are deterministic given B and L. No transition requires operator discretion.

5.1 States

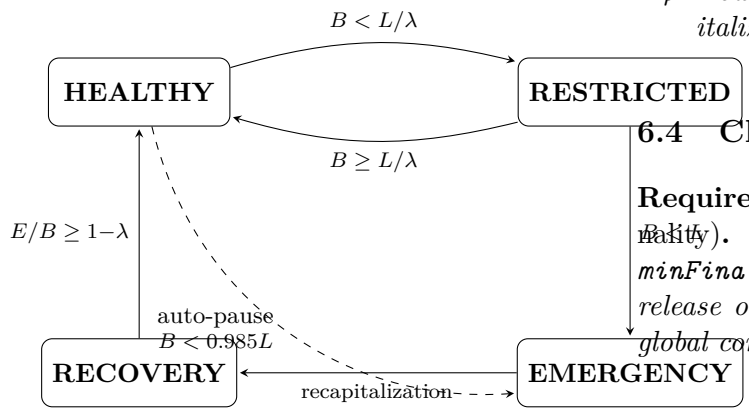
Definition 5.1 (Healthy)**.** *$B \geq L/\lambda$ and $B \geq L$. All operations permitted.*

Definition 5.2 (Restricted Mint)**.** *$B < L/\lambda$ but $B \geq L$. Equity buffer depleted; principal still covered. Mint **disabled**. Burn and release allowed. Strategy deployment halted; withdrawals initiated.*

Definition 5.3 (Emergency)**.** *$B < L$. System undercollateralized on principal basis. Mint **disabled**. Burn allowed. Release governed by emergency policy. Strategy withdrawals **forced**. Recovery governance **active**.*

Definition 5.4 (Recovery)**.** *Post-emergency transitional state. Mint disabled until $E/B \geq 1 - \lambda$. Burn and release allowed. Exit requires governance or MPC quorum.*

5.2 Transition Diagram



5.3 Auto-Pause Trigger

An accelerated emergency trigger fires when $B < \alpha \cdot L$ with $\alpha = 0.985$. This immediately halts all outbound operations except burns.

6 Failure Behavior

6.1 MPC Liveness Failure

Requirement 6.1 (MPC Halt Policy)**.** *The protocol MUST select exactly one mode:*

Mode A: MPC-gated releases. *Mint disabled. Release queue accumulates. No liveness guarantee until $\geq t$ signers recover.*

Mode B: User-executable fallback. *After timeout T, users self-relay with burn proof from source chain. Recommended for institutional deployments.*

6.2 Stale Backing Report

Requirement 6.2 (Backing Staleness)**.** *Define $\tau_{\max}$ = `maxBackingReportAge`[`assetId`]. If no valid `BACKING_V1` received within $\tau_{\max}$: mint disabled, burn/release allowed against last confirmed backing, emit `BackingStale` event.*

6.3 Strategy Loss Waterfall

Requirement 6.3 (Loss Absorption)**.**

1. *Loss reduces surplus E.*
2. *$E \leq 0$: Restricted Mint.*
3. *$B < L$: Emergency.*
4. *New strategy deployments halted until recapitalization.*

6.4 Chain Reorg and Finality

Requirement 6.4 (Per-Chain Finality)**.** *For each chain c: f_c = `minFinalityConfirmations`[c]. Mint and release only after $\geq f_c$ confirmations. A single global constant is **prohibited**.*

6.5 Oracle and Valuation Failure

Requirement 6.5 (Oracle Failure)**.** *Freeze mint for affected asset. Allow only like-for-like redemption. Disable cross-asset accounting on stale prices.*

7 ATS to Lux Boundary

Requirement 7.1 (Wrapping Rule)**.** *A security token may be transported into Lux **only** through a wrapper whose compliance posture is explicitly declared and enforced at the wrapper level.*

Model 1: Strict wrapper. ATS Security → Restricted Wrapped Security. Whitelist, restrictions, lockups preserved. Recommended.
Model 2: Dual wrapper. Adds a second composable representation after explicit compliance transformation. Higher flexibility; more legal ambiguity. Deferred.

8 State Transition Reference

9 Trust Assumptions

1. **MPC signer set.** At least t of n signers honest and available. Liveness degrades; safety holds unless $\geq n - t + 1$ collude.
2. **Vault custody.** Strategy contracts lack unilateral withdrawal authority exceeding configured limits.
3. **Backing oracle.** Attestations delivered within $\tau_{\max}$ and reflect true vault state.
4. **Chain finality.** Source transactions final after f_c confirmations. Deeper reorgs outside threat model.
5. **ATS compliance.** Wrapper contracts enforce transfer restrictions; the protocol layer does not.

10 Open Questions for Audit

1. **Stale-backing redemptions.** Do exits remain permissionless during stale backing? *Recommendation:* stop mint, preserve exits.
2. **Emergency release policy.** Halt releases, pro-rata, or FIFO queue? *Recommendation:* FIFO queue with governance-controlled cadence.

These are core user-rights definitions, not implementation details.

11 Implementation Sequencing

1. Freeze this specification.
2. Freeze the message schema.
3. Freeze the solvency state machine.
4. Map invariants to contract-level revert conditions.
5. Deployment review and audit.

Specification before implementation review: line-by-line analysis is more valuable when semantics are fixed.

Acknowledgments

This specification incorporates structured review feedback from protocol engineers, compliance counsel, and independent security auditors.

Table 1: Allowed operations per solvency state.

Operation	**Healthy**	**Restricted**	**Emergency**	**Recovery**
Mint credit	Yes	No	No	No
Burn credit	Yes	Yes	Yes	Yes
Release backing	Yes	Yes	Policy	Yes
Strategy deposit	Yes	No	No	No
Strategy withdrawal	Optional	Initiated	Forced	No
ATS security wrapping	Yes	Yes	No	No
Backing report required	Yes	Yes	Yes	Yes

Table 2: State transition conditions and authorized actors.

From	**To**	**Condition**	**Actor**
Healthy	Restricted	$B < L/\lambda$	Automatic
Restricted	Healthy	$B \geq L/\lambda$	Automatic
Restricted	Emergency	$B < L$	Automatic
Any	Emergency	$B < 0.985 \cdot L$	Automatic (auto-pause)
Emergency	Recovery	Recapitalization	Governance / MPC
Recovery	Healthy	$E/B \geq 1 - \lambda$	Governance / MPC