

Mutual Credit on Verifiable Trust Infrastructure

Oct 18, 2026 · @Geoff Turk

Summary

A Verifiable Trust Community (VTC) already holds the hard part of a mutual-credit system: a membership roll whose standing is verifiable in real time, policies that gate what members may do, and keys that every member controls. What is missing is a ledger. This note proposes adding one as a **credit-clearing module**: a double-entry ledger per community where every entry is a Trust Task signed by both counterparties, debit limits are set by policy from membership tier and standing, and balances net to zero by construction because credit is issued only at the moment of a trade.

The design has three layers, each useful on its own:

1. **One VTC, one clearing circle.** Members trade; each trade is a signed, countersigned entry that credits one account and debits the other. Members may go negative up to a limit the community's policy derives from their tier and standing. The sum of all balances is always zero.
2. **A network of VTCs (a VTN).** Each community keeps its own books. A member of community A paying a member of community B produces a mirrored pair of entries plus an inter-community balance between A and B, bounded by limits the two communities set for each other. Periodic netting across the network keeps inter-community exposure small.
3. **Signed entries as contracts.** Every entry carries the terms it was made under — the unit, the community's rules document, the limits in force — so the entry is also the evidence. This is Grigg's triple-entry idea: the signed receipt *is* the transaction, held by payer, payee and the community.

The proposal follows the rules Thomas Greco distills from a century of credit-clearing practice: credit is created by members, not a bank; limits are tied to a member's demonstrated capacity to earn credits back; and the system's integrity rests on the community knowing who its members are, which is exactly what a VTC is for.

What this note deliberately leaves to each community, or each network of communities, to decide for itself: the unit of account, how limits are first set for a member with no history, who may see a balance, whether a VTC may hold a balance of its own, and how

inter-community exposure is governed. The tool's job is to make each of those choices expressible in a community's rules document and enforceable by its policy, never to make them; the list at the end records what must be left open.

What the stack already gives us

Five of the six things a clearing circle needs exist in the `verifiable-trust-infrastructure` workspace today; the sixth, the ledger itself, does not exist in any form, and neither does voting. The table is from a read of the repo at [OpenVTC/verifiable-trust-infrastructure](https://github.com/OpenVTC/verifiable-trust-infrastructure) on 18 Oct 2026.

Need	What exists	Where	Gap for clearing
Who is a member, right now	<code>Member</code> records with VMC and role VAC; suspension and revocation on Bitstring status lists; <code>MembershipSyncer</code> publishes recognition records over TRQP v2	<code>vtc-service/src/members</code> , <code>status_list</code> , <code>registry</code> ; <code>docs/03-vtc/trust-registry.md</code>	none for standing; a <code>clearing</code> action string on registry records would follow the <code>git.commit.sign</code> precedent in <code>docs/03-vtc/git-namespaces.md</code>
Rules the community can change	one active Rego policy per <code>PolicyPurpose</code> ; the ceremony pipeline gives a policy <code>actor</code> , <code>subject</code> , <code>context</code> , <code>evidence</code> , <code>state</code> and takes <code>allow</code> , <code>deny</code> , <code>refer</code> , <code>request_more</code>	<code>vtc-service/src/policy</code> , <code>ceremony</code> ; <code>docs/05-design-notes/vtc-ceremony-*.md</code>	a new purpose (limit evaluation) and <code>state</code> facts from the ledger (balance, turnover, account age)
A signed request and response between two parties	Trust Tasks: <code>eddsa-jcs-2022</code> proof per envelope, <code>threadId</code> , replay records, 10-minute freshness; generic ceremony verbs include <code>accept</code> meaning counter-sign	<code>vta-sdk/src/trust_tasks.rs</code> ; <code>docs/05-design-notes/vtc-ceremony-protocol.md</code>	a <code>clearing/*</code> task family, specified upstream first (see holes)
Tamper-evident history	VTC audit log: hash-chained envelopes with signed checkpoints every 15 minutes; witnesses that issue VWCs; witnessed <code>did:webvh</code> logs	<code>vti-common/src/audit</code> ; <code>docs/05-design-notes/vtc-audit-checkpoints.md</code> ; <code>vti-witness</code>	the ledger log can reuse the audit envelope and checkpoint code directly
Credentials a member holds and presents	DTG v1 VCs, BBS+ and SD-JWT selective disclosure, OID4VCI/OID4VP wrapped in <code>credential-exchange/*</code> tasks, holder vault with status refresh	<code>vtc-service/src/credentials</code> , <code>vta-vault</code>	two new credential types (clearing standing, balance attestation)

Need	What exists	Where	Gap for clearing
Shared data between members	data rooms: a versioned key-value store with MLS encryption tiers, Merkle commitments, community-owned rooms	<code>vti-rooms*</code> , <code>room-host</code> , <code>docs/05-design-notes/data-rooms.md</code>	rooms are not append-only and have no policy engine, so they hold the rules document and statements, not the books
Cross-community trust	<code>auth/recognise</code> verifies a foreign VMC and VAC through TRQP, then <code>cross_community_roles.rego</code> maps the role (deny by default)	<code>vtc-service/src/recognition</code>	a VTN is mentioned only in passing; no federation design note exists yet
Two-party signatures	VRC edges are two credentials, one each way; VMC acknowledgements carry a digest of the grant; task-consent decisions are threshold-signed over a payload digest	<code>docs/03-vtc/personhood-and-graph.md</code> ; <code>docs/02-vta/task-consent.md</code>	the countersigned entry is the same shape as a VMC acknowledgement over a digest

Three facts from the survey shape the design more than the rest:

- 1. Rooms are the wrong home for the books.** The data-room design rejects CRDTs and append-only logs in favour of a single write-primary with version preconditions, and a room host keeps no member list and runs no policy. A ledger needs an ordered log, a policy check on every write and an invariant over all accounts. It belongs beside the audit log inside `vtc-service`, as a capability module like git namespaces, with rooms used for what they are good at.
- 2. The spec comes first.** `CLAUDE.md` is explicit that the VTI specification is normative and that Trust Task payload types are generated from `dtgwg-trust-tasks-tf`; local schemas cannot add fields. A `clearing` task family therefore starts with its specs and schemas, written alongside the code. That is a step you already know how to take quickly, not a blocker.
- 3. Standing is already an external query.** Because membership is published over TRQP and `auth/recognise` already verifies foreign credentials, the network-level question, is Bob's community in good standing and does it stand behind Bob, needs no new trust machinery; it needs a limit.

Mutual credit inside one VTC

A clearing circle needs six rules, and a VTC can enforce each one at the moment a trade is signed rather than after the fact. The rules below are those Greco distills in [Credit Clearing: Pure and Simple](#) and his books; the numbers given as rules of thumb are his

approximate guidance, not sourced figures.

Rule	What it means	Where the VTC enforces it
Members only, and both ways	Every account holder is a buyer and a seller; the exchange is an association, not a bank	A ledger account exists only for a DID holding a valid membership credential; a trade is refused unless both parties are in good standing at signing time
Credit is created at the trade	Nothing is pre-funded; a purchase debits the buyer and credits the seller by the same amount, so all balances sum to zero	The ledger has no mint and no deposit operation; the only write is a countersigned two-sided entry
Debit limit tied to earning capacity	A member may go negative only up to a limit sized to what they demonstrably sell back into the circle. Greco's rule of thumb is a line of credit on the order of a few months of the member's average sales in the exchange; a new member starts small	Policy computes the limit from tier and trailing turnover (section on limits); the signing step refuses an entry that would breach it
Credit ceiling too	A large positive balance is credit withdrawn from circulation and a claim others must eventually honour	A maximum balance per account, also policy-set, with a nudge to spend rather than a hard stop
Balances are meant to return to zero	A debit that neither shrinks nor turns over signals a problem; a member leaving must first clear to zero	Stagnation and turnover are visible from the ledger; policy can lower a limit or flag an account for the community's review; leaving requires a zero balance or a community decision
The operator takes no credit for itself	Greco's central criticism of commercial barter exchanges is that the operator issues credit to itself and dilutes the unit	The community account, if one exists, is a normal account with a policy-set limit and a published balance; there is no privileged write path

The unit of account is a community choice. Most circles denominate in the national currency to keep prices legible (a Sardex credit is one euro, a WIR franc one Swiss franc); time-based units are the other common choice. The design below is unit-agnostic, but the unit must be named in every entry.

A trade, step by step

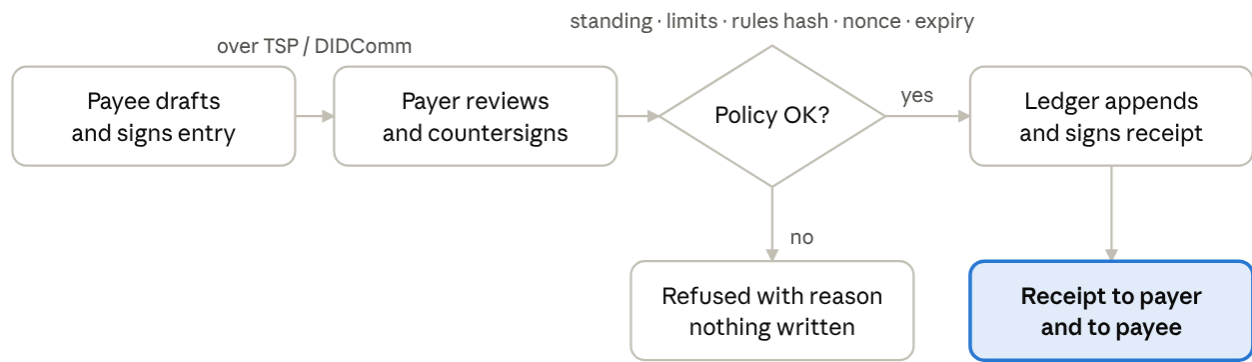
A trade is a short protocol between two members' agents (their VTAs, or the VTC acting for a member who delegates), with the VTC's ledger service as the third party that checks policy and records the result. The payee drafts the entry because the payee is the party extending credit; the payer countersigns because the payer is the party taking it on. Either order works, but the entry is not an entry until both signatures and the ledger's acceptance exist.

The steps:

1. Payee drafts an entry: payer DID, payee DID, amount, unit, memo, the ledger's current rules hash, a nonce, and a short expiry.
2. Payee signs it and sends it to the payer over TSP or DIDComm (the same transport every Trust Task already uses).
3. Payer reviews the terms and countersigns, producing a two-signature entry.
4. Either party submits the entry to the ledger service as a Trust Task.
5. The ledger checks policy: both DIDs currently in good standing (trust-registry query), payer balance minus amount at or above the payer's debit limit, payee balance plus amount at or below the payee's ceiling, rules hash current, nonce unseen, expiry not passed.
6. On success the ledger appends the entry, signs a receipt over it (entry hash, resulting balances, sequence number, previous-entry hash) and delivers the receipt to both parties. On failure it returns the specific reason and nothing is written.

A refused trade leaves no trace in the books, only in the audit log. That matters for privacy: a member who is at their limit should not leak that fact to every counterparty through a visible failed entry.

An entry exists only after two signatures and a policy check



trade flow · 4 steps, 1 policy decision

The two member agents agree first; the ledger only checks and records. A failed check returns the reason to the submitter and writes nothing to the books.

The ledger: signed entries as the record

The books are derived from the receipts, not the other way round. This is the core of Grigg's [triple entry accounting](#): a receipt that carries the payer's signed instruction and the issuer's signed response is a complete record of the event, auditable from the signatures alone, and the same receipt is held by payer, payee and issuer. Balances and histories can be rebuilt from the receipt set at any time; the ledger's running books are a cache for fast limit checks.

In VTI terms the roles map cleanly:

Grigg's role	VTI party	Holds
Alice (payer)	member's VTA	her copy of every receipt she is party to, in her vault
Bob (payee)	member's VTA	his copy of the same receipts
Ivan (issuer)	the VTC's ledger service	the append-only receipt log, the derived account books, the rules document

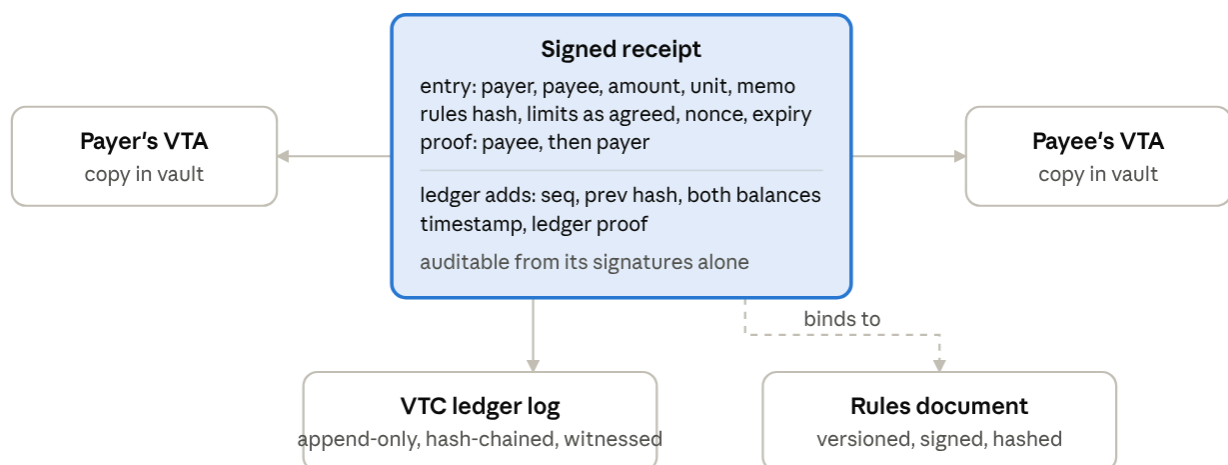
What an entry contains

The entry is a Trust Task payload, signed with Data Integrity proofs the stack already verifies. Fields:

- **payer** , **payee** : member DIDs. Pseudonymous personas are fine; standing is checked against the DID the membership credential binds to.
- **amount** , **unit** : integer minor units and the ledger's unit identifier, never a float.
- **memo** : free text, optionally encrypted to the two parties only.
- **rules** : the hash of the ledger's rules document in force when the entry was drafted. This is the Ricardian link: the terms are not restated in each entry, but each entry binds to exactly one version of them.
- **limits** : the payer's debit limit and the payee's ceiling as the drafter believed them, so a later dispute can show what both parties thought they were agreeing to.
- **nonce** , **expires** , **thread** : replay protection and the Trust Task thread the two signatures belong to.
- **proof[]** : two Data Integrity proofs, payee then payer.

The ledger's receipt wraps the entry and adds **seq** , **prev** (hash of the previous receipt), the two resulting balances, a timestamp and the ledger DID's own proof. The **prev** chain makes the log tamper-evident; a witness (section on components) can periodically sign the head so that even the ledger operator cannot rewrite history unnoticed.

One receipt, three holders, bound to one version of the rules



signed receipt · 3 holders, 1 rules version

Payer and payee keep the same receipt the ledger keeps; the rules hash inside it says which version of the community's terms both signed under.

Double entry, enforced by shape

There is one write operation and it always moves the same amount out of one account and into another. A **balances** table is maintained alongside the log, but a reconciliation job rebuilds it from the receipts and refuses to serve if the two disagree. The invariant

`sum(balances) = 0` is checked on every write. Fees, if the community charges any, are themselves entries whose payee is the community account, so they appear in the same books under the same rules.

Corrections are new entries, never edits: a mistaken trade is reversed by a countersigned entry in the other direction that references the original. The log never shrinks.

The rules document

The Ricardian contract here is the community's clearing rules: the unit and its definition, limit policy, fees, what happens on exit, dispute procedure, and governance (how the rules change). It is a signed document published by the VTC, versioned, with each version hashed; the policy bundle that enforces the rules references the same hash. A member joining the ledger countersigns the current version once, and that signature is itself a receipt.

Credit limits and standing

A limit is a policy output, not a stored number. Each time an entry is checked, the ledger asks the community's policy engine for the payer's current debit limit and the payee's ceiling, and the policy derives them from facts the VTC already holds plus facts the ledger holds. That keeps the rule in one place (the Rego bundle), lets the community change it without migrating accounts, and makes every limit explainable: the receipt records which rules hash produced it.

Inputs a limit policy sees:

Input	Source	Why it matters
Membership status	trust registry (is this DID a member in good standing right now)	no standing, no account; suspended members can receive but not spend, or neither, as the rules say
Tier or role	membership credential claims	a community may grant higher starting lines to vetted or long-standing members
Trailing turnover	ledger: credits earned over the last N months	Greco's rule: the line tracks what the member demonstrably sells back into the circle
Account age	ledger: first receipt date	new accounts start small whatever their tier

Input	Source	Why it matters
Guarantees	optional credentials from other members or a group	a member can vouch for another's line up to a bounded amount, Greco's way of bootstrapping a newcomer
Community parameters	rules document	base line, multiplier on turnover, hard cap, ceiling

A plausible first policy, written in words rather than Rego: base line by tier (say three tiers), plus a multiple of the last ninety days of credits earned, capped by tier; ceiling equal to some multiple of the line; a member under review has both frozen at their current balance. The numbers belong in the rules document, not the code.

Standing as a verifiable claim

Two things are worth making portable as credentials rather than leaving as rows in the ledger:

- **A clearing-standing credential:** issued by the VTC to a member, stating that the holder has an account in good standing, their tier, and optionally a bracket for their limit (not the balance). It is what another community checks before extending cross-community credit, and what a member presents to prove they can be paid.
- **A balance attestation:** a short-lived, member-requested statement of the member's own balance and limit at a point in time, signed by the ledger, for the member to present where they choose. Balances are private by default; the attestation is how a member discloses one deliberately.

Both revoke the way the stack already revokes: standing follows the trust registry, so a suspension in the community propagates to the credential's validity without a separate revocation mechanism.

Clearing across groups

An inter-community ledger is the same ledger with communities as its members. That is the one idea that makes the network level tractable, and it is the idea behind the [Credit Commons Protocol](#): ledgers nest, every ledger has an account pointing at the rest of the world, and balances sum to zero at every level. A VTN clearing ledger has one account per VTC; each VTC's ledger has one account for the VTN. Nothing new is invented for the network except the limits the communities set on one another.

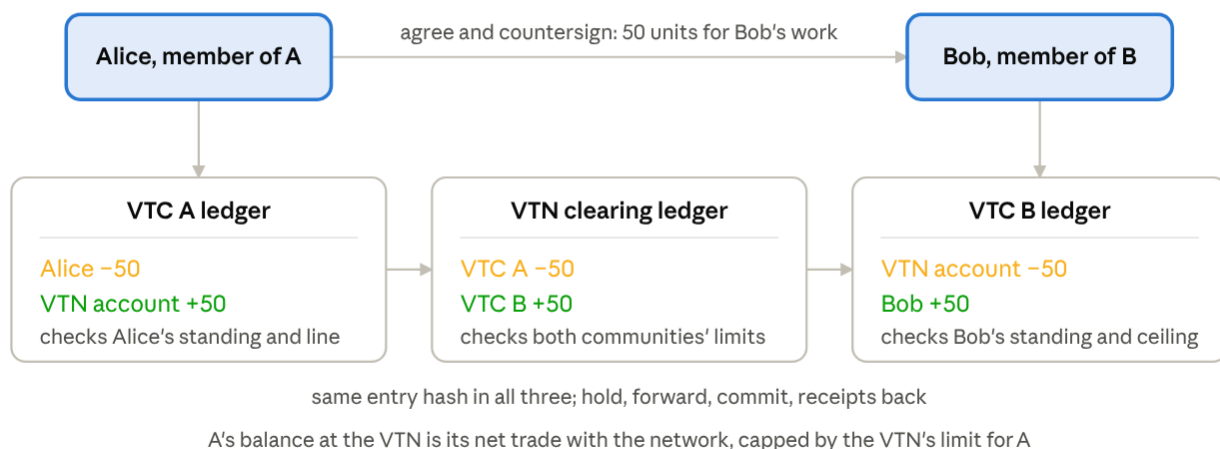
A payment from Alice in VTC A to Bob in VTC B

Alice and Bob agree and countersign exactly as they would inside one community; the entry just names Bob's community as well as Bob. Recording it takes three mirrored entries, one per ledger, all carrying the same entry hash:

1. VTC A's ledger: Alice -50, VTN account +50. A checks Alice's standing and limit as usual, and that A's own balance at the VTN has room.
2. The VTN clearing ledger: VTC A -50, VTC B +50. The VTN checks both communities are members in good standing and that the move stays inside the limits A and B each have there.
3. VTC B's ledger: VTN account -50, Bob +50. B checks Bob's standing and ceiling.

Each ledger still sums to zero. A's running balance at the VTN is its members' net trade with the rest of the network, and the VTN's limit on that account is the network's exposure to A. Inside A, the VTN account is just another account with a policy-set limit, which is how A caps how far its members can collectively go into debit against outsiders.

A cross-community payment is three mirrored entries; every ledger still sums to zero



cross-community payment · 3 ledgers, 1 entry hash

No ledger sees past its neighbour: A's books name Alice and the VTN, the VTN's name only communities, B's name the VTN and Bob.

The three writes must land together or not at all. The simplest safe mechanism is a hold-and-commit over the Trust Task thread: A's ledger places a time-limited hold on Alice's line and forwards; the VTN does the same for A and forwards; B accepts and commits; commit receipts flow back and each hold becomes an entry. Any hold that expires unconfirmed is released, with nothing written to the books. The same thread id ties the three receipts, so an auditor at any ledger can ask the others for their halves.

Who vouches for whom

A's ledger never sees Bob's balance and B's never sees Alice's. What crosses the boundary is standing, not books: Bob presents his clearing-standing credential from B, and A's ledger checks two things, that B is a VTN member in good standing (trust-registry query against the VTN) and that B's credential for Bob verifies and is unrevoked. If Bob later defaults, that is B's problem to govern; the VTN's claim is against B's account, and B's rules decide what happens to Bob. Liability follows the nesting, which is why a community should think of its VTN limit as the amount it is prepared to stand behind for its members.

Bilateral first, hub when it pays

Two communities can clear without any VTN entity at all: A opens an account for B in its ledger and B one for A, and the two accounts are kept mirror images through the same hold-and-commit. That is a two-member clearing ledger with no separate operator and is the right shape for a pilot. A hub (a VTN ledger run by one of the communities, by a VTSP, or by a jointly governed VTC whose members are communities) pays off once more than a handful of communities trade, because it turns n -squared bilateral limits into n accounts and nets multilaterally: A's surplus with B offsets its deficit with C automatically. Entries are identical in both shapes, so moving from bilateral to hub is a governance change, not a migration.

Rates, imbalances, and what not to automate

Where two communities use different units, the rate is a parameter of the inter-ledger rules document, fixed and reviewed by governance rather than floating. Persistent imbalance between communities is, as the Credit Commons authors say, a political matter: a surplus community can lend, invest, or spend its surplus back, and the right response to a structural deficit is to lower the limit, not to adjust prices behind members' backs. The protocol's job is to make the position visible and the limit enforceable; the decision stays with the communities' own voting, which the VTC will also host.

Mapping onto VTI components

The ledger is one new capability module in `vtc-service`, plus a task family, a policy purpose, two credential types and two UI surfaces. Everything else is reuse.

In the member's VTA. The VTA signs entries with the member's DID key, stores every receipt and the clearing-standing credential in `vta-vault`, and refreshes standing the way it refreshes any status list. `vta-policy` can require the existing N-of-M consent flow for entries above a member-set threshold, which is how a member's agent gets to pay small amounts unattended and large ones only with a human. Through `vta-mcp`, an agent

acting for a member can propose and accept entries like any other Trust Task, which is worth noting now because agent-to-agent settlement inside a community is a natural second use.

In `vtc-service`. A `clearing` capability module, shaped like git namespaces:

- Task handlers for `clearing/entry/{propose, accept, submit, status}`, `clearing/account/{open, close, statement}`, `clearing/rules/{publish, acknowledge}` and, for the network, `clearing/hold/{place, confirm, release}`. Names are placeholders until the family is specified upstream.
- Keyspaces for accounts, the receipt log, holds and the rules versions, following the existing `vta-keyspaces` registry convention and marked for backup.
- A `Clearing` policy purpose evaluated through the ceremony pipeline, with ledger facts (balance, trailing turnover, account age, holds outstanding) added to `state`.
- The log written as audit envelopes with the existing hash chain and signed checkpoints, and a reconciler that rebuilds balances from receipts and refuses writes while the two disagree.
- The registry syncer extended to publish a `clearing` recognition record per account in good standing, and to withdraw it on suspension.

In the trust registry. No change to the query shape; standing queries already answer the question. For the network, a VTN trust registry lists communities the same way a community lists members.

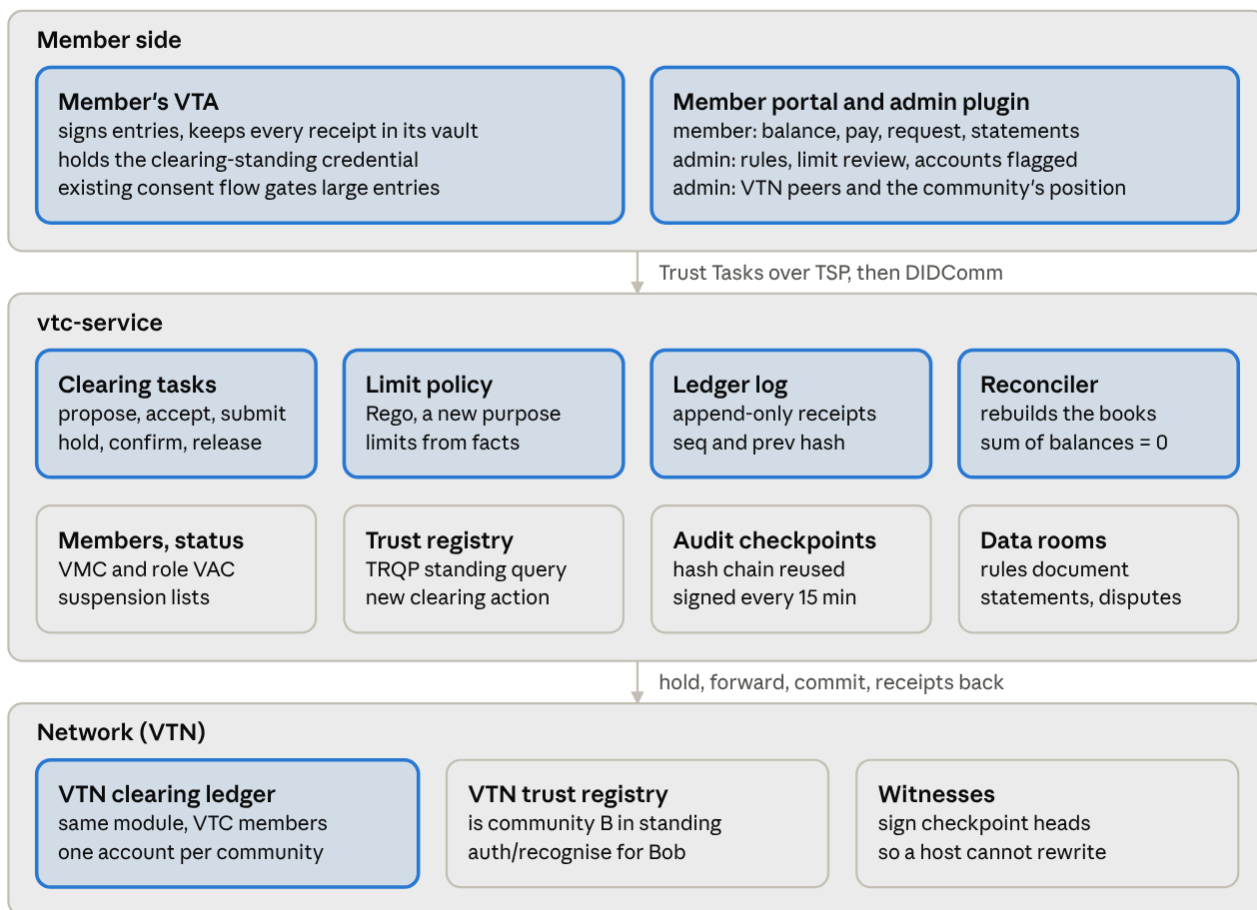
In rooms. The rules document and its history, periodic community statements (aggregate position, number of accounts, VTN balances), and dispute evidence, in a community-owned room with the `attributed` tier so authorship is provable.

In the UIs. An admin-console plugin (custom elements, the existing mechanism) for rules editing, limit review and VTN peers; member-portal pages for balance, pay, request and statements.

At the network. The VTN clearing ledger is the same module running in a VTC whose members are communities, reached over TSP or DIDComm like any peer. A community joins the VTN by the ordinary join ceremony, receives a VMC, and is given a limit by the VTN's policy.

One new module in the VTC; the rest of the clearing system is reuse

shaded = new, outlined = exists today



component map · member side, vtc-service, network

The shaded row inside **vtc-service** is the whole of the new code; the row beneath it and the network's registry and witnesses already run.

Phased plan

Four phases, each leaving something a community can use. The gate between phases is a working circle, not a date, because the risk in mutual credit is governance, not code.

- 1. Specify.** A **clearing** Trust Task family and the entry and receipt schemas go to **dtgwg-trust-tasks-tf**; the clearing-standing and balance-attestation credential types go beside the DTG v1 types; a short VTI spec section states the invariants (two signatures, sum zero, append-only, rules hash per entry). Gate: the specs and schemas are registered and **trust-tasks-rs** generates the types.
- 2. One circle.** The **clearing** capability module in **vtc-service**: accounts, countersigned entries, the log on the audit chain, the reconciler, a **Clearing** policy purpose with a default Rego bundle and a worked rules document, the registry action, member-portal

pay and request pages, and an admin plugin for limits and review. Run it in one friendly community with a fixed unit and conservative limits. Gate: a quarter of real trading with no reconciliation failures and a community that still wants it.

3. **Portable standing and holds.** The two credential types issued and refreshed; `clearing/hold/*` implemented inside one ledger first (a hold is also how a member reserves a line for an agreed job); witness anchoring of checkpoint heads; statements published to a room. Gate: a member can prove standing to a second community without that community seeing the books.
4. **Two circles, then a hub.** Bilateral clearing between two communities through mirrored accounts and the hold-and-commit thread; `auth/recognise` extended with the clearing action; a VTN rules document and limits. Then stand up a VTN clearing VTC whose members are those communities, and move the bilateral accounts onto it. Gate: a cross-community payment settles end to end with three receipts sharing one hash, and an exposure limit refuses the next one.

Deferred on purpose: floating exchange rates between units, interest or demurrage of any kind, automated limit increases without review, and the `private` room tier for statements, which waits on the DTG working group as the rooms design already notes.

Future enhancement: a Credit Commons gateway

The [Credit Commons Protocol](#), begun by Matthew Slater and Tim Jenkin in 2016 and carried since by a small community of practitioners, worked out the nested-ledger model this design leans on: ledgers that federate into a tree, each with an account for the rest of the world, balances summing to zero at every level. It has a REST protocol, a reference node, and live deployments, and the people behind it have more operating experience with mutual credit than anyone building on VTI. A later phase should let a VTC community trade with that network rather than beside it.

What that would take is a leaf-node adapter: the clearing module speaking the Credit Commons node API, member DIDs mapped to Credit Commons addresses, our VTN account mapped to their trunkward account, and a policy for what that account may do. The adapter is a modest piece of work; what VTI brings to the junction is verifiable standing and countersigned receipts on its own side, and what it inherits from the other side is reach into communities that already clear.

To keep that door open at no cost now, the entry format keeps the fields a Credit Commons transaction carries (payer, payee, quantity, unit, description), and every ledger keeps a rest-of-world account, which is their trunkward account under another name. The adapter is built when a Credit Commons group and a VTC community want to trade with each other, and not before.

Holes, risks and open questions

The technical design is the easy half. The holes below are mostly about governance, law and cold start, and several of them change the design if answered one way rather than another.

Hole	Why it matters	What the design should do about it
One person, many personas	Members can hold pseudonymous <code>did:webvh</code> personas. A line of credit per persona is a line of credit per Sybil	Open an account only against a personhood-bound DID, using the VTC's existing personhood and vetting machinery; allow personas to transact, never to hold a separate line
Regulation	A clearing circle denominated in a national currency can fall under e-money or payment-services rules depending on jurisdiction and scale; WIR holds a bank licence, Sardex operates under a specific Italian regime, LETS schemes survive on smallness and non-profit status. The hosting VTSP may sit in the regulated chain	Get counsel per target jurisdiction before phase 2; keep the first circle small, non-profit and closed; decide early whether a VTSP hosts ledgers or only the community around them. Barter income is also taxable income for members in most places; statements should make that easy
Default and exit	Someone leaves at -400. Greco's answer is that the loss is shared; the Credit Commons answer is a pre-agreed collective account. Neither is a protocol matter	The rules document must say which, and the community account and fee entries exist so either can be implemented; the admin review flow is where a stagnant debit becomes a decision
Cold start for limits	Greco's limits assume businesses with sales history. A community of open-source maintainers or association members has no sales to measure	Start lines from tier and guarantees, not turnover; let turnover raise them later. Which community goes first decides the first policy

Hole	Why it matters	What the design should do about it
Who sees what	Credit works on known standing, yet balances are sensitive. The registry already publishes membership only with consent	Standing public within the community, balances private, aggregate position published in a room; the balance attestation exists so disclosure is the member's act
Liveness of a single write-primary	If the VTC is down, nobody trades; if its ledger key is compromised, history could be rewritten	Members hold every receipt, so the log is reconstructable; witnessed checkpoints bound rewriting; read mirrors already exist for rooms and should for statements. A VTA-to-VTA offline entry that is submitted later is possible but should wait
Three-ledger atomicity	Hold-and-commit is a two-phase commit with the VTN as coordinator; a coordinator failing mid-thread leaves holds in place until they expire	Safety comes from expiry and from the rule that an entry is final only when the receiving ledger's receipt exists; liveness problems show up as released holds, which is acceptable at pilot scale
Voting does not exist yet	Rule changes, limit disputes and VTN admission all need a community decision	Phase 2 can run on the existing N-of-M admin approvals; polling is a separate module and should be sequenced, not folded in
Prior art	The Credit Commons Protocol has a REST reference implementation and running nodes; Sardex and others have a decade of operating experience	Keep the entry semantics compatible so a gateway to a Credit Commons node is possible; what VTI adds is verifiable standing and countersigned receipts, not a new accounting model
Agents paying agents	<code>vta-mcp</code> makes it trivial for an agent to spend a member's line	The consent threshold in <code>vta-policy</code> is the control; it should default on, with a low unattended limit

Governance choices for an implementing community

These are decisions for the community, or the network of communities, that adopts the tool. The tool does not make them; it lets each be written into the community's rules document and enforced by its policy, with defaults the community can change. A community preparing to open a clearing circle should be able to answer every question below before it opens accounts.

- ☐ Unit of account: national currency at par, hours, or an abstract unit?
- ☐ One account per person, bound to personhood, with personas allowed to transact but not to hold lines?
- ☐ May the VTC itself hold an account (fees, a commons budget), and under what limit? Is the hosting VTSP paid in credits or in fiat?
- ☐ Who sees a balance: the member and admins only, or also counterparties at the moment of trade?
- ☐ Loss on default: shared across members, covered by guarantors, or written off from a fee-funded reserve?
- ☐ Who operates the VTN clearing ledger: a VTSP, one of the member communities, or a VTC whose members are communities?

Sources

- [OpenVTC/verifiable-trust-infrastructure](#), read 18 Oct 2026: `README.md` , `CLAUDE.md` , `docs/05-design-notes/data-rooms.md` , `docs/03-vtc/trust-registry.md` , `docs/05-design-notes/vtc-ceremony-protocol.md` , `docs/05-design-notes/vtc-audit-checkpoints.md` , `docs/05-design-notes/vtc-mvp.md`
- Thomas H. Greco, [Credit Clearing: Pure and Simple](#); the debit-limit rule of thumb is from his books (The End of Money and the Future of Civilization, 2009) and is given here as approximate
- Ian Grigg, [Triple Entry Accounting](#) (2005)
- [Credit Commons Protocol](#), [Credit Commons Accounting](#), and [The Credit Commons Protocol](#) (Growing Commons)
- [Credit Clearing](#) (P2P Foundation wiki)