Vault Construction and Coordination Transport: Two Orthogonal Layers of Bitcoin Custody

v1.0 · Custody Agents S.L. and BitVault · 31 August 2026

Reference PDF · Licence CC BY 4.0

Canonical: intelligence.custodyagents.com/papers/two-orthogonal-layers-v1.0.pdf

Section 1 · The conflation problem

Picture a custody setup done properly. Three keys on three hardware devices, descriptors backed up, a modern vault construction with time-locked recovery paths, every best practice observed. Now watch what happens when the owner actually spends. The wallet builds a PSBT and sends it to a server. The server forwards it to the second signer, collects the signature, passes it along, and eventually broadcasts. Every byte of coordination flowed through one company's infrastructure. The keys are distributed. The coordination is not.

This gap survives because Bitcoin custody discussion keeps treating two independent questions as one. The first question: which spending paths exist, under which conditions, enforced by which on-chain mechanism? That is the vault construction layer, and it has absorbed nearly all of the ecosystem's design attention, from multisig standards to miniscript to the covenant debate. The second question: how do the humans and devices holding shares of those paths exchange the bytes required to produce a valid transaction? That is the coordination transport layer, and it has absorbed almost none.

The two questions are orthogonal. A brilliant vault construction can be operated over a fragile, observable, censorable transport. A robust transport can carry signatures for a naive two-of-two hot wallet. Neither layer substitutes for the other, and the security claims of a custody system are the composition of both. Yet ask a vendor about "custody security" and you will almost always receive an answer about the first layer alone.

This article separates the layers deliberately, examines one serious example of each, and then shows why the composition, not either layer in isolation, is what practitioners should evaluate. The vault-layer case study is B-SSL, BitVault's covenant-free Taproot construction. The transport-layer case study is PSBT coordination over Nostr relays, an

open research line published by Custody Agents. The two were developed independently, which is rather the point: layers designed to be orthogonal can be examined, criticised, and improved independently, and composed without either absorbing the other.

Section 2 · Layer one: the vault construction

For most of Bitcoin's history, "vaults" have been a promise waiting on a protocol change. The idea is old and attractive: coins that cannot simply be swept the moment a key is compromised, spending paths that impose delays and observability, recovery routes that survive the loss of everything. The standard assumption was that this required covenants, and covenants require a soft fork that has not arrived. A newer family of designs rejects that assumption. Using nothing beyond what Bitcoin ships today, Taproot script trees, relative timelocks (CSV), absolute timelocks (CLTV), and presigned transactions, these constructions deliver delay-based vault behaviour with current consensus rules. Some practitioners reserve the word vault for designs built around a pre-signed cancel transaction; B-SSL takes the delay-and-policy route instead, and this article uses the broader sense. B-SSL, the Bitcoin Secure Signing Layer published by BitVault in October 2025, is one of the cleanest examples, and it serves here as the case study for what a vault construction layer actually specifies. Elements of the design are already shipping: BitVault's first App Store release (July 2026) implements a subset of B-SSL's capabilities, with the full construction, including protection against key loss, as the stated destination.

B-SSL distributes authority across a small set of keys with distinct roles. In the reference deployment, the user holds a primary key A plus a secondary key B. A recovery participant holds a copy of B, and a co-signer participant holds key C used in delayed paths. Alongside the keys sits a required off-chain component, the Convenience Service (CS), which validates transaction policy, coordinates the co-signatures, and emits notifications. The construction caps the CS's authority rather than trusting it, as the spending paths below make concrete.

These keys are arranged into three Taproot spending paths, each gated by time and each serving a different threat horizon. The operational path combines a user key A with the co-signer C, behind a configurable delay of two hours to fifteen days, carried by the transaction itself. This is the everyday spending route, and the delay is its entire point: a signature obtained through coercion or malware no longer settles instantly. The user fallback path combines A and B after a one-year absolute timelock (CLTV) and requires no participation from the provider or the CS at all. It is the architectural guarantee that the provider cannot become a custodian: if the CS disappears, refuses to cooperate, or changes policy, the script itself hands the user a sovereign exit. The recovery path combines the recovery keys B with C after a three-year horizon, covering total user-side key loss, incapacity, and inheritance. Because Taproot commits to script paths without revealing them, none of this structure appears on-chain until a path is actually used; routine spends look like ordinary transactions.

Two design decisions are worth pausing on because they answer the standard objections to delay-based custody. The first is what this article will call hybrid delay protection. The delay on the operational path is enforced twice, by different actors: the client constructs every operational transaction with an nLockTime the network will refuse to mine early, and the CS independently parses each PSBT and declines to co-sign anything whose delay falls short of vault policy. Neither enforcement is absolute on its own: a compromised CS could waive its check, and consensus only enforces the nLockTime a transaction actually carries. Defeating the delay therefore requires compromising both the CS and the client side that builds and signs the transaction, and that is the point: the delay's strength is the independence of its enforcers, a property of policy and construction working together rather than a single consensus switch. The second is that expiration is treated as a scheduled lifecycle event rather than a lurking risk. Any vault with a three-year recovery horizon will eventually approach it; B-SSL's answer is rotation into a fresh vault instance before maturity, with timelocks reset, keys rotated, and recovery participants updated. In the current design the refresh cadence is 350 days, fifteen days ahead of the fallback path's maturity, with a second full refresh proposed at day 700 to stay ahead of the recovery horizon. Custody is modelled as a living process, a chain of vault instances, rather than a static configuration that silently ages.

The design keeps moving: the team's published work on package relay and ephemeral anchors points at the next front, fee management under a changing fee market, without altering the custody setup.

What does the construction guarantee, taken together? Before the recovery horizon matures, the provider can never move funds unilaterally: every earlier path requires a user-side key. Recovery participants cannot spend early: every route available to them is gated by consensus-enforced time, so compromise or collusion buys a waiting room measured in years, not a withdrawal; and when the horizon does mature, corporate recovery participants answer to contract and law, a deterrent the cryptography deliberately leaves in place. And the defender is structurally positioned to win the race that theft becomes: the CS sees every co-signing attempt the moment it is made, before anything reaches the network, so alerts fire while the clock runs.

Just as important is what B-SSL deliberately does not specify: how the parties communicate. The whitepaper defines which keys must sign, under which conditions, on which schedule. It does not define how the PSBT travels from the user's wallet to the CS, how the CS returns its signature, or what that channel is allowed to observe. In the current implementation this coordination question is even sharper, because the operational keypath is a MuSig2 aggregate of the user key and the co-signer key, which means the parties exchange nonces and partial signatures, not merely a serialized PSBT. The construction's guarantees are conditional on properties that some transport layer must supply. That handoff, unexamined in almost all custody discussion, is where the second layer begins.

Section 3 · Layer two: the coordination transport

Strip any multi-party custody arrangement down to its plumbing and you find the same traffic. Partially Signed Bitcoin Transactions (BIP-174) move between signers, accumulating signatures. Where Taproot key aggregation is used, MuSig2 (BIP-327) adds two preliminary rounds of nonce exchange before partial signatures can be produced at all. Setup has its own standardised ceremony (BIP-129, BSMS) for agreeing on the descriptor before any coins move. All of this is bytes that must travel between humans and devices that are rarely in the same room. Coordination transport becomes a live question from the second signature onward; a lone single-sig spend needs only broadcast. The question of how those bytes travel is the transport layer, and in practice it has one dominant answer: a coordination server operated by the wallet vendor.

The vendor coordinator works, which is why it is everywhere. But it concentrates exactly the properties that the on-chain construction just distributed. Three consequences follow. First, metadata: the coordinator observes who signs, when, in what order, against which descriptor, which amounts to the full spending graph of the arrangement and, for a firm or family, its organisational structure. Second, obstruction: a coordinator can refuse to forward, or simply delay, partial signatures. The signers keep their keys and lose their liveness; under adversarial conditions this is operationally indistinguishable from a custody freeze. Third, fragility: one operator, one point of compromise, one subpoena surface, shared by every customer simultaneously.

There is also a regulatory dimension European practitioners can no longer treat as theoretical. MiCA regulates custody as a service: providing custody and administration of crypto-assets on behalf of clients means safekeeping or controlling, on behalf of clients, crypto-assets or the means of access to them, and it triggers CASP authorisation. Recital 83 draws the complementary line: hardware and software providers of non-custodial wallets, who neither possess nor control client assets, sit outside the regime. The settled part is the axis: authorisation follows control. The open part is what control means one layer up from keys, and here is the reading we defend: an operator with the standing technical capacity to systematically block or delay a client-initiated transaction is exercising a form of control, whatever the key architecture says. Under that reading, non-custodial claims that hold at the key layer can fail at the transport layer, and the coordination path becomes a compliance surface rather than an implementation detail.

The alternative is to treat coordination as a protocol rather than a service, and the community has been probing this direction since 2022: Joinstr coordinates coinjoin rounds over Nostr relays, Munstr demonstrated interactive MuSig signing sessions over them, Smart Vaults specified PSBT and miniscript policy management on the same substrate, and Nunchuk ships collaborative signing through encrypted relay messages. What the lineage has not yet produced is a formal treatment of the transport itself: its invariants, its threat model, its failure modes at serious stakes. The PSBT Coordination over Nostr Relays working note published by Custody Agents is an attempt at that treatment. It routes PSBT and MuSig2 coordination over Nostr relays, borrowing the remote-signer transport pattern standardised in NIP-46. The shape is simple to state. Signers exchange end-to-end encrypted message fragments through a small mesh of

independent relays. No relay holds key material, no relay can decrypt the fragments it carries, and no relay learns the spending graph, because it cannot distinguish a nonce from a partial signature from noise. The only capability an adversarial relay retains is denial of service, and redundancy across independent relays reduces even that to an inconvenience. Assuming a sufficiently independent relay mesh, the failure mode of the transport collapses from "can observe, delay, and censor" to "can, at worst, drop packets," which is the property a non-custodial claim actually needs.

Honesty requires stating clearly what this is and is not. It is a published architecture sketch with named open engineering problems, not a shipped product. The hardest of those problems is precisely the one B-SSL's current implementation makes relevant: MuSig2's security proof forbids a signer from ever using the same secret nonce in more than one signing attempt, and an asynchronous relay substrate with retries and duplicate delivery is exactly the environment in which naive implementations do so. Getting this right requires formal work on transport invariants, adversarial testing, and cryptographic review, and the working note frames it as an invitation to the ecosystem rather than a solved claim. The relevant point for this article is narrower: whether over a vendor server today or a relay mesh tomorrow, the transport is a load-bearing layer of every custody arrangement, with its own threat model, its own failure modes, and its own claims to verify. Evaluating a custody design without evaluating its coordination path is reviewing half a system.

Section 4 · The composition

Put the two layers side by side and the full custody lifecycle comes into focus. Setup first: the parties agree on the descriptor, exchange public keys, and verify the vault structure, a ceremony standardised by BSMS (BIP-129). Operation next: for every spend, PSBTs and, where key aggregation is used, MuSig2 nonces and partial signatures travel between signers over some transport. Settlement last: the Bitcoin network enforces the timelocks and script conditions the construction defined. The vault layer owns the first and third phases. The transport layer owns the middle, and the middle is where every transaction is actually built.

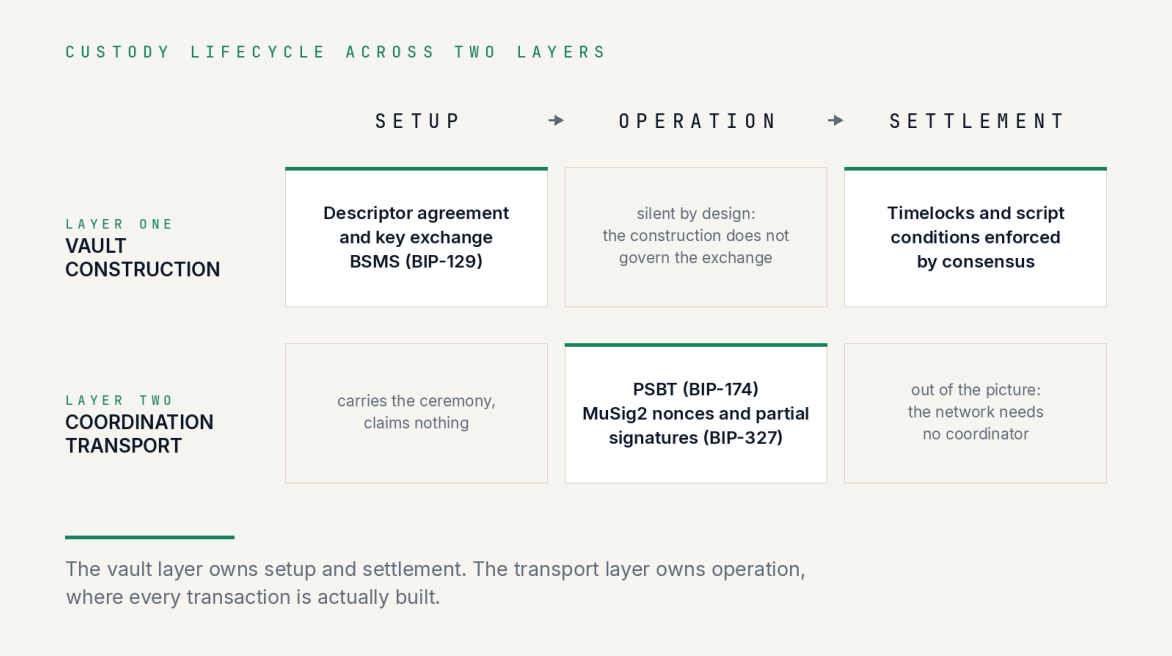

Figure 1 · The custody lifecycle across the two layers.

The composition argument is sharpest at the point where B-SSL is most interesting: the Convenience Service. The construction goes to great lengths to keep the CS powerless. It cannot move funds alone. The user fallback path routes around it entirely. Its co-signature is policy-gated, withheld until the delay elapses, and the on-chain structure caps what it could ever do alone. But now ask what the coordination rails observe, and the picture depends entirely on who operates them. The CS must see each PSBT in plaintext; that is its job, policy cannot be validated blind. The transport, by contrast, should see nothing beyond transport metadata. The trouble with the incumbent model is that the same vendor operates both the co-signing service and every rail the coordination crosses, so observation and obstruction concentrate in one

operator: it sees each spending attempt the moment it exists, amounts, timing, frequency, the owner's entire operational pattern; it can delay traffic it dislikes; it can go dark and take liveness with it. None of this violates the on-chain construction, because the construction never claimed to govern it.

This is the central sentence of the article: the on-chain construction limits what the co-signer can do; the transport layer limits what the coordination path can know and what it can quietly obstruct. Both limits are necessary. Neither is sufficient alone. A delay-enforcing co-signer reached through centralised, observing rails reintroduces, one layer up, exactly the intermediation the script design worked so hard to eliminate. The trust-minimisation claim of any CS-style participant is only as strong as the transport that connects it to the user.

The current B-SSL implementation makes the dependency concrete rather than theoretical. Its operational keypath is a MuSig2 aggregate of the user key and the co-signer key, which means every routine spend involves interactive rounds of nonce and partial-signature exchange. Whatever carries those rounds inherits real obligations: message ordering within a signing session, protection against replayed or duplicated deliveries, and above all never inducing a signer to reuse a secret nonce. The transport is not a passive pipe under such a scheme. It is a component of the security model, with invariants of its own to uphold.

None of this is specific to B-SSL, which is precisely why the composition matters. Any inheritance construction whose recovery participants coordinate to produce a PSBT, any collaborative-custody quorum, any deadman-switch design with a notification layer faces the same split: a construction that defines authority, and a transport that defines observability and liveness. The two layers even assume things of each other, and stating those assumptions is clarifying. The vault assumes the transport delivers coordination messages reliably and privately, without becoming an authority of its own. The transport assumes the vault defines signing policy precisely enough that the transport can remain entirely blind to it, forwarding ciphertext without ever needing to understand, and therefore without ever being able to censor, what it carries. Each layer is strongest exactly where the other is silent.

Section 5 · What remains open

The vault layer described in Section 2 is specified, published, and reviewed. The transport layer described in Section 3 is younger, and honesty about its open problems is part of the argument, not a weakness in it. Five problems stand between the architecture sketch and a deployable specification.

Accountability without a global observer. A vendor coordinator at least produces logs; an ephemeral relay mesh produces nothing by default. What cryptographic structure on individual messages lets honest parties prove misbehaviour without reintroducing someone who watches everything?

Policy-blind transport that still resists garbage. A relay that validates what it forwards becomes a censor. A relay that forwards anything becomes a spam vector. Where exactly the line sits, and whether it can sit anywhere other than fully client-side, needs formal analysis rather than assertion.

A threat model sized to the stakes. Custody arrangements of the kind B-SSL targets routinely protect eight-figure positions. Supply-chain attacks on signers, coercion of signing parties, and protocol-level liveness attacks all belong in the model, and the coordination layer must state which of them it absorbs.

Liveness roles in a decentralised setup. A vendor server earns its place by being always online: someone must broadcast matured transactions, watch the mempool, and send notifications. Remove the single server and those duties do not disappear; they need owners. Which participants hold the broadcast and monitoring roles, how they coordinate so that someone is always awake, and whether a set of independent convenience services can jointly provide what one vendor provides today are open design questions, not details.

MuSig2 nonce safety under adversarial delivery. The sharpest problem, and after Section 4, the least abstract one. BIP-327's security collapses if a signer ever reuses a secret nonce, and its proofs assume orderly communication. Retries, duplicate delivery, and adversarial scheduling are exactly the conditions of a relay substrate. The engineering question is which invariants the transport must enforce so that nonce uniqueness holds end to end; the cryptographic question is proving those invariants sufficient. Property-based testing explores the state space and finds counterexamples, but

exploration is not proof, and this problem needs specialist cryptographic collaboration, which is exactly what the research line is structured to attract.

These problems are stated publicly, in the working note and on the Delving Bitcoin thread where B-SSL itself was first reviewed, because that is where they should be attacked: in the open, by implementers, cryptographers, and critics. Technical criticism is the most valuable contribution either of these layers can receive.

Section 6 · Implications for practitioners

For the practitioner evaluating custody offerings, whether for themselves, a family, or an institution, the takeaway is a habit: ask the two questions separately, every time. First, what does the on-chain construction guarantee, and what happens when each key, each participant, and each service disappears or turns hostile? Second, what does the coordination path observe, what can it delay, and what fails when it fails? A vendor with an excellent answer to the first question and no answer to the second has described half a security model. So has a vendor in the opposite position. It is also the habit our own work systematises: the Sentinel assessment framework Custody Agents maintains for vault constructions asks exactly these two questions, in public, of every construction it covers.

The two-layer lens also changes how progress in this space should be read. Vault constructions and coordination transports can, and should, evolve independently, be criticised independently, and be composed freely. A world where one vendor's vault only works with that vendor's coordinator has quietly re-centralised what the scripts decentralised. A world of orthogonal, composable layers is the one in which self-custody genuinely competes with custodial convenience, on safety and on operational dignity alike. One credible destination, raised inside this collaboration itself, is a mesh of independent convenience services, chosen by the user at setup and speaking a common transport.

Both firms behind this article continue building at their respective layers, and the composition sketched here is the direction of travel we are committed to. The technical conversation continues in the open, on the Delving Bitcoin thread linked below, where criticism of either layer is welcome.

About this article

Co-published by BitVault and Custody Agents S.L. on 31 August 2026 · Version 1.0

Written by Rafael Turon (Custody Agents), with the B-SSL sections reviewed for accuracy by Riccardo Biffi and Francesco Madonna (BitVault).

BitVault: Beyond Keys Protect Humans · bitvault.sv

Custody Agents: Sentinel, a vault-agnostic governance layer for Bitcoin custody · custodyagents.com

References

B-SSL concept review and technical discussion, Delving Bitcoin: delvingbitcoin.org/t/2047

PSBT coordination over Nostr relays, working note: intelligence.custodyagents.com/articles/psbt-over-nostr

B-SSL whitepaper (October 2025): github.com/ilghan/bssl-whitepaper

Package relay and ephemeral anchors, BitVault blog: blog.bitvault.sv

Technical criticism of either layer is welcome on the Delving threads linked above.

Also published at blog.bitvault.sv.

Revision history

v1.0 · 31 August 2026 · First distributable version.

Versioning: typo-level fixes are published as 1.0.x, substantive additions that do not change conclusions as 1.x, and any change of conclusion or recommendation as 2.0. Files are never replaced in place; every revision is a new file at a new versioned URL, agreed by both firms under the corrections protocol.

Integrity: the SHA-256 of this file is published alongside it at the canonical URL.