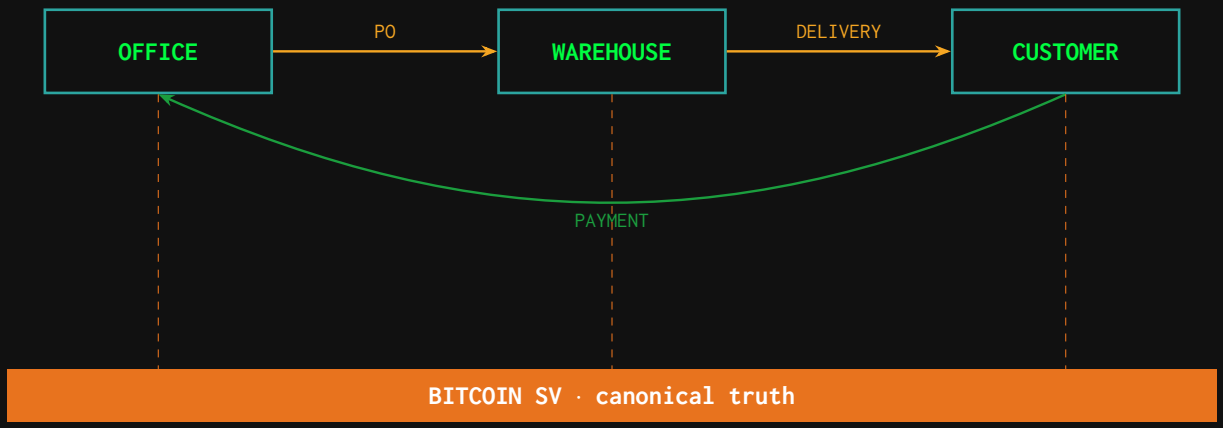
Scripted Supply

Legible, replayable contracts for procurement on Bitcoin SV

A Henceforth whitepaper

// today’s procurement flow:



Where EDI ends,
the ledger begins.

Henry Hudson

v0.1 · 2026

// ok.

Abstract

Every B2B purchase order today is a polite fiction. The invoice says goods shipped; the receipt says goods received; the payment instruction says trade complete — but no party can prove any of those documents corresponds to anything that actually happened in the warehouse. Reconciling the fiction against reality costs five to ten percent of every transaction value, in labour, dispute, and audit.

Scripted Supply replaces that fiction with cryptographic enforcement, building on Craig Wright's architectural thesis [1]. Every purchase order is an on-chain state token whose locking script encodes the exact conditions under which payment may move and ownership may transfer. Delivery is not a confirmation email; it is a cryptographic attestation signed by the warehouse and the customer, atomically bundled with the payment that settles the trade. Half-completed flows — invoices paid for goods never shipped, goods delivered against invoices that get lost — become structurally impossible.

Contracts are written in a restricted Forth dialect: human-readable for procurement officers and lawyers, compiled deterministically to Bitcoin Script for consensus enforcement. The same artefact a CFO reads is, byte-for-byte, the artefact the chain executes. Suppliers and customers participate without installing anything: a one-time email link opens a browser-resident signing surface that produces real ECDSA signatures over their own private keys, never leaving their device.

The platform ships as a new application within Henceforth, an existing Bitcoin SV operator app on the iOS and macOS App Stores. A representative 1,000 PO/month customer recovers approximately **$88,000 per year** on operational reconciliation costs alone. Three months free for the anchor pilot customer.

1 Introduction: The Cost of Well-Formed Lies

In the average B2B transaction today, four documents are exchanged: a purchase order, a shipping notice, an invoice, and a payment instruction. All four are syntactically correct. Any combination of them can be false. The system that produced them — Electronic Data Interchange, designed in the early 1980s and not seriously reconsidered since — has no way of telling. It is the architectural equivalent of a fax machine: well enough understood to be entrenched, badly enough designed to embarrass anyone who has to operate it daily.

The defining feature of EDI is not that it sometimes fails. The defining feature is that it cannot tell the difference between a true and a false document. As Craig Wright observes in the architectural critique on which this work is built:

> EDI does not fail when it transmits incorrect information. It fails when it successfully transmits well-formed lies. — Wright [1]

A correctly-formatted invoice for goods that were never shipped passes every EDI validator. A correctly-formatted delivery confirmation for items that never arrived passes every EDI validator. A correctly-formatted payment instruction that does not actually correspond to any fulfilled order passes every EDI validator. The system's job ends at syntactic correctness — whether the message conforms to the X12 or EDIFACT schema. Whether the message corresponds to anything that actually happened in the physical world is, in the architecture's view, somebody else's problem.

1.1 The reconciliation tax

That "somebody else" is the procurement department, the accounts payable team, the auditor, and the lawyer. Industry estimates suggest that the cost of reconciling EDI-based commerce against physical reality runs to between five and ten percent of transaction value. That cost is not theoretical — it shows up on every B2B income statement as a line item, divided across multiple functions:

- Reconciliation labour: full-time-equivalent staff matching purchase orders to receipts to invoices to payments.
- Dispute resolution: weeks of email threads, phone calls, and occasional litigation when the four artefacts disagree.
- Audit preparation: external auditors paid hourly to confirm that the company's records of what happened correspond to what happened.
- Value-Added Network (VAN) fees: the per-transaction toll levied by the messaging intermediaries who route EDI between trading partners.
- Software integration: the perennial cost of mapping each new partner's slightly-different EDI dialect into the company's enterprise resource planning system.

For a mid-market business handling 1,000 purchase orders per month, the total cost can easily exceed $90,000 per year. For an enterprise handling tens of thousands of POs per month, the figure runs into the millions.

1.2 Why messaging cannot fix messaging

The instinct of every previous attempt to improve EDI has been to improve the messaging — richer schemas, faster networks, more validation rules. None of these address the architectural flaw, because the flaw is not in the messages. The flaw is that no message, however well-formed, can cause a payment to move or a contract to be enforced. Messages report; they do not act. Every commercial outcome still requires a human to read the messages, decide that they are consistent with reality, and instruct a separate system to settle the trade.

What is needed is not better messaging. What is needed is a system in which the validation and the execution are the same act — where confirming that a delivery

happened and releasing the corresponding payment are not two separate steps with two separate trust assumptions, but a single, cryptographically enforced transition. This is precisely what Bitcoin Script makes possible, and what this paper proposes to build.

1.3 What this paper proposes

Scripted Supply replaces the settlement moment of B2B commerce — the point at which ownership transfers and payment moves — with a cryptographically enforced state transition on the Bitcoin SV ledger. The purchase order ceases to be a message and becomes a state token: an on-chain unspent transaction output whose locking script encodes, opcode by opcode, the exact conditions under which it may be consumed. Delivery is no longer a confirmation email; it is a cryptographic attestation by the warehouse, recorded on chain, that the parcel left the dock. Payment is no longer a separate banking instruction; it is the same transaction that consumes the purchase order, atomically. Whether the chain accepts that transaction is the question and the answer of whether the trade settled.

The rest of this paper develops the architecture in three movements:

1. The architectural fix (Section 2): how purchase orders, deliveries, and payments are represented as on-chain state, and why Bitcoin SV is the right substrate.
2. The adoption fix (Section 3): why the article from which this work descends does not solve the hardest practical problem — getting trading partners to participate — and how Scripted Supply solves it through an asymmetric deployment model that requires zero installation from suppliers.
3. The legibility fix (Section 4): why on-chain contracts must be readable by procurement officers and lawyers, not just by cryptographers, and how a restricted Forth dialect provides exactly that legibility while compiling to Bitcoin Script.

Section 6 presents the economic model and shows the credible savings for a typical mid-market customer. Section 8 offers the terms under which the first anchor customer can run a pilot.

2 The Architectural Fix

The architectural insight that drives this work, articulated by Craig Wright in Scripted Supply: A Bitcoin-Based Architecture [1], is that the difference between messaging and execution corresponds exactly to the difference between EDI and Bitcoin Script. Wright's framing is mathematical:

> There is no formalism by which the reception and syntactic validation of an EDI message leads to an enforced, convergent commercial outcome.— Wright [1]

The implication, developed throughout Wright's paper and adopted directly here, is that the only architecture that escapes the well-formed-lies failure mode is one in

which the validation and the execution are the same primitive operation. Bitcoin Script provides that primitive; Wright shows how to use it for commerce.

2.1 From message to state token

In conventional EDI, a purchase order is a message: a structured document sent from one party to another, asserting an intent to buy. The receiving party may agree, may disagree, may file it incorrectly, may lose it. The PO has no independent existence outside the parties' respective databases, and those databases can disagree without either party knowing.

In Scripted Supply, a purchase order is a state token: an unspent transaction output (UTXO) on the Bitcoin SV ledger, sitting at a specific outpoint, locked by a specific script. The state token has exactly one canonical existence — the chain — and either party can verify its existence and its terms by querying any Bitcoin SV node. There is no "my copy versus your copy"; there is only the copy.

The locking script of the PO state token encodes the precise conditions under which it may be spent, and therefore the precise conditions under which the underlying commercial event may proceed. A simplified four-path script for a single purchase order looks like this:

Bitcoin Script: PO Locking Script (4 spend paths)

```
OP_DUP OP_0 OP_EQUAL OP_IF                    // path 0 - fulfilment
   OP_DROP
   <customerScanPreimage> OP_SWAP OP_SHA256
   <scanHashCommitment> OP_EQUALVERIFY
   <officePubKey> OP_CHECKSIGVERIFY
   <warehousePubKey> OP_CHECKSIGVERIFY
   <customerPubKey> OP_CHECKSIG
OP_ELSE OP_DUP OP_1 OP_EQUAL OP_IF            // path 1 - amendment
   OP_DROP <officePubKey> OP_CHECKSIG
OP_ELSE OP_DUP OP_2 OP_EQUAL OP_IF            // path 2 - cancellation
   OP_DROP
   OP_2 <officePubKey> <warehousePubKey> OP_2 OP_CHECKMULTISIG
OP_ELSE                                       // path 3 - timeout
   OP_3 OP_EQUALVERIFY
   <expiryHeight> OP_CHECKLOCKTIMEVERIFY OP_DROP
   <officePubKey> OP_CHECKSIG
OP_ENDIF OP_ENDIF OP_ENDIF
```

Each branch corresponds to one possible commercial outcome. Figure 1 shows the lifecycle of a purchase order in state terms: a main fulfilment flow from draft to settled, plus four side branches for amendment, cancellation, timeout, and post-delivery dispute. Each transition between states corresponds to exactly one Bitcoin SV transaction, and each transaction is enforced by the locking script above.

The fulfilment branch requires a signature from the buyer (office), a signature from the seller (warehouse), a signature from the receiving party (customer), and — crucially — the cryptographic preimage of a hash that was printed on the physical parcel as a scannable code. The amendment branch allows the buyer to revoke and reissue. The cancellation branch requires the buyer and seller to agree. The timeout

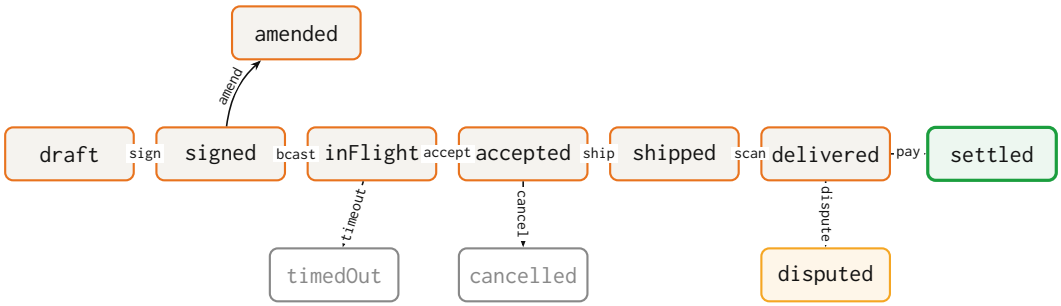


Figure 1: The purchase-order state machine. Main flow is left-to-right from draft to settled; side branches handle amendment (office signature), timeout (after OP_CHECKLOCKTIMEVERIFY expiry), cancellation (office + warehouse multisig), and post-delivery dispute (customer signature within seven days). Each transition corresponds to exactly one Bitcoin SV transaction enforced by the locking script.

branch lets the buyer reclaim the PO after a specified block height if no fulfilment has occurred. There is no fifth path. The chain does not enforce what it has not been instructed to enforce; equally, the chain enforces, exactly and forever, what it has.

2.2 Atomic delivery-versus-payment

The most important property of the state token architecture is that it makes atomic delivery-versus-payment (DvP) possible without a trusted intermediary. In a single Bitcoin transaction, the fulfilment-path spend of the PO state token can be co-bundled with a payment from the customer's wallet to the office's wallet. The transaction either confirms in full — in which case the PO is consumed, the warehouse's delivery attestation is recorded, the customer's payment moves, and the audit trail is committed — or it fails in full, leaving every party in their prior position. There is no possibility of a payment moving without a corresponding delivery attestation, and no possibility of a delivery attestation being recorded without payment moving. The pathologies of half-completed EDI flows — the invoice paid for goods that never arrived, the goods delivered but the invoice lost — become structurally impossible.

> Verification disappears because the ledger is immutable. Auditing disappears because the ledger is immutable. [. . .] Legal costs drop to zero in all cases where contractual behaviour is governed entirely by script logic.— Wright [1]

Wright's claim is correct within the scope of what scripts can see. Sections 3 and 4 address the cases where scripts cannot see — the oracle problem and the legibility problem — which the article notes but does not solve.

2.3 Why Bitcoin SV specifically

Several blockchain platforms support some form of programmable settlement. Scripted Supply is built on Bitcoin SV for four substantive reasons.

2.3.1 Restored Script.

Bitcoin SV is the only Bitcoin-derivative chain that preserves the full original Script opcode set, including OP_MUL, OP_LSHIFT, OP_CAT, and the unrestricted use of OP_IF branching. The four-path PO script above is impossible to express in Bitcoin Core (BTC) without expensive workarounds, and would be a Solidity-style contract with a custodial smart-contract trust model on Ethereum. On Bitcoin SV it is a single unspent output with a vanilla script.

2.3.2 Predictable, sub-penny transaction fees.

The economic model in Section 6 assumes BSV transaction fees of approximately $0.01–$0.05 per PO. This is true today and has been true since 2018. By contrast, Bitcoin Core fees fluctuate between $1 and $50 per transaction and Ethereum gas can spike higher; neither is viable for high-frequency B2B settlement.

2.3.3 Large blocks for on-chain payloads.

Bitcoin SV blocks routinely exceed 1 GB. Encrypted purchase-order payloads (line items, addresses, prices) can be stored off-chain with on-chain hash commitments, but where on-chain payload is desirable — for example, for delivery-photograph hash chains or for inline carrier bill-of-lading numbers — the chain has the capacity.

2.3.4 Native identity.

Bitcoin SV has a mature identity stack via the Bitcoin Attestation Protocol (BAP) and BRC-42 (Type42) key derivation [2], which permits per-relationship and per-purchase-order key derivation without on-chain key reuse. This is what makes the privacy story of Section 7 possible.

The choice of Bitcoin SV is not ideological; it is the only Bitcoin-derivative chain on which the architecture described in this paper is technically and economically feasible.

3 Beyond Wright: The Adoption Problem

Wright's paper is architecturally sound but commercially silent. It establishes that on-chain settlement is technically superior to EDI messaging; it does not address the question that determines whether the architecture will ever be used: what does the supplier have to do?

3.1 Wright's gap, honestly named

EDI is a network-effect business. Its value to any one company is determined by how many of that company's trading partners use it; its cost of switching is the integration overhead multiplied by the number of partners. A purely cryptographically-pure alternative that requires every supplier and every customer to install a Bitcoin wallet, manage a seed phrase, and learn a new operational

discipline will not displace EDI, because the install friction at the partner edge is the dominant cost.

Wright's omission is honest but consequential: he describes what the system would do if everybody used it, without describing how to get anybody to start. Scripted Supply takes Wright's architecture as given and addresses the omission directly.

3.2 The asymmetric deployment model

Scripted Supply resolves the adoption question with an asymmetric deployment model. The platform operator — the company issuing purchase orders — runs everything: a Bitcoin SV wallet, an indexer, a relay server, and an operator app on iOS and macOS. The platform operator is a sophisticated party with cryptographic infrastructure.

The platform's counterparties — suppliers, warehouses, customers — run nothing. They receive an email containing a one-time link. They click it. They are presented with a mobile-responsive web page that shows them the purchase order, asks them to accept or reject, and produces a cryptographically real signature on their behalf. They are not told that any of this is on a blockchain, and they are not asked to install anything.

The cryptographic content of the supplier's interaction is preserved by performing the key generation and signing in their browser, using the established @noble/secp256k1 library which adds approximately ten kilobytes of JavaScript and produces real ECDSA signatures over the secp256k1 curve used by Bitcoin SV. The browser stores the supplier's per-purchase-order private key in IndexedDB, scoped to the relay's origin; the key never leaves the device. When the supplier signs an acceptance or a delivery attestation, the signature is real, verifiable on chain, and indistinguishable from a signature produced by a heavy-weight native wallet.

> **Not custodial.** The relay server holds no signing keys for the supplier. It routes signed payloads onto the chain; it does not produce them. The supplier's cryptographic sovereignty is genuine, even though they never installed an application.

Figure 2 shows the resulting deployment topology. The operator runs a native app and a Mac mini relay; the supplier interacts via a single email link; both parties verify against the same Bitcoin SV chain.

3.3 The pattern that always wins

Every B2B platform that has achieved meaningful adoption — Stripe, Plaid, DocuSign, Square — has shipped a v1 in which the operator-side party does the technical work and the counterparty interacts via a web form. The operator gets the audit trail, the cryptographic guarantees, and the integration into their existing systems. The counterparty gets a link in their email. The platform's job is to make sure that the cryptography under the link is real.

Scripted Supply ships this pattern from day one. The native iOS and macOS app is for the operator side; the web portal at scripted.henceforth.club is for everybody else. The web portal is not a degraded alternative to the app; it is the front door for the ninety-five percent of users who will never install anything. The native app is the

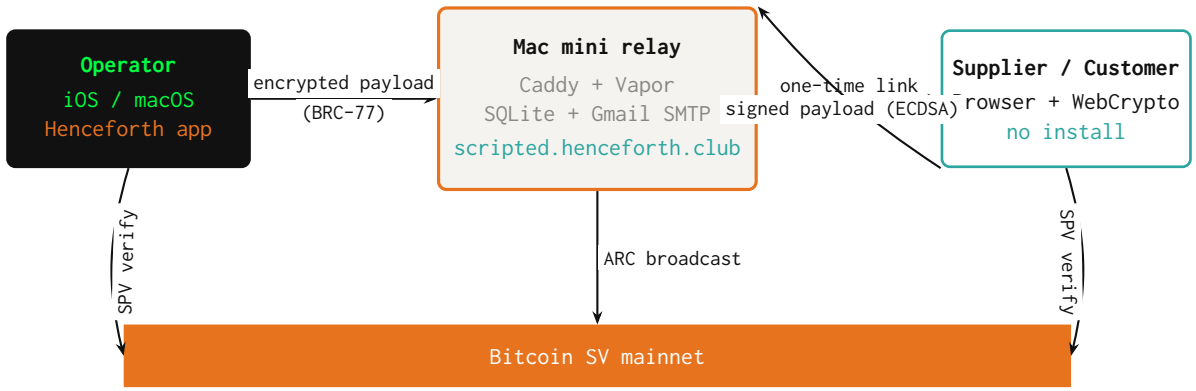


Figure 2: The asymmetric deployment topology. The operator (left) runs a native iOS/macOS application backed by their own Mac mini relay; the counterparty (right) participates via a browser link with no installation. Cryptographic signatures are produced at the edges (operator's wallet keys; supplier's browser-resident WebCrypto keys) and consumed by the chain. Both parties verify on-chain state independently via SPV.

back door for the five percent who eventually want to. Both produce equally valid on-chain signatures.

4 Forth-as-Contract-Language: A Legible Substrate

The architecture described in Section 2 has a problem Wright does not address. Bitcoin Script is a low-level concatenative bytecode. A correctly-written locking script is, to a procurement officer, a wall of hexadecimal. A subtly-wrong locking script is also a wall of hexadecimal, indistinguishable to the same procurement officer. An immutable contract that nobody on the buying side can read is not a contract; it is a faith-based ritual. Worse: the immutability that gives the architecture its strength becomes a liability when the contract is wrong, because there is no edit-after-the-fact.

4.1 The legibility gap

Consider what a finance director or a chief procurement officer is asked to do today, in any B2B setting where contracts touch software. They review a contract drafted by their counterparty's lawyer. They negotiate redlines. They sign. They store the document. If a dispute arises, they retrieve the document and read it; if necessary, they hand it to a court that also reads it. The artefact is legible to every party who needs to assess it.

The on-chain version of the same flow, naively implemented, asks the finance director to sign off on a SHA-256 hash. They cannot read the hash. They cannot redline the hash. They cannot retrieve the hash six months later and confirm it says what they thought it said. They cannot hand the hash to a court. The on-chain artefact, while cryptographically superior, is operationally illegible.

This is not a hypothetical concern. It is the dominant reason that procurement-tech

blockchain projects have failed for ten years — not because the cryptography did not work, but because the people whose budgets paid for the projects could not read what they had bought.

4.2 Why Forth

Bitcoin Script is, structurally, a Forth: a stack-based concatenative language in which operations push and pop values from a single shared stack. This is not coincidence. Forth was the linguistic family Satoshi reached for when designing the protocol because it is the simplest possible representation of executable logic, and the original Bitcoin Script is closer to Forth than to any other working programming language.

If Script is a Forth, then a higher-level Forth dialect is the natural source language for it. Where Script gives us <pub> OP_CHECKSIG, our Forth dialect gives us require-warehouse-sig. Where Script gives us <height> OP_CHECKLOCKTIMEVERIFY OP_DROP, our Forth dialect gives us 30 days-net payment-terms. The mapping is one-to-one and structure-preserving: a sequence of Forth words compiles, deterministically, to a sequence of Script opcodes. There is no hidden runtime, no interpreter, no semantic gap. What the Forth says is, character-for-character, what the chain enforces.

This matters because it means the contract a procurement officer reads is, provably and inspectably, the contract the chain enforces. The Forth source is the contract; the compiled hex is a verification of the contract. The artefact that lives in the company's contract repository is human-readable. The artefact that runs on the chain is the same artefact, in a different alphabet.

4.3 An annotated sample

Below is a complete Scripted Supply purchase order, expressed as Forth. A procurement officer can read it; a lawyer can review it; an auditor can replay it. The compiler converts it to Bitcoin Script for on-chain enforcement.

Forth: Sample PO Contract (acme-widget-500-v1)

```
CONTRACT-BEGIN scripted-supply.po.v1
  \\ Counterparties
  s" acme-warehouse@acme.com" counterparty
  s" zenith-stationery@zen.io" customer

  \\ Commercial terms
  10000 unit-price-max
  500   quantity
  30 days-net payment-terms
  s" SKU-WIDGET-A" sku

  : ACCEPT-DELIVERY ( -- )
    require-warehouse-sig
    require-customer-scan-preimage
    delivery-attest ;
```

```
    : RELEASE-PAYMENT ( -- )
      require-customer-receipt
      require-after-delivery
      payment-release ;

    : DISPUTE ( reason -- )
      require-customer-sig
      7 days-after-delivery within
      dispute-flag ;
  CONTRACT-END SCRIPT-SAVE acme-widget-500-v1
```

What this contract says, in English: "A purchase order from us to Acme Warehouse, for the benefit of our customer Zenith Stationery, for 500 units of SKU-WIDGET-A at a maximum unit price of $100. Payment due 30 days after delivery. Acceptance of delivery requires the warehouse's signature and the customer's scan of the parcel's QR code. Payment is released when the customer confirms receipt, but no earlier than the delivery attestation. The customer may dispute within seven days of delivery."

What the compiler produces, in Bitcoin Script: a locking script with four mutually exclusive spend paths (fulfilment, amendment, cancellation, timeout), as shown in Section 2, with the specific public keys and time-lock values that correspond to the Forth source above.

4.4 The audit-replay artefact

The act of saving a contract via SCRIPT-SAVE produces not just the compiled Script hex, but a bundle of artefacts that together constitute the legal record of the agreement:

1. The Forth source verbatim, as the counterparties agreed it.
2. The compiled Script hex that will be embedded in the on-chain locking script.
3. A source map linking each Forth word to the byte range of Script it produced, so a reviewer can trace the source-to-hex correspondence.
4. Auto-generated test fixtures: for every require-X-sig clause, a positive test (correct signature unlocks the appropriate path) and a negative test (incorrect signature fails). These run before the contract is broadcast.
5. A dry-run trace: a deterministic execution of the contract against mocked counterparty inputs, showing exactly which path the contract takes under which conditions.

This bundle is the artefact that ships to the counterparty for review, the artefact that lives in the company's contract repository, the artefact that the auditor reviews at year-end, and the artefact that a court would examine in a dispute. The chain enforces; the bundle explains.

4.5 Why this is the differentiator

Every blockchain B2B platform to date has had a story for the cryptography and no story for the legibility. Their pitch documents talk about consensus, about Byzantine fault tolerance, about smart-contract security. None of those words mean anything to the chief procurement officer who has to sign the purchase requisition.

The Forth-as-contract-language layer is the part of Scripted Supply that converts a cryptographic system into a procurement system. It is what makes the on-chain enforcement story credible to the people whose job titles do not contain the word "crypto." It is also, technically, the most novel contribution of this work. The article that inspired this project assumed that the script would be written by a sophisticated party. Scripted Supply assumes that the script will be written — and read — by a procurement officer, and designs the language accordingly.

5 Henceforth: The Operator Platform

Scripted Supply does not ship as a standalone product. It ships as a module within Henceforth, an existing Bitcoin SV operator application that Henry Hudson has been developing since 2022. This matters for one reason: it eliminates the perpetual problem of new blockchain products, which is that the buyer is being asked to trust both a new architecture and a new vendor at the same time.

5.1 What Henceforth already does

Henceforth ships today on iOS and macOS via the Apple App Store. It is a fully-functional Bitcoin SV wallet and scripting environment, with the following capabilities relevant to Scripted Supply:

- **Wallet**: BIP-32 hierarchical-deterministic key management, Type42 [2] per-relationship key derivation, Keychain-backed private key storage, multi-address UTXO management.
- **Identity**: BAP-derived identity [3], signed-message authentication, Paymail [4] integration for human-readable addresses.
- **Settlement**: ARC broadcasting via the GorillaPool and TAAL networks with automatic failover; transaction construction with fee estimation; BEEF (Background Evaluation Extended Format) for offline-verifiable payment proofs.
- **Validation**: simplified payment verification (SPV) against a locally-validated header chain anchored to a hard-coded checkpoint and synced via WhatsOnChain's WebSocket service; Merkle proof verification in three formats (Bitcoin Core, TSC, WhatsOnChain).
- **Privacy**: BRC-2 ECIES encryption [5] (ECDH + AES-CBC + HMAC-SHA256) for end-to-end encrypted messaging compatible with the @bsv/sdk JavaScript ecosystem.
- **Programmability**: a Forth-2012 compliant interpreter with Bitcoin-specific extensions for transaction construction, script primitives, and on-chain data

composition. This is the foundation on which the Scripted Supply contract language of Section 4 is built.

- **Interoperability**: BRC-100 wallet interface for application-to-application transaction signing requests, using a custom URL scheme for cross-app integration.

In aggregate, Henceforth represents approximately 60,000 lines of production Swift code under active development, with comprehensive test coverage on the cryptographic and settlement layers.

5.2 Why this matters for Scripted Supply

The implication is that Scripted Supply is not a greenfield rewrite of a Bitcoin SV stack. It is a new product target within an existing application that already implements the parts of the Bitcoin SV stack that matter: the wallet, the settlement, the validation, the cryptography, the identity. The work that remains for Scripted Supply is the EDI-specific layer — the purchase-order state machine, the Forth contract language, the supplier web portal, the asymmetric relay — and not the foundational Bitcoin SV plumbing.

The practical consequence for an anchor customer is that Scripted Supply enters their evaluation cycle with three years of platform code behind it, deployed in the App Store today. The cryptography, the settlement, the wallet management, and the operational experience of running real Bitcoin SV transactions are not unknowns that the customer is being asked to risk capital on. They are facts of the existing Henceforth product, observable today.

5.3 Brand and distribution

Scripted Supply ships as a distinct application on the App Store — a separate bundle identifier, a separate App Store listing, a separate brand identity within the Henceforth family — but draws on the same underlying code base, the same wallet infrastructure, the same operational maturity. This is the same product-line architecture that, for example, an enterprise software vendor uses when shipping a B2B procurement product alongside a consumer wallet: shared technical foundation, distinct go-to-market.

6 Economic Model: Fees and Savings

The architecture is interesting; the economics are what the buyer decides on. This section decomposes the cost of running EDI today, decomposes the cost of running Scripted Supply, and presents a worked example for a representative mid-market customer. The numbers are illustrative and the goal is order-of-magnitude credibility, not accounting precision. Real engagement requires a customer-specific decomposition, which is exactly the conversation a pilot enables (Section 8).

6.1 Decomposing the legacy EDI cost stack

The total cost of EDI to a buyer is the sum of five line items, only two of which appear on the invoices the company receives. The other three are absorbed into salaries, professional fees, and overhead, which is why the total cost is consistently underestimated.

Value-Added Network (VAN) fees. The per-kilocharacter or per-transaction charges levied by the messaging intermediaries (TrueCommerce, SPS Commerce, Loren Data, OpenText, etc.) that route EDI documents between trading partners. Typical industry rates run $0.05–$0.50 per transaction, depending on contract terms and document size.

Software licence and integration. The cost of the EDI translator software that maps each trading partner's EDI dialect into the company's enterprise resource planning system. New partner integrations typically cost $10,000–$50,000 each in professional services, amortised over the life of the relationship.

Reconciliation labour. One full-time-equivalent employee in accounts payable typically processes 800–1,200 purchase orders per month, with a fully-loaded cost of $5,000–$8,000 per FTE per month in mid-market settings. The work consists of matching purchase orders to receipts to invoices to payments, resolving the inevitable disagreements between them.

Dispute resolution time. When the four artefacts disagree, somebody must figure out which one is right. Industry data suggests 3–5% of B2B invoices involve a dispute; resolution averages 2–4 hours of staff time per dispute, plus occasional escalation to legal.

Audit preparation. Annual external audit costs include preparing reconciliations between EDI records and physical receipts. The marginal cost of EDI-related audit work, separated from the rest of the financial audit, is typically $5,000–$15,000 per year for a mid-market company.

Industry estimates of the total cost run between five and ten percent of transaction value. For a company handling 1,000 POs per month at an average PO value of $10,000, gross transaction value is $120 million per year and reconciliation cost is therefore in the range of $6 million to $12 million per year. The illustrative number used below is more conservative: only the operational costs are counted, not the value at risk from undetected reconciliation errors.

6.2 The Scripted Supply cost stack

Scripted Supply replaces most of these line items with chain fees and a SaaS subscription:

Chain fees. Bitcoin SV transaction fees are approximately $0.01–$0.05 per purchase order, including the PO issuance, the supplier acceptance, the delivery attestation, and the atomic fulfilment transaction. For a customer handling 1,000 POs per month, total monthly chain fees are approximately $30.

Scripted Supply subscription. A flat monthly subscription covering platform access, the supplier web portal, the indexer, and the relay infrastructure. The illustrative figure used below is $99 per month for the small tier (up to 500 POs/month) with volume-tiered pricing above that. Final pricing is part of the pilot conversation; the figure here is a working hypothesis, not a commitment.

Reconciliation labour. Approaches zero in steady state. The canonical state of every purchase order is the on-chain UTXO; there is no separate "my records vs. your records" to reconcile against. The indexer surfaces every state transition into the operator's existing accounting system via a simple webhook.

Dispute resolution. Reduced, not eliminated. Disputes still happen — a parcel may be visibly damaged, a delivery quantity may be short — but they are litigated against a cryptographic audit trail rather than against incomplete and inconsistent records. Resolution time falls from hours to minutes for most cases.

Audit preparation. Substantially reduced. The on-chain transition log is the audit trail. An external auditor can verify every purchase order independently by querying the chain, without requesting documents from the company.

6.3 Worked example: 1,000 POs per month

Table 1 shows the cost decomposition for a representative mid-market customer handling 1,000 purchase orders per month.

Table 1: Cost comparison for a representative 1,000 PO/month customer. All figures in USD per month, rounded.

Cost category	Detail	Legacy EDI	Scripted Supply	Savings
VAN / messaging fees	$0.20/tx	$200	$0	$200
Software licence & integration	amortised, multi-partner	$1,500	included	$1,500
Reconciliation labour	1 FTE @ $5,000	$5,000	$0	$5,000
Dispute resolution	3% disputes, 3h avg	$500	$100	$400
Audit preparation	amortised annual	$500	$100	$400
BSV chain fees	$0.03 per PO	–	$30	–
Scripted Supply subscription	small tier	–	$99	–
Monthly total		**$7,700**	**$329**	**$7,371**
Annualised		**$92,400**	**$3,948**	**$88,452**

The headline saving is **approximately $88,000 per year** for a customer of this size. The figure is dominated by reconciliation labour, which is also the largest single component of legacy EDI cost and the line item that on-chain settlement reduces most aggressively.

6.4 Sensitivity across volume

Different customers run very different PO volumes, and the savings calculation is not linear. Table 2 shows annualised savings across a representative range.

Table 2: Annualised savings across volume tiers. All figures in USD per year, rounded.

Monthly POs	Legacy EDI	Scripted Supply	Annual savings
100	$14,400	$2,556	$11,844
500	$48,000	$3,372	$44,628
1,000	$92,400	$3,948	$88,452
5,000	$408,000	$8,388	$399,612
10,000	$816,000	$13,788	$802,212

The savings curve is approximately linear in volume because the dominant variable cost in legacy EDI — reconciliation labour — scales linearly with PO count. Customers above 5,000 POs/month should expect six-figure annual savings; customers above 10,000 should expect seven-figure savings.

6.5 What this analysis does not include

Three categories of additional value are deliberately excluded from the headline numbers, because quantifying them requires customer-specific data:

Value at risk from reconciliation errors. Undetected disagreements between purchase orders, receipts, and invoices result in real losses — overpayments, double payments, payment for goods never received. Industry studies suggest 0.5–2% of transaction value, which would dwarf the operational savings calculated above for high-volume buyers.

Working capital efficiency. Faster settlement and faster dispute resolution mean cash spends less time tied up in unresolved trades. For a mid-market company doing $120 million in annual procurement, a one-week reduction in days-payable-outstanding cycle frees roughly $2.3 million of working capital.

Supplier relationship value. Suppliers paid faster, with cryptographic proof of delivery acceptance, are more willing to offer favourable credit terms and price concessions. This is real but unmeasurable from outside.

> **Honest caveat.** All figures in this section are illustrative and based on industry-standard estimates of legacy EDI costs. The actual savings any specific customer realises depend on their actual EDI cost decomposition, their actual purchase order volume and value distribution, their willingness to migrate existing trading partners onto the asymmetric web portal, and the pricing of the Scripted Supply subscription tier they ultimately negotiate. The defensible claim made by this paper is that the order of magnitude is correct: a mid-market customer can credibly expect five-figure annual savings, and a large-enterprise customer can credibly expect six- or seven-figure annual savings, with a payback period of less than one quarter.

7 Privacy and Confidentiality

Supply chain data is competitively sensitive: who you buy from, what volumes, at what prices, on what cadence. A public blockchain that exposes this graph is unusable

for serious B2B procurement. Scripted Supply's privacy model is designed for that constraint from the first line of code, not retrofitted after a customer complaint.

7.1 Per-PO Type42 derivation

Scripted Supply uses BRC-42 (Type42) invoice-number-based key derivation [2] to produce a fresh child key for every (counterparty pair, purchase order) tuple. Two purchase orders from the same buyer to the same supplier use distinct on-chain public keys. A passive observer querying the chain cannot link those two POs without holding one of the two relationship private keys. The unlinkability property is not a matter of operational hygiene — the kind that fails the first time a developer accidentally reuses an address — but a structural consequence of ECDH's computational hardness.

The derivation hierarchy is four levels deep, and each level is a pure function of the level above it:

1. **BAP root identity** per legal entity, anchored off-chain.
2. **Per-role child key** (office, warehouse, customer) derived from the BAP root using a fixed invoice number per role.
3. **Per-relationship child key** derived from the role key using the counterparty's BAP identity as the invoice-number seed; lexicographic sort of the two BAP identifiers prevents the two sides deriving different roots.
4. **Per-PO child key** derived from the relationship key using the PO identifier as the invoice number.

The leaf is the only key that appears on chain. Nothing stored persistently links a public key to its parent without one of the private keys in the chain — which makes the storage discipline question moot.

7.2 Off-chain encrypted payloads

The body of a purchase order — line items, quantities, prices, delivery addresses, custom terms — is never stored on chain. It is encrypted client-side via BRC-2 ECIES [5] (ECDH + AES-128-CBC + HMAC-SHA256) under a per-PO session key. The encrypted blob lives in the operator's relay storage or any equivalent off-chain location.

For multi-party readability, the 32-byte session key is envelope-encrypted to each authorised reader's pubkey — office, warehouse, customer, and optional fourth parties (auditor, customs broker, regulator). Adding a fourth-party reader after the fact is a re-encrypt of the 32-byte session key only, not a re-encrypt of the (potentially large) payload. The cost of inviting an auditor into an existing PO is therefore proportional to the number of authorised readers, not to the size of the data.

7.3 On-chain commitment

The only on-chain trace of the encrypted payload is a SHA-256 hash committed in the PO's OP_RETURN output, plus a URL hint indicating where the ciphertext can be fetched. The hash is the tamper-evidence: every reader independently verifies that

the off-chain payload they decrypted matches the on-chain commitment, and refuses to act on it otherwise. The URL is a convenience, not an authority — if the relay disappears, any reader holding the ciphertext can republish it and the on-chain hash still validates.

7.4 What leaks unavoidably

Three categories of metadata cannot be hidden by this design and are honestly named here rather than discovered by a customer later:

Transaction timing. The chain records when each on-chain state transition was broadcast and confirmed. An adversary observing the chain can therefore infer when POs are issued and when they fulfil, even without being able to identify the parties or read the contents. Mitigation: batch multiple POs into a single transaction where commercial timing allows, blurring per-PO timing within the batch.

Rough value bands. The customer's payment output sits on chain in cleartext satoshis (BSV's native value-bearing unit). An observer can therefore infer the rough order of magnitude of each PO. Mitigation: route payments through a Paymail P2P intermediary so the on-chain settlement nets several POs against a single payment leg.

Counterparty graph if Paymail handles are reused. If the same Paymail handle (e.g. procurement@acme.com) resolves to the same on-chain receive address across POs, the address graph leaks counterparty identity. Mitigation: the Paymail capability paymail-with-bsv-type42 returns a fresh per-PO derived address per resolution, eliminating address reuse at the Paymail layer.

7.5 Compliance posture

A privacy-preserving architecture has a useful side effect: it makes regulatory compliance dramatically easier. The General Data Protection Regulation's right-to-erasure obligation is famously awkward on conventional blockchain systems, because the chain is immutable. Scripted Supply resolves this by storing all personally-identifiable and commercially-sensitive data off chain. Erasing the off-chain ciphertext renders the on-chain hash a stranded commitment to nothing recoverable; the right-to-erasure is honoured in fact even though the chain is unchanged. The on-chain trace records that a transaction happened, not what it was about.

The same property applies for export-controlled goods, healthcare data under HIPAA-equivalent regimes, and any other regulatory surface where the data class is sensitive but the existence of the transaction is not.

8 Pilot Program

Scripted Supply is looking for an anchor customer. This section is the ask, the terms, and the timeline.

8.1 Who this is for

The right anchor customer is a mid-market business currently running EDI-based procurement, with at least one trading-partner relationship willing to migrate to a new platform alongside the existing one. They do not need to be cryptography-literate; their suppliers especially do not. The pilot succeeds if at the end of three months the customer can demonstrate, on a real production purchase-order channel, the operational savings claimed in Section 6.

8.2 The two-sided commitment

Scripted Supply commits to: An eight-week implementation from contract signature to a working bilateral channel: the operator app deployed to the customer's iOS/macOS workstation, the supplier web portal hosted on a public subdomain, the Forth contract drafted and reviewed with the customer's legal team, and the first three production POs broadcast on Bitcoin SV mainnet.

The customer commits to: One specific trading-partner relationship migrated onto Scripted Supply within the eight-week window. A weekly thirty-minute check-in with Henry Hudson. Willingness to share the customer's actual EDI cost decomposition for the ROI verification.

8.3 Pilot terms

> **Three months free, paid thereafter.**
> For the first three calendar months from the customer's first production purchase order, the Scripted Supply subscription is zero. The customer pays only Bitcoin SV chain fees (approximately $0.03 per PO; typically $10–$50 per month at pilot volume). After the three-month period, the subscription transitions to $99 per month for the small tier (up to 500 POs/month), with volume-tiered pricing negotiated above that threshold.

8.4 The ROI gate

At the end of month three, the customer and Scripted Supply jointly compute the realised operational savings against the projection in Section 6. The customer is entitled to exit the engagement at no cost — including a refund of accumulated chain fees — if measured savings fall below seventy-five percent of the projected savings for their PO volume tier.

The gate is symmetrical and not subjective. It is computed against the customer's pre-pilot cost decomposition (reconciliation hours saved, dispute resolution time reduced, audit-prep hours avoided), measured at the end of month three. If the architecture does not produce its claimed savings, the customer leaves whole. Scripted Supply takes the risk of the architecture; the customer takes the risk of internal change management.

8.5 Implementation timeline

Week	Deliverable
1–2	Onboarding workshop: current EDI cost decomposition; trading-partner selection; commercial-terms drafting.
3–4	Forth contract authoring and customer-legal review; dry-run execution against mock counterparty. Operator app installed on customer workstation.
5–6	First production purchase orders broadcast on Bitcoin SV mainnet. Supplier receives email link, opens portal, signs acceptance.
7	First atomic delivery-versus-payment cycle: supplier signs delivery attestation, customer scan-preimage submitted, payment moves on chain.
8	Audit replay validation: customer's auditor independently verifies the on-chain state transitions for the pilot purchase orders. Handoff to customer operations team.
9–12	Steady-state operation under monitoring. Weekly check-ins.
End of month 3	ROI verification against pre-pilot baseline. Decision to convert to paid subscription or exit.

8.6 In scope and out of scope

In scope for the pilot: one trading-partner relationship, full PO/acceptance/delivery/payment lifecycle, Forth contract authoring and training for the customer's procurement team, supplier portal access for the partner's warehouse and accounting staff, weekly architectural reviews.

Out of scope for the pilot: integration with the customer's existing ERP system (a Phase 2 deliverable, post-pilot), multi-partner deployment, white-label branding of the supplier portal, custom Forth-vocabulary extensions specific to the customer's industry.

8.7 Next steps

The route from this whitepaper to a signed pilot is a thirty-minute discovery call. Email henry@henceforth.club or open the link at henceforth.club/scriptedsupply. Initial calls are returned within two working days; pilot decisions are typically reached within two weeks of the discovery call.

About Henry Hudson and Henceforth

Henceforth is a Bitcoin SV wallet and scripting platform for iOS and macOS, in active development since 2022. It is built and operated by Henry Hudson, an iOS engineer with a decade of experience building production Swift applications and a multi-year focus on Bitcoin protocol work. Henceforth has been live on the Apple App Store since 2024.

The technical foundation

The Henceforth platform comprises three components that Scripted Supply builds on directly:

- **Henceforth app** (henceforth.club): the iOS and macOS application that houses both the consumer Bitcoin SV wallet and — as a new product target on the same code base — the Scripted Supply operator interface. Approximately 60,000 lines of Swift, written under modern Swift practices (Swift 6 strict concurrency, value-typed wallet primitives, structured concurrency throughout).
- **SwiftBSV** (github.com/henryhudson/SwiftBSV): a pure-Swift Bitcoin SV SDK. Used as the cryptographic foundation of Henceforth and shipped publicly as a Swift Package for any iOS or macOS application that needs Bitcoin keys, transactions, scripts, or SPV verification. Includes BIP-32 hierarchical-deterministic key management, BRC-42 Type42 derivation (wire-compatible with @bsv/sdk), BRC-2 ECIES encryption, full Bitcoin SV opcode coverage, BEEF parsing, and Merkle-proof SPV verification in three formats.
- **The Henceforth Forth interpreter**: a Forth-2012 compliant interpreter with Bitcoin-specific extensions for transaction construction, script primitives, and on-chain data composition. The Scripted Supply contract language is built on this interpreter, extending its vocabulary with EDI-specific words.

Why this team for this project

Scripted Supply requires a builder who is fluent in both Bitcoin SV at the protocol level and modern iOS/macOS development at the platform level. The intersection of those skills is small. Henry Hudson has been working in it for three years; the artefacts above are public proof.

Equally important, Scripted Supply requires a builder who has shipped a production application to the Apple App Store and operates real Bitcoin SV infrastructure today — not a research prototype, but a live wallet handling real value. Henceforth is that artefact. Scripted Supply enters the customer's evaluation cycle with a working, deployed, monitored platform underneath it.

Contact

- Email: henry@henceforth.club
- Web: henceforth.club
- Code: github.com/henryhudson
- Pilot enquiries: henceforth.club/scriptedsupply

9 References

[1] Craig S. Wright. Scripted Supply: A Bitcoin-Based Architecture. The architectural framing on which Scripted Supply is built. The four-path UTXO state-token model, the atomic delivery-versus-payment construction, and the “well-formed lies” framing of EDI failure all originate here. Quoted throughout. Craig’s Substack. June 10, 2025. URL: https://singulargrit.substack.com/p/scripted-supply-a-bitcoin-based-architecture (visited on 05/11/2026).

[2] Ty Everett. BRC-42: BSV Key Derivation Scheme. The Type42 invoice-number-based child key derivation used for per-purchase-order privacy. Bitcoin SV Specifications. 2023. URL: https://github.com/bitcoin-sv/BRCs/blob/master/key-derivation/0042.md (visited on 05/11/2026).

[3] Bitcoin Attestation Protocol Working Group. BAP: Bitcoin Attestation Protocol. Identity layer on Bitcoin SV; the off-chain anchor for organisation identity. 2020. URL: https://github.com/icellan/bap (visited on 05/11/2026).

[4] nChain and BSV Association. Paymail Specification. Email-shape address resolution layer used for actor discovery. 2019. URL: https://bsvalias.org (visited on 05/11/2026).

[5] Ty Everett. BRC-2: Data Encryption and Decryption. The ECIES encryption scheme used for encrypted off-chain payloads. Bitcoin SV Specifications. 2023. URL: https://github.com/bitcoin-sv/BRCs/blob/master/wallet/0002.md (visited on 05/11/2026).