

VERIFIED BITCOIN

vBTC V2 Technical Specification

Revised edition · VerifiedX Core 9.0

October 2026 · for integrators and institutional reviewers

**REVISED EDITION**

This edition replaces the vBTC V2 Technical Specification withdrawn in September 2026. The protocol has changed since that edition: Bitcoin attestation, watched vaults, chain-fixed withdrawals, the validator bond and the hardened key ceremony are new in Core 9.0. Where the two editions differ, this one describes Core 9.0.

ABSTRACT

vBTC V2 represents Bitcoin on the VerifiedX network one to one. Each vault is a smart contract with its own Taproot address, whose key is created by a FROST distributed key generation among bonded validators and never exists whole. From Core 9.0, VerifiedX consensus verifies Bitcoin directly: validators submit Bitcoin headers and Merkle-proved vault transactions, and every node checks proof of work, burial depth and inclusion before any balance changes. Withdrawals are paid by a Bitcoin transaction that consensus computes when the request is mined; the vault's signers sign exactly that transaction, and Bitcoin's confirmation completes it. This document specifies the roles, rules, parameters and trust assumptions of that design.

APPLIES TO

VerifiedX Core 9.0 and 9.0.1

NETWORKSMainnet (Bitcoin mainnet)
Testnet (VerifiedX signet)**CRYPTOGRAPHY**FROST on secp256k1
BIP340 Schnorr, Taproot**COMPANION DOCS**docs.verifiedx.io
vBTC · vBTC Raw Integration

Contents

1	Introduction scope, design principles	3
2	System model roles, layers, trust boundaries	3
3	Validators, the bond and signer sets	4
4	Vault creation the key ceremony and its on-chain binding	5
5	Bitcoin attestation headers, reports, inclusion proofs, reorgs	6
6	Vault accounting and transfers	8
7	Withdrawals request, pin, signing, settlement, controls	8
8	Key maintenance health, re-keying, sweeps	11
9	S3C: Self-Custodial Smart Contracts	11
10	Transaction types	12
11	Protocol parameters and network heights	12
12	Security properties and assumptions	13
13	Earlier vaults	14
14	References	14

What changed from the withdrawn edition

TOPIC	WITHDRAWN EDITION	THIS EDITION (CORE 9.0)
Vault balances	Owner's deposit read from Bitcoin servers (Electrum).	The vault's Bitcoin as consensus records it from validator reports that every node verifies (§5, §6).
Withdrawal completion	Requester completes with a FROST signing session and a completion transaction (type 28).	Consensus pins the Bitcoin transaction when the request is mined; signers sign it; Bitcoin's confirmation completes it. Type 28 is refused for these vaults (§7).
Cancellation	Validator vote, 75% to approve (type 30).	Requester cancels at once while nothing is pinned; no vote. After the pin: raise the fee or release (§7.6).
Key ceremony ID	A separate session GUID.	The vault's contract UID, <code><32 hex>:<unix time></code> (§4.1).
DKG shares	Relayed by the coordinator in plaintext.	Sealed to each recipient's registered key; plaintext refused (§4.4).
Deposit address binding	Written by the creator; unsigned proof.	Must be the Taproot address of a group key every participant attested and proved possession of (§4.5).
Ceremony participation	75% of active validators.	At least 90% of eligible (bonded, aged) validators; re-keys 67% (§4.2, §8).
Threshold relaxation	Dynamic threshold with a 24-hour safety gate.	A key's signing threshold is fixed when the key is made. Lost holders are handled by re-keying the vault (§8).
Validator stake	5,000 VFX checked at registration; liquid.	5,000 VFX bonded while registered; unlocks 1,200 blocks after exit; two days of bond before a validator counts (§3).
Bitcoin testnet	Bitcoin testnet4.	The VerifiedX signet, a private Bitcoin signet (§11).

CONVENTIONS

“Must” states a rule consensus enforces: a transaction or block that breaks it is invalid on every node. Amounts are in BTC with at most 8 decimals unless given in sats. Heights are VerifiedX (VFX) block heights unless marked as Bitcoin heights. Route names refer to the node API under /vbtapi/VBTC.

1 Introduction

vBTC is Bitcoin on VerifiedX, one to one. A vBTC balance is a ledger claim on Bitcoin held at a specific vault's Taproot address. It is not a derivative, a debt instrument or an algorithmic representation, and nothing in it tracks a price.

vBTC V2 is designed around a separation of four functions. **Bitcoin establishes transaction truth:** deposits and payouts are Bitcoin transactions, counted only once buried under proof of work. **VerifiedX consensus verifies and enforces that state:** every node checks the evidence and applies it to balances by the same rules. **Asset owners retain authorization:** only a holder can withdraw its vBTC, and only the requester can cancel, re-price or release its withdrawal. **FROST threshold signing executes** the one Bitcoin transaction the protocol permits for each withdrawal.

The design aims for these properties, each specified in the sections that follow:

- **No whole key.** A vault's Bitcoin key is generated in shares and is never assembled, by any participant or by the vault's owner.
- **Verified, not reported.** Balances follow Bitcoin events that each node verifies from headers and Merkle proofs, not a server's or a block producer's statement.
- **Payouts fixed by rule.** The Bitcoin transaction that pays a withdrawal is a deterministic function of chain state, computed by every node before anyone signs.
- **No discretionary votes.** No completion transaction and no cancellation vote exist for watched vaults. Every outcome follows from what the requester sent and what Bitcoin shows.
- **Keys that outlive their holders.** When too many key holders leave the validator set, the network replaces the vault's key while the old one can still sign.

Out of scope: the Base bridge (vBTC.b, its own specification), the shielded vBTC pool, and wallet user interfaces. Vaults created before Core 9.0's attestation rules are covered only in §13.

2 System model

2.1 Roles

ROLE	DEFINITION
Owner	The VFX address that owns a vault. Starts its key ceremony and signs its creation. Bitcoin deposited to the vault is credited to the owner. Ownership can be transferred.
Holder	Any VFX address with a vBTC balance on a vault. Holders transfer vBTC and withdraw their own balance to Bitcoin.
Validator	A registered VerifiedX validator with a 5,000 VFX bond. Produces blocks; may take part in key ceremonies and sign Bitcoin reports when eligible (§3).
Key holder	A validator holding a share of a vault's key. A threshold of a key's holders signs that key's Bitcoin transactions.
Signer set	The validators whose signatures count on Bitcoin reports during one 600-block epoch (§3.3).
Every node	Any VerifiedX node, validator or not. Verifies every report, creation, request and pin from chain state alone.

2.2 Layers

Owners and holders

Sign VerifiedX transactions with their own keys: create a vault, transfer, request, cancel, raise a fee, release. Nothing else can debit their vBTC.

AUTHORIZATION

Requests

Signed VFX transactions types 25 to 29 and 44.

VerifiedX consensus

Holds the attested Bitcoin header chain, each watched vault's coins, the vBTC ledger and every open withdrawal with its pinned Bitcoin transaction. All of it is chain state, identical on every node.

VERIFICATION AND ENFORCEMENT

Reports and rekeys

Sender-less types 43, 45, 46, authorized by the validator signatures inside them.

Validators

Run a Bitcoin light client, build and sign Bitcoin reports, take part in key ceremonies, hold key shares and sign pinned transactions over the FROST network protocol (ports 7295 and 7296).

OBSERVATION AND EXECUTION

Pins

Computed in block application; no transaction carries them.

Bitcoin

Each vault's coins sit at its Taproot address. Deposits, payouts and sweeps are ordinary Bitcoin transactions; key-path spends carry a 64-byte Schnorr signature.

TRANSACTION TRUTH

Bitcoin

Signed pinned transactions, broadcast by the signers.

2.3 Trust boundaries

The protocol separates what each node verifies for itself from what rests on a quorum. Reviewers should read the security properties in §12 against this table.

ESTABLISHED BY	FACTS
Every node, from the evidence	That reported headers form a valid Bitcoin chain from the checkpoint (linkage, required target, proof of work, median time); that an event's block is buried to the required depth; that each event transaction is in its block (Merkle branch); what each transaction does to vault coins; that a vault's address is the Taproot address of an attested key; which Bitcoin transaction pays each withdrawal.
Two thirds of the epoch's signer set	Which Bitcoin branch the chain follows among valid ones; that blocks passed over held no vault event (completeness); when a reported event is taken back after a Bitcoin reorganisation.
A threshold of a key's holders	That the pinned transaction is signed and broadcast (liveness). Honest holders sign only the pinned transaction; a colluding threshold holds the key's full signing power.
Bitcoin proof of work	That a buried block stays in Bitcoin's chain. Six blocks on mainnet, three on the testnet signet.

3 Validators, the bond and signer sets

3.1 Registration and bond

A validator registers its VFX address, IP address and FROST public key on chain. The registered FROST public key must be the uncompressed public key (04 and 128 hex digits) that derives the validator's own VFX address; key-ceremony shares are sealed to it (§4.4).

From the bond height, a mined registration or heartbeat **bonds** the validator's 5,000 VFX: no transaction from that address may take its balance below 5,000 VFX, at admission, in a block, or across the address's transactions in one block. A mined exit transaction starts a wait of 1,200 blocks (two signer epochs), after which the stake unlocks. Going offline unlocks nothing. There is no slashing.

3.2 Eligibility

A validator is **eligible** for a key ceremony, and for a place in a signer set, when it is active and its stake has been bonded for at least 14,400 blocks (two days at 12 seconds a block). Validators already active at a release-fixed grandfather height that held the stake when bonds began count as aged at once.

3.3 Signer sets

At the first block of every **600-block epoch** (about two hours), each node stores the epoch's signer set: the validators active at the previous block whose bond is aged. A Bitcoin report is valid with signatures from distinct members of the signer set of the current or the previous epoch, each a valid VFX signature over the report hash, numbering at least $\max(3, \lceil 2n/3 \rceil)$ for a set of n .

4 Vault creation

A vault is created in two parts: a FROST distributed key generation (DKG) among eligible validators, authorized by the owner, and a contract creation transaction (type 25) that binds the result on chain.

4.1 Ceremony identifier

The owner's node starts the ceremony (`InitiateMPPCeremony`, or `PrepareMPPCeremonyRaw` and `ExecuteMPPCeremonyRaw` for an owner whose key is outside the node). The ceremony identifier is the new vault's **contract UID**, `<32 lowercase hex>:<unix time>`, so every attestation names the vault it creates. An owner can run one ceremony at a time; a node runs at most 100. Status moves through `Initiated`, `ValidatingValidators`, `Round1InProgress`, `Round2InProgress`, `Round3InProgress` to `Completed`, or ends `Failed` or `TimedOut` (after one hour).

4.2 Participants and threshold

Public vault participants	At least 90% of eligible validators, and never fewer than 3; at most 512.
S3C vault participants	Every validator in the owner's configured pool, at least 3 (§9).
Signing threshold	$\lceil \text{participants} \times 51\% \rceil$: a majority of the key's holders.
Dropped validators	A participant that fails a round is dropped and the ceremony reruns without it under a retry session the owner pre-signed, if enough participants remain.

4.3 Rounds

- Round 1, commitments.** Each participant generates its secret polynomial and broadcasts its commitments. It also publishes its constant term (a 33-byte compressed point) with a BIP340 signature, made with that term's secret, over a possession message naming this vault's UID, owner and group key.
- Round 2, shares.** Each participant evaluates its polynomial for every other participant and seals each share to the recipient (§4.4). The coordinator relays ciphertext only.
- Round 3, verification.** Each participant opens its shares, verifies each against its dealer's Round 1 commitment, derives its signing share and the public key package (the group key and every holder's verifying share), and signs an attestation (§4.5). A validator signs only for a ceremony it took part in and whose key package it stored.

The ciphersuite is FROST-secp256k1-SHA256-TR (the Zcash Foundation's `frost-secp256k1-tr`), producing BIP340 Schnorr signatures.

4.4 Share sealing

Each Round 2 package is sealed by its dealer to the recipient's registered FROST public key (§3.1) with ECIES over secp256k1: an ephemeral ECDH exchange, a SHA-256 key derivation over a fixed domain string, the shared point, the ephemeral key and the recipient key, and AES-256-GCM with the session ID and recipient identifier as associated data. A recipient accepts only packages sealed to it and refuses plaintext. A participant whose registered key does not derive its address is left out before the ceremony starts. The dealer keeps each package's one-time key, which settles a recipient's signed complaint about a bad share without revealing anything else.

4.5 Attestation and on-chain binding

Every participant signs, with its validator key, the message:

```
VFX_DKG_ATTEST_V2|<contract UID>|<group public key>|<Taproot address>|<owner>|<threshold>|<sorted participants>[|REKEY|<vault UID>]|PKP|<key package hash>
```

The key package hash is the SHA-256 of the public key package the participant finished with. Participants given different Round 1 commitments by a dishonest coordinator finish with different packages, so their attestations cannot all match. The proof (type `DKG_ATTESTED_V2`) carries every attestation and every participant's constant term with its possession signature.

A contract creation (type 25) carrying a vBTC V2 feature is valid only if:

- its deposit address is the Taproot address of its FROST group key;
- every participant signed the attestation, and the participants' constant terms sum to the group key, each with a valid possession signature for this vault;
- every participant is an active validator of the vault's class (public or S3C) holding the validator balance, and a public ceremony includes at least 90% of the eligible validators;
- the threshold is at least a majority of the participants, and the transaction's sender is the owner named in the attestations.

Because the proof grows with the participant count, vault creation transactions may be up to 256 KB; other transactions keep the 30 KB limit.

4.6 Deposit address

The deposit address is a Taproot (SegWit v1) output whose key is the **raw x-only FROST group key**, with no BIP341 tweak. The FROST signer signs against the untweaked key, and spends use the key path with a 64-byte signature in the witness. A vault whose address duplicates another vault's address is not registered and holds no coins.

The address is usable once the creation transaction is in a block. `GetMPCDepositAddress` serves it only then, and any node answers it from chain state.

INTEGRATOR RULE

Do not deposit before `GetMPCDepositAddress` returns the address. Bitcoin sent to the address of a creation that never reaches the chain has no vault and cannot be withdrawn.

4.7 Creation sequence (node API)

#	CALL	RESULT
1	<code>POST InitiateMPCCeremony/{owner}</code>	<code>CeremonyId</code> (the contract UID), status <code>Initiated</code>
2	<code>GET GetCeremonyStatus/{id}</code>	Poll to <code>Completed</code> : group key, DKG proof, threshold, <code>ContractUID</code>
3	<code>POST CreateVBTCContract</code>	Signed and broadcast type 25; <code>DepositAddressConfirmed: false</code>
4	<code>GET GetMPCDepositAddress/{uid}</code>	The address, once the creation is in a block

5 Bitcoin attestation

From the **recording height**, VerifiedX holds its own view of Bitcoin in chain state: a header chain from a fixed checkpoint, and the coins of every watched vault. Validators extend it with Bitcoin reports; every node verifies each report from the report and chain state alone, using the same function at mempool admission, in block validation and when the block is applied.

5.1 Checkpoint

The view starts from twelve consecutive Bitcoin headers set in the release, ending at the first block of a difficulty interval (a height divisible by 2,016): Bitcoin height 969,696 on mainnet and 2,016 on the testnet signet. Nothing about the checkpoint is trusted except its bytes: the headers must link and each must meet its own proof-of-work target. Twelve headers cover the median-time rule for the first header after it.

5.2 Header rules

A header is accepted into the attested chain only if it extends the stored chain and:

- names the previous header's hash;
- carries exactly the target Bitcoin's difficulty rules require at its height (2,016-block intervals, two-week target timespan);
- has a hash at or below its own target (proof of work);
- is timed after the median of the eleven headers before it.

The rule that a header may not be more than two hours in the future is deliberately not checked: it needs a local clock, which consensus cannot use, and an event counts only once buried under further proof of work. On a network with Bitcoin's testnet minimum-difficulty rule, a minimum-difficulty header is followed but does not count toward burial.

5.3 Burial

A Bitcoin block's events count only when **6 counted headers** sit on top of it on mainnet, **3** on the testnet signet. Validators report an event as soon as it is that deep, so a deposit is normally credited one VFX block after the burying Bitcoin block.

5.4 How validators observe Bitcoin

Each signing validator runs a built-in Bitcoin light client: it finds peers through Bitcoin's DNS seeds, keeps eight outbound connections spread across different networks, follows the heaviest valid header chain from the checkpoint using the same header rules, and fetches recent blocks without witness data. A fetched block must carry its validated header, start with its only coinbase transaction, and hash to the header's Merkle root. Validators also exchange headers with one another; public block explorers are asked only after ninety minutes without a new header, and are checked like any other source. On the testnet signet each block must also carry the signet's challenge signature (BIP 325). Nodes that do not sign reports run none of this: they judge each report from the report itself.

5.5 Reports (type 43)

A Bitcoin report is a sender-less transaction (`From` and `To` are `BTC_Attestation`, no amount, no fee) whose authority is the signatures inside it. A forward report is about one Bitcoin block and carries:

Headers	Up to 500 new headers, oldest first, extending the attested tip through the header that buries the block.
Coinbase proof	The block's coinbase transaction (up to 10,000 bytes) and its Merkle branch; for a larger coinbase, a constant-size form (SHA-256 midstate, final bytes and length) showing the first leaf is longer than 64 bytes.
Events	Up to 2,000 vault transactions, each with its position, its serialization without witness data, and its Merkle branch.
Parts	A block with more events is reported in numbered parts applied in order; the last is marked final.
Binding	The block hash, the signer epoch and the validator signatures over the report hash. A report transaction is at most 512 KB.

5.6 Inclusion proofs

Nothing a report says about a transaction is taken on the signers' word. The verifier recomputes each transaction's ID as the double SHA-256 of exactly the bytes carried, so what is parsed and what is hashed cannot differ, and checks its Merkle branch against the header. Two further rules close known ways to fool a proof checker: every event's branch must have the same length as the coinbase's branch, because all leaves of Bitcoin's tree sit at one depth; and a transaction of 64 bytes or fewer is never accepted. Together these stop an inner node of the tree from being passed off as a transaction.

The effects of each event are derived by the verifier: an output paying a watched vault's deposit script is a new coin, and an input spending a recorded coin is a spend. A transaction too large to carry is reported as a spend-only event; the recorded coins it names are **frozen**, never treated as spent on the signers' word.

5.7 Other report forms

Quiet report	When the next 144 Bitcoin blocks hold no vault event, the signers say so once. The range is fixed by state, so every validator builds the same report.
Backfill report	Scoped to one vault: a deposit made in the 144 already-reported blocks before the vault's creation was mined.
Header replacement	A Bitcoin reorganisation of blocks with no reported events. The replacing headers must carry more work than those they replace.
Bitcoin reorg (type 45)	Replaces headers below reported events and takes those events back. Signed like a report; the signed content marks it as a rewind, so one can never serve as the other.

5.8 What a valid report establishes

A report that passes verification establishes, for every node: that its headers are Bitcoin's (each extends the last and meets the required target); that the reported block is buried to the required depth; that each event transaction is in that block; and what each transaction does to vault coins. Which branch is followed, and that skipped blocks held nothing for any vault, rest on the signatures of two thirds of the epoch's signer set (§2.3).

6 Vault accounting and transfers

6.1 Watched vaults

A vault whose creation is applied at or after the recording height has its deposit script registered in state and is **watched**. Its Bitcoin, for every purpose, is the set of coins reports have recorded for it: no Bitcoin server is asked, and a block that spends more than the vault holds is invalid on every node. Coinbase outputs are credited but cannot be spent for 100 Bitcoin blocks.

6.2 Balances

Vault Bitcoin	Its unspent coins plus coins pinned to a withdrawal that has not settled (that withdrawal is already debited).
Holder balance	The sum of the holder's ledger entries on the vault.
Owner balance	The vault's Bitcoin plus the owner's ledger entries (negative for vBTC sent away), plus add-backs for Bitcoin that left the vault and was already charged on the ledger: settled withdrawals, settled bridge exits and release fees. The result equals the vault's Bitcoin less what every other address holds.

Deposits accrue to the owner. Holders are protected from the owner: the owner's figure is what remains after every other holder, so it can never spend vBTC already sent. A withdrawal's Bitcoin fee is taken from its own amount, so the vault never pays it. Former owners' negative positions count as zero, so every spendable figure on a vault adds up to its Bitcoin.

6.3 Transfers (type 26)

- **Single vault** (`TransferVBTCV2()`): vault, sender, recipient, amount. Checked against the sender's balance, including its transactions still waiting to be mined.
- **Multi-vault** (`TransferVBTCMultiV2()`): one transaction, one nonce, one fee, drawing on up to 25 vaults. Allocations must sum to the total.
- **Reserve senders** (`xRBX` addresses): the transfer is held for 24 hours and can be called back; it cannot go to another reserve address or use the multi-vault form. Reserve accounts cannot request Bitcoin withdrawals.

6.4 Ownership (type 18)

The owner moves the vault with a smart-contract transfer (`Transfer()`). The new owner takes over the vault's vBTC that no holder holds; vBTC already sent to holders stays theirs. A reserve owner's transfer takes effect after the reserve's unlock delay.

6.5 Reading state

Balances (`GetVBTCBalance` , `GetAllVBTCBalances`), vault Bitcoin (`GetVaultBitcoin`), withdrawals (`GetWithdrawalPin` , `GetWithdrawalRequestShares`) and the attestation view (`GetBitcoinAttestationStatus`) are answered from chain state, so any node gives the same answer for any address or vault.

7 Withdrawals

A withdrawal from a watched vault takes one transaction from its requester. Everything after it is a function of chain state and Bitcoin.

7.1 Request (type 27)

The request (`VBTCWithdrawalRequest()`) names the vault, the requester, the Bitcoin destination, the amount and a fee rate. It is valid only if:

- the fee rate is a whole number of sat/vB from 1 to 5,000;
- the destination is an address on the chain's Bitcoin network, with nothing around it, and is not the vault's own deposit address;
- after the smallest fee (154 vB at the rate), at least 546 sats remain to pay: the minimum is $154 \times \text{rate} + 546$ sats;
- the amount has at most 8 decimals and is within the requester's balance, together with its transactions waiting to be mined;
- the vault's current key can still sign (otherwise it is refused before anyone is debited, pending a re-key).

When the request is mined, its amount is debited from the requester at once. A holder may have several withdrawals open on one vault or many.

7.2 The pin

In the block that mines the request, every node computes the Bitcoin transaction that pays it from the vault's free coins, records it and holds those coins for it. Nothing in it is chosen:

Coins	The vault's unspent coins under its current key, excluding coinbase outputs younger than 100 blocks. The smallest single coin that covers the amount; otherwise the largest coins first until covered. Ties go to the older coin, then the lower output. At most 50 inputs.
Fee	Rate × size estimate: 57.5 vB per input plus 2 × 43 vB for outputs plus 10.5 vB, rounded up (154 vB for one input, 212 for two). Taken out of the amount.
Outputs	Output 0 pays the destination the amount less the fee. Output 1 returns the rest of the inputs to the vault's deposit script. A remainder under 546 sats is raised to 546 out of the payout, so nothing of the vault's goes to miners.
Form	Version 2, lock time 0, empty input scripts, every input sequence <code>0xFFFFFFFF</code> (replaceable under BIP 125).

A Taproot key-path signature sits in the witness and does not change the transaction ID, so the pinned txid is known before anyone signs. If the vault's free coins cannot pay the request yet, it waits and is tried again whenever the vault's coins change.

7.3 Signing and broadcast

The vault's signers build the same bytes from state and sign exactly the pinned transaction with a two-round FROST signing ceremony (nonce commitments, then signature shares, aggregated into one 64-byte Schnorr signature). They refuse to sign any other transaction for the withdrawal. Signers broadcast a pinned transaction on their own, typically within a few minutes. A holder that withheld its share in recent attempts is left out while the rest still meet the threshold.

7.4 Settlement

When a Bitcoin report shows one of the withdrawal's pinned transactions buried as deep as a deposit, the withdrawal is **Settled**. Nobody sends a completion transaction: Bitcoin is the completion. On watched vaults the node refuses a completion (type 28) and a cancellation vote (type 30).

7.5 States

STATE	MEANING	REQUESTER MAY
Waiting	Mined, but the vault's free coins cannot pay it yet. Retried as coins change.	Cancel
Pinned	Its Bitcoin transaction is fixed and its coins held. Signers sign and broadcast it.	Raise the fee, or release
Settled	Bitcoin shows a pinned transaction, buried. Paid.	
Cancelled	Cancelled while waiting; vBTC refunded at once.	
Expired	Not pinned within 360 VFX blocks of its request (later while it waits to take over coins held by a lower-rate withdrawal); vBTC refunded. Safe because nothing was signed.	
Void	A coin it held was spent by another transaction, so it can never confirm; vBTC refunded.	
Released	Its coins went back to the vault at the requester's wish; vBTC refunded less that transaction's fee.	

7.6 Requester controls

Cancel (type 29)	<code>VBTCWithdrawalCancel()</code> . Accepted only while the withdrawal is Waiting. Takes effect when mined; no vote. Once pinned, the transaction may already be signed, so cancel is refused.
Raise fee (type 44)	<code>VBTCWithdrawalFeeBump()</code> . The rate must exceed the newest transaction's and be at most 5,000. The chain pins a replacement on the same coins, in the same order, with the same outputs; the higher fee comes out of the payout. Up to 8 payment transactions per withdrawal. Whichever confirms pays it.
Release (type 44)	<code>VBTCWithdrawalRelease()</code> . The chain pins a transaction returning the withdrawal's coins to the vault. Its fee must be at least the newest transaction's fee plus 1 sat per vB of its own size (Bitcoin's replacement rule) and no more than the amount. Once buried, the vBTC is refunded less that fee. Up to 4 releases. If a payment confirms first, it is paid as usual.

7.7 Stuck payments and takeover

If none of a pinned withdrawal's payments has confirmed **2,160 VFX blocks** (about 7.2 hours) after its first pin, it no longer keeps its coins to itself: another withdrawal of the vault paying a higher fee rate may be pinned on them, with a fee large enough to replace the stuck transaction on Bitcoin. If that one confirms, the stuck withdrawal becomes Void and is refunded. A fee raise restarts the count only if it at least doubles the rate; a release never does. Waiting withdrawals are retried against coins that become takeable every 30 blocks.

7.8 Multi-vault withdrawals

One type 27 transaction (`VBTCWithdrawalRequestMultiV2()`) can draw a total from up to **10 vaults**. Each vault's portion is a share, pinned, signed and paid by its own Bitcoin transaction with its own fee, and every share must leave 546 sats after its fee. Shares settle independently, and each can be raised, released or cancelled with that vault's UID and the shared request hash. Allocation without explicit inputs takes the largest available balances first. A frozen earlier vault is never allocated.

7.9 Worked example

A holder requests 2,000 sats at 3 sat/vB from a vault holding one coin of 10,000 sats.

Minimum at 3 sat/vB	$154 \times 3 + 546 = 1,008$ sats: the request is valid
Coin chosen	The 10,000 - sat coin (the smallest single coin that covers 2,000)
Fee	$154 \text{ vB} \times 3 = 462$ sats
Output 0 (destination)	$2,000 - 462 = 1,538$ sats
Output 1 (back to the vault)	$10,000 - 2,000 = 8,000$ sats
Holder's vBTC	Debited 0.00002000 when the request is mined

7.10 Integration sequence

CALL	PURPOSE
<code>POST RequestWithdrawal</code> or the raw pair <code>GetRawRequestWithdrawalTxData /</code> <code>SendRawRequestWithdrawalTx</code>	Sends the request. Its hash is the withdrawal's <code>RequestHash</code> .
<code>GET GetWithdrawalPin/{uid}/{hash}</code>	State, pinned transactions, unsigned hex, and what the requester can do. Answered from the block that mines the request.
<code>POST CancelWithdrawal , BumpWithdrawalFee ,</code> <code>ReleaseWithdrawal</code>	Requester controls, each with a raw build and submit pair.

For raw signing, the node returns the unsigned transaction with its hash and hash preimage; the client checks the preimage and the `Data` before signing the hash with its VFX key (see vBTC Raw Integration).

8 Key maintenance

A key with fewer active holders than its threshold can never sign again. A withdrawal fixed on its coins could be neither paid nor released. So the protocol moves a vault to a new key while the old one still signs.

- Health.** Every 100 blocks, and at each epoch start, each node counts every vault key's holders that are active validators with a bond and no exit. A key with h holders and threshold t may lose $\max(1, \lfloor (h - t + 1) / 2 \rfloor)$ holders; past that it is **due** and stays due until replaced. The other half of the margin is the time the replacement takes.
- Re-key ceremony.** While a key is due, eligible validators run a new ceremony for the vault under a new key UID fixed by the chain. At least 67% of eligible validators take part. The new key signs at 67% of its holders, never all of them and never fewer than a majority. Every attestation names the vault being re-keyed.
- Vault re-key (type 46).** Sender-less, authorized by the attestations in its proof. The new key becomes the vault's current key.
- Sweep.** The old key's coins move to the new key by transactions fixed by rule and signed by the old key's holders like any withdrawal, shown as a Sweep in the vault's history. Bitcoin sent to the old address is still credited while its holders can sign.

Any validator may lead a re-key and submit its result; validators take turns. The leader chooses nothing: who holds a key, when it is due and what the new key must satisfy are read from the chain. S3C vaults are never re-keyed by the chain (§9).

9 S3C: Self-Custodial Smart Contracts

S3C lets an owner have a vault's key held only by validators the owner runs, instead of by the public validator set. It is the mechanism by which an institutional allocator runs its own FROST signing infrastructure.

S3C validators	Run with <code>S3CValidator=true</code> , which flags them as S3C when they register. Like any validator they must be registered, active and bonded. They take part only in S3C ceremonies; public vaults never use them.
Minting node	Lists the pool as <code>S3C=IP:ValidatorAddress,IP:ValidatorAddress,IP:ValidatorAddress</code> : at least three entries, public IP addresses only, valid VFX addresses. Every vault the node creates is then an S3C vault.
All or nothing	Every configured validator must be registered, active, flagged S3C, funded and bonded long enough, and all of them take part in the ceremony. If any is unusable, the ceremony fails and names it.
No fallback	If the <code>S3C=</code> value is invalid, the node refuses to create vaults rather than use the public pool.
Keys	The chain never re-keys an S3C vault. Who holds its key is its owner's matter.
Pre-flight	<code>GetS3CStatus</code> reports, per configured validator, whether it is registered on chain, reachable, and whether its configured IP differs from its registered one.

S3C vaults use the same attestation, accounting and withdrawal rules as public vaults. Their trust assumption differs: the threshold of key holders is the owner's own validators (§12).

10 Transaction types

TYPE	NAME	SENT BY	PURPOSE
18	TKNZ_TX (Transfer())	Owner	Transfer vault ownership
25	VBTC_V2_CONTRACT_CREATE	Owner	Create a vault (explorer: vBTC V2 Mint)
26	VBTC_V2_TRANSFER	Owner or holder	Transfer vBTC, one vault or up to 25
27	VBTC_V2_WITHDRAWAL_REQUEST	Requester	Request a withdrawal, one vault or up to 10
28	VBTC_V2_WITHDRAWAL_COMPLETE	Earlier vaults only	Refused for watched vaults; refused for all from the enforcement height
29	VBTC_V2_WITHDRAWAL_CANCEL	Requester	Cancel a waiting withdrawal
30	VBTC_V2_WITHDRAWAL_VOTE	Earlier vaults only	Refused for watched vaults; refused for all from the enforcement height
43	VBTC_V2_BTC_REPORT	Validators (no sender)	Bitcoin report
44	VBTC_V2_COIN_PIN	Requester	Fee raise or release of a pinned withdrawal
45	VBTC_V2_BTC_REORG	Validators (no sender)	Take back events after a Bitcoin reorganisation
46	VBTC_V2_VAULT_REKEY	Validators (no sender)	Move a vault to a new key

Types 43, 45 and 46 have `From` and `To` set to `BTC_Attestation`, no amount and no fee. V1 types 17 to 21 are retired (§13).

11 Protocol parameters and network heights

PARAMETER	VALUE	PARAMETER	VALUE
Bitcoin burial depth	6 (testnet 3)	Signing threshold (creation)	[51% of holders]
Signer epoch	600 VFX blocks	Re-key participation / threshold	67% / 67%
Report quorum	max(3, [2n/3])	Key health check	every 100 VFX blocks
Headers per report	≤ 500	Vault creation tx size	≤ 256 KB
Events per report	≤ 2,000	Withdrawal fee rate	1 to 5,000 sat/vB
Report transaction size	≤ 512 KB	Size estimate	154 vB + 57.5 per extra input
Quiet report range	144 Bitcoin blocks	Dust floor (payout, change)	546 sats
Backfill look-back	144 Bitcoin blocks	Inputs per withdrawal (max)	50
Coinbase maturity	100 Bitcoin blocks	Unpinned expiry	360 VFX blocks
Validator bond	5,000 VFX	Takeover after first pin	2,160 VFX blocks
Bond age to be eligible	14,400 VFX blocks	Payment / release transactions	≤ 8 / ≤ 4
Unlock after exit	1,200 VFX blocks	Vaults per multi transfer	≤ 25
Ceremony participation	≥ 90% of eligible, ≥ 3	Vaults per multi withdrawal	≤ 10
Ceremony participants (max)	512	Reserve transfer hold	24 hours

Network heights

HEIGHT	MAINNET	TESTNET
Bitcoin network	Bitcoin mainnet	VerifiedX signet (<code>signet.verifiedx.io</code>), 10-minute blocks
Bitcoin checkpoint (Bitcoin height)	969,696	2,016
Validator bond height	7,568,349	399
Attestation recording height	7,568,350	400
Attestation enforcement height	7,658,350 (recording + 90,000, about 12 days)	401
Core 9.0.1 ledger rules	7,575,250	7,400
V1 vBTC retired	7,400,500	Since block 1 (testnet restarted October 2026)

`GET GetBitcoinAttestationStatus` returns the heights a node runs with, the attested Bitcoin tip, the events-through height, the current signer epoch and the signatures required. The attested tip moves only when a report is mined, so a tip that stands still while no vault has activity is normal.

The VerifiedX network targets 12-second blocks (minimum 10, maximum 15), so VFX-block durations in this document are approximate.

12 Security properties and assumptions

12.1 Properties

PROPERTY	MECHANISM
The key is never whole	FROST DKG with shares sealed end to end to each recipient; signing produces signature shares, never the secret.
No creator-chosen address	The deposit address must be the Taproot address of a group key every participant attested and proved possession of, for this vault (§4.5).
No coordinator key recovery	Coordinators relay sealed ciphertext only; a coordinator that feeds participants different commitments is caught by the key package hash.
No credit without Bitcoin	A credit needs a buried block on a valid header chain and a Merkle proof; a block producer's word adds nothing (§5).
No overspend of a vault	Spends are checked against the vault's recorded coins in admission and block validation; a block that spends more is invalid.
No signer discretion over payouts	The paying transaction is computed by rule in block application. Honest signers sign only that transaction.
No double payment	Every transaction pinned for a withdrawal spends the same coins, so at most one can confirm. Cancel is refused once anything may have been signed.
No stranded withdrawal	Fee raise, release, takeover, expiry and void each end a withdrawal by what Bitcoin shows, with no vote.
Keys renewed in time	Re-keying starts when half the spare holders are gone, while the old key can still sign the sweep.
Stake behind every signer	Key holders and report signers are validators with 5,000 VFX bonded for at least two days.

12.2 Assumptions

- **Key holder threshold.** Fewer than a threshold of a vault's key holders collude. A colluding threshold can sign any transaction with the vault's key; the protocol makes such a spend visible (a coin spent outside a pin voids the withdrawals holding it) but cannot prevent it.
- **Key holder liveness.** At least a threshold of holders is online to sign pinned transactions and sweeps. Re-keying preserves this as validators leave, provided it completes in the margin.
- **Signer quorum.** Fewer than two thirds of an epoch's signer set collude. A colluding two thirds could follow a weaker valid branch, or omit a vault's events, though not fabricate a transaction, a proof of work or an inclusion proof.
- **Bitcoin.** Bitcoin blocks buried six deep (three on the testnet signet) are not reorganised away. On the testnet signet, block validity also rests on the signet's challenge key.
- **S3C.** For an S3C vault the key holders are the owner's own validators; the owner is trusted with its own key infrastructure, and the chain does not re-key it.

13 Earlier vaults

A vBTC V2 vault created before the recording height is not watched and is outside this specification. It is frozen at the enforcement height: from then on nothing debits a balance on it and Bitcoin sent to it is not credited. Holders should move to a new vault or withdraw before that height. V1 vBTC (arbiter tokenization, types 17 to 21) was retired at block 7,400,500 on mainnet; V1 balances are frozen as they stand.

14 References

Node API	docs.verifiedx.io: Integration › vBTC (every route, example and error)
External signing	docs.verifiedx.io: Integration › vBTC Raw Integration
Validators	docs.verifiedx.io: Core › Run a Validator (bond, ports, exit)
Testnet	docs.verifiedx.io: Integration › Testnet (VerifiedX signet)
Source	VerifiedX Core, VerifiedXCore/Bitcoin/Attestation , Bitcoin/FROST , Bitcoin/Services/S3CService.cs
Standards	BIP 125 (replace by fee), BIP 325 (signet), BIP 340 (Schnorr), BIP 341 (Taproot), FROST (RFC 9591, with the Zcash Foundation Taproot ciphersuite)

This document describes protocol behaviour as implemented in VerifiedX Core 9.0. It is not an offer of any asset or a statement of any asset's legal or regulatory status. Where this document and the Core source differ, the source governs.