PATTERN · STORAGE · BSTACK ON 1BSV

# bstore: storage as a market, where providers race for your data and are paid only on proof

Storage today is bought from one provider, at its price, on its word: you pay, and you trust that your data is still there, still whole and still listed. bstore makes every object a market. A writer puts it once, through ordinary S3, with a bounty attached; every storage provider on the bucket sees the offer at the same moment and the ones whose published price it covers race to hold it; the first to prove they hold it are paid, by rank; the rest earn nothing. Providers compete on price and on proof, a new one joins a bucket by subscribing, and a writer switches by changing an offer, not by moving a byte. Afterward anyone can buy a challenge that only a provider holding the bytes can pass, and the bucket's own chain of small mined tokens proves what every key names, which keys are absent, and that a listing is complete. Storage has had contracts, resellers and exchanges of capacity; it has not had a market for each object, settled on proof. This is that.

**Author:** Jeff Harris · Lightweb Inc. · jeff@lightweb.net **Pattern** 006 · **Version** 02

Audience: application developers, founders and business leaders. No Bitcoin or networking background assumed; every term is explained on first use.
**We** = Lightweb Inc., designer of the bstack patterns and operator of the 1BSV network; **you** = a builder or decision maker evaluating them.

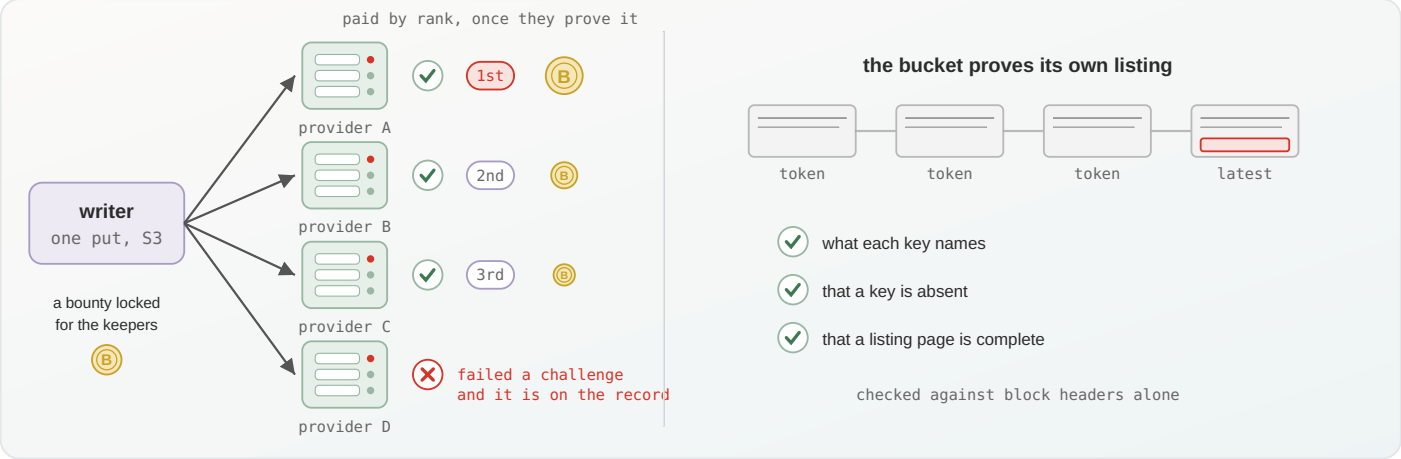



*Figure 1: one put, many keepers, each paid only after it proves it holds the bytes.*

## The series, and where this fits

The bstack pattern series rebuilds the building blocks nearly every product needs so that the data defends itself: state proves itself on the BSV ledger, BSV Layered Multicast delivers each publication once to every subscriber, and readers verify instead of trusting whoever served them.

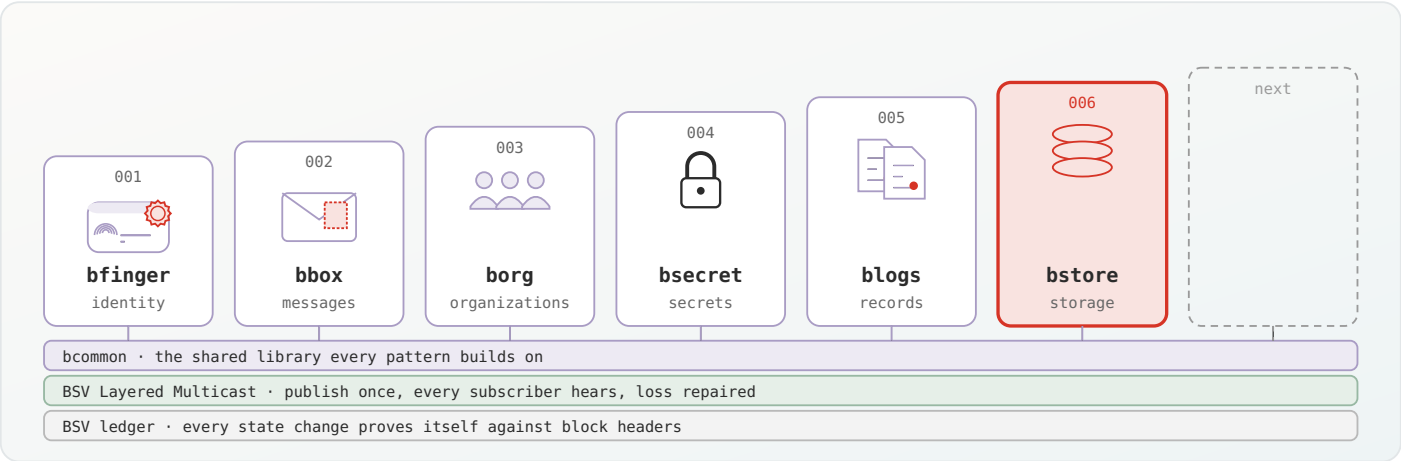



*Figure 2: the series as one build. bstore, in red, is the sixth pattern.*

**So far. bfinger** (001) made an identity prove itself and made it payable. **bbox** (002) delivered privately to one party, payment inside. **borg** (003) gave teams a roll and keys that change when someone leaves, **bsecret** (004) built a vault on it whose hosts never hold a key, and **blogs** (005) turned the foundation toward volume: a log any archive can prove complete.

**This one.** bstore turns it toward bulk. Records stay small: an entry, an envelope, a batch. Objects are photos, backups, video and datasets, far too large for any record, so they need keepers who are paid to hold them and who can be made to show they still do.

## The problem

Every product stores files, and every storage contract has the same shape: one provider, one bill and a promise. You cannot check that a provider still holds your data without downloading all of it. A second provider means a second upload, a second contract and a reconciliation to show the two copies match. And the listing a bucket returns is the provider's word: nothing proves that a key you expected is really absent, or that the page you were handed is the whole list.

There is no market to turn to. Prices are set provider by provider and compared by hand; no provider ever bids for your object, because none can see your offer. Switching means moving every byte, so the provider you have keeps you by inertia, not by price or by proof. And when a provider loses data, the first sign is usually a read that fails, long after the money was paid.

What a market would need is simple to state: an offer every provider sees at once, competition settled by evidence rather than by brand, and a way for a reader to check any keeper without asking the keeper. bstore is built as exactly that.

## What bstore does

A **writer** puts an object from the AWS CLI, an SDK or the bstore command line. The object is cut into pieces whose hashes name it everywhere, and every **storage provider** on the **bucket** receives the pieces. From there:

- **Providers race to hold it.** A put can carry a **bounty**: satoshis the writer locks for keepers. Each provider that checks the object and decides to keep it says so publicly, at once. The writer's client ranks them by when it first saw them, challenges each, and pays the first few that pass: the most to first place, and enough to second and third that showing up stays worth it.
- **Anyone can make a keeper prove it.** A **challenge** samples small segments of the object, chosen from a recent block and the buyer's own number, so no provider can know in advance which ones it will be asked for. It answers each with the bytes and a proof that they belong to the object, signed. The buyer can publish the outcome on the bucket for every later writer to read.
- **Anyone can buy more time.** A provider's promise to hold an object expires; a **renewal**, bought by anyone with an interest, a reader above all, extends it.
- **The bucket proves its own listing.** The bucket's owner signs a chain of small mined **tokens**; each folds in the latest writes and commits to every key. Against the latest token and its own block headers, a reader proves what a key names, that a key is absent, and that a page of a listing is complete.

```
$ aws s3 cp ./scan.tiff s3://photos/2026/scan.tiff
$ bstore holds -watch 10m             # rank the providers that hold it, challenge each
$ bstore settle                       # pay the first to pass, by rank
$ bstore -bucket photos_hmqzvtkwrx get 2026/scan.tiff ./scan.tiff -proof-root
```

*Figure 3: put, race, settle, read. Readers need no node and no coin; every byte they get is proven.*

## How it works

1. **An object is named by its content.** Its bytes are cut into 4 MiB pieces whose hashes form a tree (CHIRP, BRC-167); the hash at the top names the object everywhere, and a reader holds every piece to its hash before it uses a byte. A second, finer tree over 4 KiB segments is the object's **proof root**, which every challenge answers against.
2. **A write is an entry and its pieces.** The writer signs an entry (a put or a delete of a key) and publishes it on the bucket's records; the pieces go to the providers. An entry rides a small transaction that is never mined, spending one output of a mined **funding tree**, so a writer pays per object, not per byte of block space.
3. **Providers race, and the writer's client judges.** Each provider that checks the object advertises it with the ordinary UHRP advertisement (BRC-26): a small mined token, which expires. The writer's client ranks the advertisers by when it first saw them, challenges each, and pays the first that pass, by rank, with the square-root split of BRC-178. It pays nobody until enough distinct advertisers have passed.
4. **Anyone can challenge, and outcomes are public.** A challenge samples 4 KiB segments chosen from a recent block, the buyer's own number and the provider's key, so one provider cannot answer from another's response. The answer is each segment with its path to the proof root, signed. About a thousand samples find a single missing piece of a 1 GiB object with 98 percent confidence.
5. **Anyone can buy time.** A renewal, bought from a provider by anyone, extends its advertisement with a successor that spends the old one.
6. **The bucket proves its own listing.** The owner's tokens form one chain. Each folds in the writes since the last and commits to the root of the whole key map, using the same RFC 6962 tree construction Certificate Transparency uses. Hosts replicate the chain; none is believed on its word.

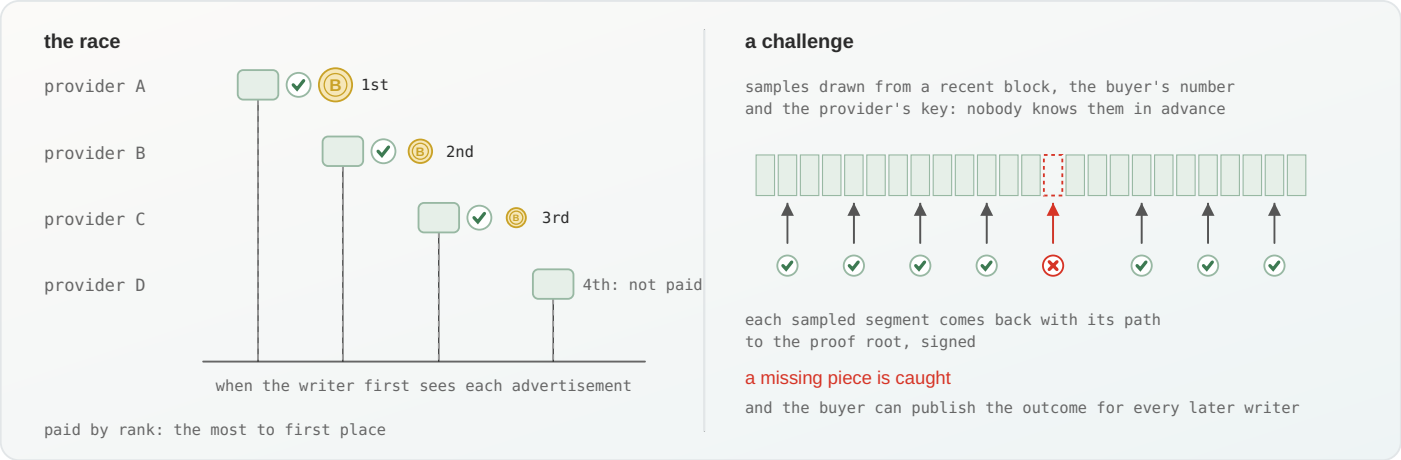



*Figure 4: providers race for the bounty; a challenge catches a missing piece, and the outcome is public.*

**Words in this brief.** *Writer*: whoever puts objects into a bucket. *Storage provider*: a service that keeps objects and is paid to. *Host*: a server that keeps a bucket's records (entries, advertisements, tokens and outcomes), never its bytes. *Bounty*: satoshis a writer locks for the providers that hold an object. *Advertisement*: a provider's mined, expiring statement that it holds an object. *Challenge*: a paid request that a provider can answer only by holding the bytes. *Proof root*: one short fingerprint that commits to every 4 KiB segment of an object.

## Why it is new

- **A write is a market, not a contract.** Each put can carry its own bounty, and the providers on the bucket compete for it the moment it arrives. The rank-and-split rule BSV's standards define for hosts answering the same lookup is applied here to keeping data.
- **Checking costs samples, not a download.** Anyone can ask a provider to prove it still holds an object, for a small price, and the answer is checked against the object's proof root, not against the provider's word. A buyer can publish the outcome on the bucket, so a provider's record travels with it.
- **The listing is a proof, not a report.** What a key names, what is absent and whether a page is complete are all checked against block headers alone.
- **Nothing new to learn.** Ordinary S3 clients write and read through a gateway. Existing UHRP hosts join through a bridge; current uhrp-server-basic releases need a one-line fix to commit these objects, and the bridge checks before it pays.

## Built to scale: BSV Layered Multicast

A bucket with one provider is a hosting contract, not a market. The value is in several independent providers, in different organizations and places, each holding the same object and each able to prove it. Point to point, that costs one upload per provider, and every provider added is one more upload.

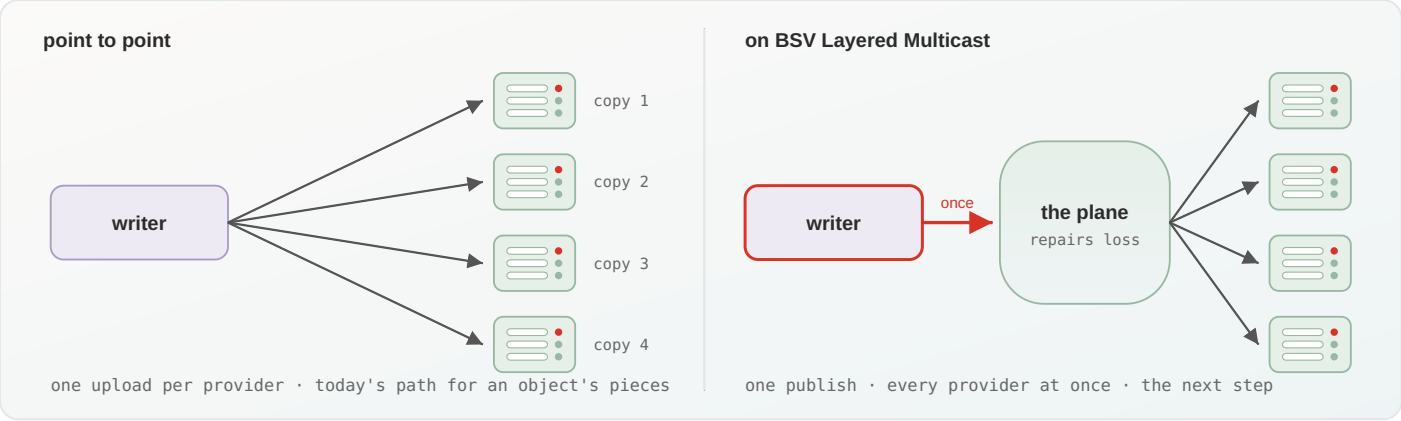



*Figure 5: where it goes next: one publish per piece, whatever the number of providers. A bucket's records already travel this way.*

Today bstore puts a bucket's records on BSV Layered Multicast: every entry, advertisement, token and challenge outcome reaches every host that carries the bucket at the same moment, with loss repaired by the network, and a new host joins by subscribing. An object's pieces go to each provider directly, one upload per provider, the way storage hosts take uploads today. The protocol already defines the next step: each piece published once on the plane, so that every provider on the bucket receives it at once and a writer's cost stops growing with every provider added.

## Two flows, one proof

Most people measure a blockchain by what it mines, and by that measure keeping data on chain scales badly: every byte pays a miner, and the objects a block can hold shrink as the objects grow. bstore separates two flows. The bytes of an object ride the **data carrier**: a small transaction that is never mined, delivered to every provider on the bucket. What is mined is the proof of the object: one funding output per piece, the hold output and the settlement, a few hundred bytes to a few kilobytes whatever the object's size. A 1 GiB put mines about 14 KB; mining the gigabyte itself would be 76,000 times as many bytes. More objects cost more proofs, not more bytes mined per byte stored.

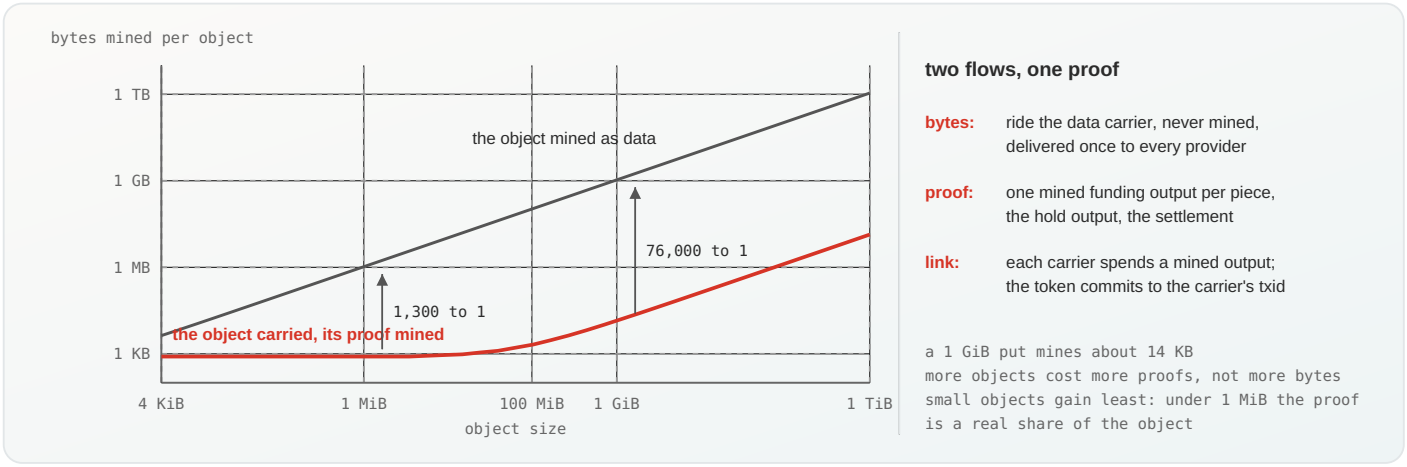



*Figure 6: bytes mined per object against its size. The two flows are decoupled; the link between them is not.*

The link between the two flows is provable either way: each carrier spends one output of a mined funding tree that only the writer's key can spend, and the entry's name, its carrier's transaction id, is what the bucket's mined token commits to, so a reader walks from a block header to the token, the entry and every byte with nothing taken on anyone's word. The bytes are retractable and the commitment is not: a sweep of the funding outputs removes the carriers from every host, while the token's record of them stays on chain. The honest edge is small objects: under about 1 MiB the proof is a real share of the object, part of why they cost more here than at S3.

## What it means for the business

| Who | Pays | Gets |
|---|---|---|
| A writer | a bounty for holds; miner fees per object (a small share of a funding tree), from coin it already owns; no node of its own | copies held by independent providers, each of which had to prove it |
| A storage provider | its disk and its bandwidth | bounties, renewals and paid challenges; a public record of the checks it passed |
| A reader | nothing to read: no node, no coin; a challenge or a renewal when it wants one | every byte proven, and every listing |

- **Storage competes object by object.** Providers race for each put, and a writer raises the bounty until enough of them hold it.
- **An audit without a download.** "Do you still have it?" becomes a small paid question with a provable answer.
- **Independence you can count.** The tooling says how many distinct providers hold an object, and which published checks they passed or failed.
- **It fits the tools a team already runs.** The AWS CLI and the S3 SDKs work through the gateway, and existing UHRP hosts can join the market.

## What it costs, by the GB

Storage is bought by the GB, so here is bstore beside the list prices people compare it with. bstore has no price list: each provider publishes its own price per GiB per day. The software's default is 250 satoshis, and the market will find its own.

| Per GB, per month | Stored | Read back |
|---|---|---|
| Amazon S3 Standard | $0.023 | $0.09 |
| Cloudflare R2 | $0.015 | no egress fee |
| Wasabi | $0.008 | free |
| Storj | $0.007 | $0.007 |
| Backblaze B2 | $0.007 | free to three times what is stored |
| bstore, three copies at 500 satoshis a GiB-day | $0.00767 | free |
| bstore, three copies at the default 250 satoshis | $0.00384 | free |

*Table 1: list prices read 2026-11-06, S3 in us-east-1 at its first tiers, Wasabi with its 1 TB and 90-day minimums. The bstore lines are arithmetic for large objects at 1 BSV = $17.05 = €15.21 on 2026-11-06, not a price list.*

bstore promises less and proves more. No company stands behind a copy: a writer buys several, from different providers, and checks them. What's left to pay for is a disk and its bandwidth, and anyone with those can compete.

Measured, a writer's whole put of 1 GiB to three providers for 30 days at the default price is 24,227 satoshis, $0.00413 (€0.00368): a 22,500 satoshi bounty and 1,727 satoshis of funding and fees. The bytes themselves are never mined; mining them would pay over 100 million satoshis, about $18.

- **Anyone can earn.** The provider is open source and needs nobody's permission. In our lab a new provider started from an empty directory with 50,000 satoshis, under a cent, and the next put it raced paid it 8,454 satoshis. It publishes its price, writers choose who to upload to, and it races for bounties, earns renewals and sells challenges, all paid in BSV. Ten TiB full at the default earns about 0.77 BSV a month.
- **Providers compete on price.** A provider races only when a writer's offer covers its own price for the object's size and time. In our lab a provider asking 1,000 satoshis sat out an offer that two cheaper ones took; it cut to 350, raced again and was paid.
- **The user wins.** A writer sees the prices, sets the offer and pays for proof instead of a promise. Three copies at the default come in under every list price above, and stay under S3 Standard's until providers ask about 6 times the default. Providers and their customers set the real price.

## Honest limits

- **A price is not a service commitment.** S3 is one company's promise; here a writer buys copies from providers who may leave, which is why it buys more than one. Reads are free because providers carry them, so a provider's bandwidth has to come out of its storage price, and a satoshi price moves against the dollar.
- **Small objects cost more.** Every object pays its own on-chain costs and each provider's minimum, so below about 10 MB S3 Standard is cheaper by the GB.
- **The market has to form.** We run no public providers. The bstore prices above are arithmetic until providers publish their own, and a new provider has to be found: there is no directory, and a writer names who it uploads to.
- **Availability is a market outcome,** not a guarantee: a bounty too small may find no keeper, and the tooling says so.
- **Keys are not copies.** Several providers' keys may sit on one disk; independence is the writer's judgment.
- **Enforcement is reputation in this version.** A failed check is caught and published; nothing is taken from the provider. A pass is the buyer's word, and only a wrong signed answer is provable by anyone.
- **Possession is proven at challenge time,** to the confidence the sample gives.
- **Metadata is public:** keys, sizes and writers are visible on the bucket. Encrypt before you put what must stay private.
- **A put is not a hold** until providers advertise it, and erasure is not possible: a delete removes a key, not bytes someone already fetched.

## What comes next

Six patterns in, the series has an identity, a private message with payment inside, a team, its secrets, a record of everything that happened, and a place to keep anything too large for a record. bstore's proof format is built so that a later version can pay a keeper by script on a valid proof, which moves enforcement from reputation to the ledger. The patterns that follow compose these into the products people use every day, each standing on the ones before it.

## Run it on 1BSV

The 1BSV network is the commercial operation of BSV Layered Multicast, run by Lightweb Inc. For an application built on bstore:

- **Writers ride free.** Publishing into the network is never billed, and neither are the network's repair retransmissions.
- **Hosts pay for what is delivered to them**, and for nothing else: overlay objects with their proofs on an application account, which also carries the live header feed to verify against. No subscriptions, no minimums, no commitments.
- **Early access.** The overlay lane is onboarding its first users. Onboarding is self-service with one approval step.

## Go deeper

- *The bstack: publish once, prove everything, and let users ride free* (Lightweb Inc., Paper 009): https://1bsv.net/papers/bstack-patterns.pdf
- *The overlay object plane* (Lightweb Inc., Paper 006): https://1bsv.net/papers/overlay-object-plane.pdf
- The bstore repository, with the specification, the user guide, the S3 gateway and a one-machine quickstart: https://github.com/lightwebinc/bstore
- The bridge for existing UHRP hosts: https://github.com/lightwebinc/bstore-bridge
- The open standards it builds on, in the BRC repository: BRC-26 (UHRP advertisements), BRC-167 (CHIRP objects), BRC-178 (the rank-paid race), BRC-29 and BRC-105 (payments): https://github.com/bitcoin-sv/BRCs
- The proof tree follows RFC 6962 (Certificate Transparency): https://www.rfc-editor.org/rfc/rfc6962; a challenge seeded from a block, as Sia's storage proofs are: https://github.com/SiaFoundation/core
- The pattern series: https://1bsv.net/patterns.html

---

**1BSV - BSV Layered Multicast** - cash moves fast on 1BSV. · 1bsv.net

---

**Lightweb Inc.** · Jeff Harris · jeff@lightweb.net · github.com/lightwebinc

*Unbounded Solutions for a Small World™ · © Lightweb Inc.*

© 2026 Lightweb Inc. All rights reserved. 1BSV and Unbounded Solutions for a Small World are trademarks of Lightweb Inc. This pattern describes an architecture in rapid development: the design, its interfaces and this document are subject to change without notice, and later versions supersede this one. Nothing in this pattern is a commitment to ship any feature, an offer of service, or professional advice.