$402 Protocol

Technical Implementation Specification

Version 0.1 Draft

February 2026

Richard Boase

b0ase.com

Abstract

The $402 Protocol is a token standard and URL convention for paywalled content on the web. It uses the HTTP 402 'Payment Required' status code, reserved since 1997 but never implemented, to enable native micropayments for web content. By embedding the '$' character in URL paths, servers signal that payment is required. Payments are made via BSV21 tokens, where the transaction itself serves as proof of validity. This document specifies the protocol mechanics, token model, and implementation requirements.

Contents

1. Introduction

The HTTP 402 status code was reserved in HTTP/1.1 (RFC 2616) for 'Payment Required' but was never standardized due to the lack of a viable micropayment infrastructure. With the advent of Bitcoin SV and sub-cent transaction fees, this infrastructure now exists.

The $402 Protocol solves a simple problem: how does a server know if a client has paid for access? The answer is equally simple: the client sends a token (UTXO) to the server. The transaction IS the proof. No indexer lookup required. No validity oracle. If the server receives the token, access is granted.

1.1 Design Principles

Simplicity: Use existing infrastructure (HTTP, BSV, BSV21 tokens).

No reinvention: Bitcoin already solved double-spend. UTXOs already enforce single ownership.

Payment as proof: The transaction is the validation. No external dependencies.

URL-native: The '$' in the path signals the paywall. Human-readable. Spec-compliant.

2. URL Convention

The '$' character is a reserved character in URLs per RFC 3986 Section 2.2 (sub-delimiters). It is legal to use in URL paths without encoding.

2.1 Syntax

A paywalled path is indicated by prefixing the path segment with '$':

```
https://example.com/$video-1
https://example.com/$chatroom
https://alice.com/$premium/episode-5
```

2.2 Path Hierarchy

Tokens are scoped to paths. A token for a parent path grants access to all child paths:

`$/alice` grants access to `$/alice/chatroom`, `$/alice/video-1`, etc.

`$/alice/video-1` grants access only to that specific path.

2.3 Free vs Paid

Paths without '$' are free (standard HTTP). Paths with '$' require payment. This creates a clear, visible distinction in the URL itself.

3. Token Model

Tokens are BSV21 fungible tokens minted by the content creator for a specific path. Each token represents a 'ticket' granting access to that path.

3.1 Token Properties

Path-bound: Each token is associated with a specific path (e.g., $/alice/chatroom).

Reusable: Tokens are not burned on use. They return to the issuer and can be resold.

Tradeable: Tokens can be bought/sold on secondary markets.

UTXO-based: Token ownership is enforced by Bitcoin's UTXO model. No double-spend possible.

3.2 Ticket Economics

The issuer (content creator) sets the initial price and supply. Conditions can be attached:

1 ticket = 1 view, or 1 hour of access, or 1 API call, etc.

When a ticket is spent (used for access), it returns to the issuer. The issuer can resell it or hold it to control supply and price.

3.3 Supply and Pricing

Total supply represents maximum concurrent capacity. If the issuer mints 100 tickets for a chatroom where 1 ticket = 1 hour, then 100 users can access concurrently per hour.

Price is market-driven. High demand with limited supply increases price. Low demand floods the market, price decreases asymptotically toward zero.

4. Payment Flow

4.1 Request-Response Cycle

1. Client requests a '$' path: `GET /$/video-1`

2. Server returns `402 Payment Required` with payment details:

```
{ "token_id": "abc123...", "price": "0.001 BSV", "address": "1Alice...",
"path": "$/video-1" }
```

3. Client sends token (UTXO) to the specified address.

4. Server detects incoming transaction.

5. Server grants access (serves content, opens connection, etc.).

4.2 Validity

The server does not need to query an indexer or validate against an external source. If the transaction arrives, it is valid. The Bitcoin network handles consensus. The payment IS the proof.

4.3 Confirmation

For low-value transactions (micropayments), zero-confirmation is acceptable. The risk of a double-spend attack for a video view is negligible. For high-value access, servers may wait for confirmations.

5. Server Implementation

A $402-compliant server must:

1. Detect '$' in incoming request paths.

2. Return 402 Payment Required for unpaid requests, with payment instructions.

3. Monitor a wallet address for incoming token transactions.

4. Grant access upon receipt of valid token.

5. (Optional) Index only its own tokens for its own paths.

5.1 Minimal Implementation

A server only needs to index tokens for paths it controls. Alice's server indexes $/alice/* tokens. It does not need to know about $/bob/* or any other namespace. This scales linearly with content, not with network size.

6. Client Implementation (path402d)

The reference client is `path402d`, a daemon providing:

Wallet: Hold, send, receive BSV21 tokens.

Server: Receive tokens, grant access to local content.

Discovery: Gossip network for finding paths and content.

Exchange: (Optional) Built-in marketplace UI for buying/selling tokens.

6.1 Installation

```
npm install path402
```

or

```
pip install path402
```

6.2 Interfaces

CLI, GUI, and Web interfaces are available. The daemon runs in the background.

7. Discovery and Gossip

Clients discover content via a gossip network, similar to BitTorrent DHT. Nodes broadcast paths they serve and tokens they hold.

7.1 Gossip Protocol

Nodes running path402d join a P2P mesh network. They announce:

- Paths they serve (content they host)

- Tokens they hold (speculation/investment)

- Tokens they are willing to sell

7.2 Relay Nodes

Nodes may relay content for others (like BitTorrent seeders). They earn from ticket velocity - the faster tickets cycle through them, the more they earn.

7.3 Indexing

Relay nodes index broadly across the network. They serve as discovery and caching layers. Content creators index only their own tokens. The network self-organizes based on incentives.

8. Staking and Dividends

8.1 Holding vs Staking

Holding: Permissionless. No KYC. Speculative position. No dividends.

Staking: Requires KYC with the issuer. Registered on cap table. Earns dividends.

8.2 Dividend Flow

When tickets are purchased from an issuer, a portion of revenue flows to stakers proportional to their stake. Dividends require KYC because they create tax and regulatory liability.

8.3 Why KYC for Dividends

Sending dividends to anonymous parties creates legal liability for the issuer. The KYC requirement is not ideological - it is practical compliance. Permissionless trading and holding remain intact.

9. Security Considerations

9.1 Double-Spend

Bitcoin's UTXO model prevents double-spending. A token can only be spent once. For zero-conf transactions, there is a small window for attempted double-spends, but for micropayments, the risk/reward makes attacks impractical.

9.2 Token Copying

Tokens are UTXOs, not files. You cannot 'copy' a UTXO. Ownership is enforced by the Bitcoin network.

9.3 Relay Trust

Relay nodes share knowledge about token states. The chain is the source of truth for ownership. Gossip is for discovery and caching, not consensus.

10. Examples

10.1 Video Content

Alice hosts a video at `alice.com/$video-1`. She mints 10,000 tokens at $0.10 each. 1 token = 1 view. Users buy tokens, send to Alice's address, watch video. Token returns to Alice. She resells. Revenue: $0.10 per view.

10.2 Live Chatroom

Alice hosts a chatroom at `alice.com/$chatroom`. She mints 100 tokens. 1 token = 1 hour of access. Maximum 100 concurrent users. Price floats based on demand.

10.3 API Access

A developer offers an API at `api.example.com/$query`. 1 token = 1 API call. Bulk purchasers can buy many tokens for high-volume access. Price per call decreases with volume (bulk discount via secondary market).

Conclusion

The $402 Protocol completes a circuit left open since 1997. By combining a simple URL convention, the HTTP 402 status code, and BSV21 tokens on Bitcoin SV, we enable native micropayments for web content without platforms, intermediaries, or complex infrastructure.

The payment is the proof. The '$' is the signal. The web finally works the way it was supposed to.

This specification is a living document. For updates, see b0ase.com/$402