/drop API Specification

Version: 1.2.0
Powered by: Covenant Protocol (C-Protocol)

1. Overview

The /drop API enables developers to create, exchange, and redeem private, self-custodied digital containers ("Drops") that can hold both assets (e.g., BSV, bridged SPL tokens) and arbitrary data payloads (e.g., media, metadata).

Each Drop is a C-Protocol covenant. This API specification defines a set of stateless endpoints that any peer can host, allowing Drops to be created, shared offline (via NFC, QR, etc.), and redeemed on the Bitcoin network without reliance on a central server. The state of any Drop can be independently verified by any peer from the on-chain covenant.

2. C-Protocol Covenant Types

- **quick Drop (Proof-of-Possession):** A P2PKH script. Unlocked by a signature from the corresponding private key.
- **locked Drop (Proof-of-Knowledge):** A hashlock script. Unlocked by the secret pre-image.
- **timed Drop (Proof-of-Time):** A time-locked covenant that becomes spendable after a specific block height or time.
- **bridged Drop (Proof-of-Bridged-State):** A voucher UTXO requiring a signature from a trusted bridge authority for cross-chain release.

3. Authentication & P2P Model

The API is stateless and non-custodial. All cryptographic signing operations (creating, funding, claiming) are performed client-side. Any participant in the network can run a peer node that hosts these API endpoints to facilitate the exchange of Drops.

4. API Endpoints

4.1. Create a Drop

POST /api/drop/create

Initializes a C-Protocol covenant and binds an optional data payload to it.

Request Body:

```
{
  "senderAddress": "1...",
  "assetId": "BSV:...",
  "amount": 50000,
```

```
  "dropType": "locked",
  "proofDefinition": {
    "hash": "f2ab34cde..."
  },
  "payload": {
    "mimeType": "image/webp",
    "data": "base64-encoded-string-of-payload",
    "size": 788
  },
  "memo": "Avatar + 50,000 sats"
}
```

- payload (object, optional): An object containing the data to be associated with the Drop. The hash of this payload will be included in the covenant.

Response:

```
{
  "dropId": "d-93ff8c-...",
  "unsignedTx": "01000000...",
  "claimLink": "drop://claim/d-93ff8c-...",
  "qrCodeData": "drop://claim/d-93ff8c-..."
}
```

- claimLink (string): A URI using a peer-to-peer schema, allowing any compatible client to handle the claim process.

4.2. Fund a Drop

POST /api/drop/fund

Submits the signed transaction to fund the Drop covenant on the Bitcoin ledger.

Request Body:

```
{
  "dropId": "d-93ff8c-...",
  "signedTx": "01000000..."
}
```

Response:

```
{
```

```
  "status": "funded",
  "txid": "...",
  "covenantUtxo": { "txid": "...", "vout": 0 }
}
```

4.3. Claim a Drop

POST /api/drop/claim

Constructs and broadcasts the transaction to unlock the covenant using the required proof.

Request Body:

```
{
  "dropId": "d-93ff8c-...",
  "recipientAddress": "1...",
  "proof": {
    "type": "secret",
    "value": "hunter2"
  }
}
```

Response:

```
{
  "status": "claimed",
  "txid": "...",
  "assetReleased": {
    "assetId": "BSV:...",
    "amount": 50000
  }
}
```

4.4. Get Drop Status

GET /api/drop/status/{dropId}

Retrieves the public state and payload information for a given Drop. A peer can serve this information from its local cache or by inspecting the blockchain.

Response:

```
{
```

```
  "dropId": "d-93ff8c-...",
  "status": "funded",
  "assetId": "BSV:...",
  "amount": 50000,
  "dropType": "locked",
  "payload": {
    "mimeType": "image/webp",
    "size": 788,
    "hash": "sha256:abcd..."
  },
  "covenant": {
    "script": "76a914..."
  }
}
```


4.5. Discover Drops

GET /api/drop/discover

Scans the local peer-to-peer network for Drops being broadcast by other nearby peers. This is intended for use with transports like Bluetooth LE, NFC, or local Wi-Fi.

Query Parameters:

- transport (string, optional): Hint for the desired discovery method (e.g., ble, nfc, local). Defaults to all available.
- radius (number, optional): Search radius in meters for location-based transports.

Response: A list of publicly broadcasted Drop summaries.


```
[
  {
    "dropId": "d-a1b2c3-...",
    "assetId": "BSV:...",
    "memo": "Coffee fund",
    "peerId": "peer-xyz..."
  },
  {
    "dropId": "d-d4e5f6-...",
    "assetId": "SOL:...",
    "memo": "USDC for lunch",
    "peerId": "peer-abc..."
  }
]
```

- peerId (string): An identifier for the peer broadcasting the Drop, which can be used to directly request the Drop status.

5. Data Models

Drop Object:

```
interface Drop {
  dropId: string;
  status: 'pending' | 'funded' | 'claimed' | 'expired';
  assetId: string;
  amount: number;
  dropType: 'quick' | 'locked' | 'timed' | 'bridged';
  covenant: {
    script: string;
    utxo?: { txid: string; vout: number; };
  };
  payload?: {
    mimeType: string;
    size: number;
    hash: string; // e.g., "sha256:abcd..."
  };
  bridgeDetails?: {
    sourceChain: string;
    depositAddress: string;
    destinationAddress?: string;
  };
  createdAt: string;
  claimedAt?: string;
}
```