

API Endpoints

Overview

This documentation covers the REST API endpoints required to implement vault creation, asset management, and governance functionality for the L4VA protocol. The API enables:

1. Vault Formation

- Creation of Private, Public, and Semi-Private vaults
- Asset contribution during timed windows
- Investment handling with Fixed/LBE options

2. Governance

- Proposal creation and management
- Voting mechanisms
- Multi-signature execution flows
- Threshold validations

3. Wallet Integration

- Wallet connection and session management
- Vault association and role management

- [Getting Started](#)
- [Vault Formation](#)
- [Governance](#)
- [Wallet Integration](#)
- [Swim Lane Diagram](#)

Getting Started

1. Obtain API credentials
2. Review authentication requirements
3. Test endpoints in development environment
4. Implement error handling
5. Add real-time updates using WebSocket endpoints

Environment URLs

- Development: <https://api-dev.l4va.example.com>
- Staging: <https://api-staging.l4va.example.com>
- Production: <https://api.l4va.example.com>

Authentication

All endpoints require API key authentication using the `X-API-Key` header.

Response Format

All endpoints follow a consistent response format:

```
{
  "status": "success|error",
  "data": {}, // Response payload
  "error": {} // Present only on errors
}
```

Blockchain Integration

- All state-changing operations return transaction hashes
- Wallet signatures required for sensitive operations
- Real-time updates available via WebSocket connections

Postman Collection

TBD: Provide a Postman Collection

Vault Formation

Create Vault

Creates a new vault with specified configuration.

Endpoint

POST /api/v1/vaults

Sample Request

```
{
  "vault_name": "Example Vault",
  "vault_type": "public",
  "privacy_type": "semi-private",
  "admin_user": {
    "wallet_address": "addr1q9example123",
    "email": "admin@example.com"
  },
  "asset_settings": {
    "allowed_asset_types": ["NFT", "CNT"],
    "policy_ids": ["policy12345", "policy67890"],
    "valuation_type": "LBE",
    "floor_price_percentage": 90,
    "max_assets": 100
  },
  "investment_settings": {
    "investment_window_duration": "48h", // ISO-like duration
    "investment_start_time": "2024-11-24T10:00:00Z", // Optional; defaults to vault creation time
    "minimum_investment_reserve": 10.0,
    "ft_supply": 100000,
    "ft_token_decimals": 6,
    "lp_percentage": 10
  },
  "governance_settings": {
    "creation_threshold": 5,
    "start_threshold": 10,
    "vote_threshold": 50,
    "execution_threshold": 60,
    "cosigning_threshold": 3
  }
}
```

```
}  
}
```

Sample Response (201: Created)

```
{  
  "vault_id": "vault123",  
  "vault_name": "Example Vault",  
  "transaction": {  
    "tx_hash": "b26f8c9a0d1a4a9b8b12f9f9a8c1234567e9d0f1c234567890abcdef",  
    "status": "confirmed",  
    "block_height": 1234567  
  },  
  "investment_window": {  
    "duration": "48h",  
    "start_time": "2024-11-24T10:00:00Z",  
    "end_time": "2024-11-26T10:00:00Z"  
  },  
  "status": "created",  
  "created_at": "2024-11-23T10:00:00Z"  
}
```

Add Assets to Vault

Add assets during the asset window period.

HTTP Request

```
POST /api/v1/vaults/{vaultId}/assets
```

Sample Request

```
{  
  "policy_id": "policy12345",  
  "asset_name": "ExampleAsset",  
  "valuation_method": "floor_price",  
  "metadata": {  
    "creator": "CreatorAddress",  
    "legal_proof": "https://proof.example.com/doc.pdf"  
  }  
}
```

Sample Response (201: Created)

```
{
  "vault_id": "vault123",
  "asset_id": "asset456",
  "transaction": {
    "tx_hash": "a92f81e3b69c4d12b34c567890abcde1234567890abcdef56789abc",
    "status": "confirmed",
    "block_height": 1234570
  },
  "status": "added",
  "valuation": {
    "method": "floor_price",
    "value": 1000000
  },
  "created_at": "2024-11-23T11:00:00Z"
}
```

Remove Asset

Remove assets during the asset window period. Will fail if attempted before or after the window period.

Endpoint

```
DELETE /api/v1/vaults/{vaultId}/assets/{assetId}
```

Sample Response

```
{
  "status": "success",
  "data": {
    "assetId": "asset_123",
    "removalStatus": "COMPLETED",
    "transactionHash": "tx_hash789...",
    "updatedValuation": {
      "total": "140000",
      "timestamp": "2024-12-01T01:35:00Z"
    }
  }
}
```

Get Vault Valuation

Calculate current vault valuation based on assets.

HTTP Request

```
GET /api/v1/vaults/{vaultId}/valuation
```

Sample Response

```
{
  "status": "success",
  "data": {
    "valuation": {
      "total": "150000",
      "breakdown": {
        "nftValue": "120000",
        "cntValue": "30000"
      },
      "assetCounts": {
        "nfts": 2,
        "cnts": 1
      },
      "timestamp": "2024-12-01T01:30:00Z"
    }
  }
}
```

List Vault Assets

List all assets in the vault.

HTTP Request

```
GET /api/v1/vaults/{vaultId}/assets
```

Sample Response

```
{
  "status": "success",
  "data": {
    "assets": [{
```

```
"assetId": "asset_123",
"type": "NFT",
"contractAddress": "addr_nft123...",
"tokenId": "42",
"addedAt": "2024-12-01T01:00:00Z",
"status": "LOCKED",
"currentValue": "90000"
}],
"pagination": {
  "page": 1,
  "limit": 20,
  "total": 45
}
}
```

Update Vault AllowList

Updates the list of wallets allowed to participate in the vault, and their type.

HTTP Request

```
PATCH /api/v1/vaults/{vaultId}/allowlist
```

Sample Request

```
{
  "type": "ASSET|CONTRIBUTOR|INVESTOR",
  "operation": "ADD|REMOVE",
  "addresses": ["addr1...", "addr2..."]
}
```

Sample Response

```
{
  "status": "success",
  "data": {
    "updatedAllowList": {
      "type": "ASSET",
      "count": 24,
      "lastUpdated": "2024-12-01T02:00:00Z"
    }
  }
}
```

```
}  
}  
}
```

Vaut Performance Metrics

Returns some vault performance metrics.

HTTP Request

```
GET /api/v1/vaults/{vaultId}/metrics
```

Sample Response

```
{  
  "status": "success",  
  "data": {  
    "valuation": {  
      "initial": "100000",  
      "current": "150000",  
      "change": "50.00"  
    },  
    "participation": {  
      "uniqueVoters": 45,  
      "averageQuorum": "68.00",  
      "proposalCount": 12  
    },  
    "timeline": {  
      "created": "2024-11-20T10:00:00Z",  
      "locked": "2024-12-01T02:00:00Z",  
      "age": "11d 16h"  
    }  
  }  
}
```

Vault Activity

Returns a vault activity log.

HTTP Request

```
GET /api/v1/vaults/{vaultId}/activity
```


Sample Response

```
{
  "status": "success",
  "data": {
    "activities": [{
      "type": "ASSET_ADDED|PROPOSAL_CREATED|VOTE_CAST",
      "timestamp": "2024-12-01T01:00:00Z",
      "actor": "addr_user123...",
      "details": {},
      "transactionHash": "tx_hash123..."
    }],
    "pagination": {
      "page": 1,
      "limit": 20,
      "total": 156
    }
  }
}
```

Update Vault Settings

Lets you update modifiable vault settings.

Sample Request

```
{
  "investorAllowList": {
    "enabled": true,
    "addresses": ["addr1..."]
  },
  "valuationType": "LBE",
  "termination": {
    "fdp": "12.00"
  }
}
```

This would return the standard Success/Fail response object.

Governance

Proposal Management

Create a new governance proposal.

HTTP Request

```
POST /api/v1/vaults/{vaultId}/proposals
```

Sample Request

```
{
  "proposer": "user_wallet123",
  "proposal_details": {
    "title": "Liquidate Asset A",
    "description": "Sell asset A for 90% of its floor price.",
    "action_type": "sell_asset",
    "affected_assets": [
      {
        "policy_id": "policy12345",
        "asset_name": "ExampleAsset"
      }
    ]
  },
  "governance_thresholds": {
    "creation_threshold": 5,
    "start_threshold": 10,
    "vote_threshold": 50,
    "execution_threshold": 60
  }
}
```

Sample Response (201: Created)

```
{
  "vault_id": "vault123",
  "proposal_id": "proposal789",
  "transaction": {
    "tx_hash": "de1234abc567890def1234567890abcdef1234567890abcdef12345678",
    "status": "confirmed",
  }
}
```

```
"block_height": 1234575
},
"status": "created",
"thresholds_met": {
  "creation_threshold": true
},
"created_at": "2024-11-23T12:00:00Z"
}
```

Submit Vote

Submit a vote on a proposal.

HTTP Request

```
POST /api/v1/proposals/{proposalId}/votes
```

Sample Request

```
{
  "voter": {
    "wallet_address": "addr1q9voter123",
    "staked_ft_amount": 1000
  },
  "vote": "yes"
}
```

Sample Response (200: OK)

```
{
  "vault_id": "vault123",
  "proposal_id": "proposal789",
  "voter": "addr1q9voter123",
  "transaction": {
    "tx_hash": "ab567890abcdef1234567890abcdef1234567890abcdef1234567890",
    "status": "confirmed",
    "block_height": 1234580
  },
  "vote": "yes",
  "staked_ft_amount": 1000,
  "status": "vote_recorded",
}
```

```
"updated_at": "2024-11-23T13:00:00Z"
}
```

Execute Proposal

Execute an approved proposal.

Endpoint

```
POST /api/v1/proposals/{proposalId}/execute
```

Sample Request

```
{
  "executing_user": "user_wallet123",
  "cosigners": [
    "wallet_cosigner1",
    "wallet_cosigner2"
  ],
  "action": {
    "type": "sell_asset",
    "details": {
      "policy_id": "policy12345",
      "asset_name": "ExampleAsset",
      "price": 900000
    }
  }
}
```

Sample Response (200: OK)

```
{
  "vault_id": "vault123",
  "proposal_id": "proposal789",
  "transaction": {
    "tx_hash": "cd1234567890abcdef1234567890abcdef1234567890abcdef123456",
    "status": "confirmed",
    "block_height": 1234585
  },
  "status": "executed",
  "executed_by": "user_wallet123",
}
```

```
"cosigners": [  
  "wallet_cosigner1",  
  "wallet_cosigner2"  
],  
"action": {  
  "type": "sell_asset",  
  "details": {  
    "policy_id": "policy12345",  
    "asset_name": "ExampleAsset",  
    "price": 900000  
  }  
},  
"executed_at": "2024-11-23T14:00:00Z"  
}
```

Wallet Integration

About CIP-08 and CIP-30

This guide is a walkthrough on how to implement the *message signing* described in [CIP-08](#) in order to authenticate users on the web with just their [CIP-30](#)-compatible wallet app:

<https://developers.cardano.org/docs/integrate-cardano/user-wallet-authentication/>

However while it is useful to study the above, in order to simplify and standardize our platform, we're going to use a multi-chain wrapper library instead (see next section).

Using Weld

We're going to use this library to integrate with Cardano wallets:

<https://github.com/Cardano-Forge/weld>

Weld lets you manage wallet connections across multiple blockchains using a single intuitive interface.

How Authentication works

In a Web3 app using wallets like **Nami** or **Vespr**, authentication typically works through **wallet-based signature verification**. Here's a simplified flow:

1. **Connect Wallet:** The user connects their wallet to the Web3 app. The wallet extension (like Nami or Vespr) interfaces with the app to allow interactions.
2. **Generate Nonce:** The app generates a unique, random string (nonce) and sends it to the wallet for the user to sign. This ensures that each authentication request is unique and prevents replay attacks.
3. **Sign Nonce:** The user signs the nonce using their private key in the wallet. This signature doesn't expose the private key but proves ownership of the wallet.
4. **Verify Signature:** The app receives the signed nonce and verifies it using the user's public key (derived from their wallet address). If the signature is valid, it confirms the user controls the wallet.
5. **Authenticate User:** Once verified, the app logs in the user and associates their wallet address with their session or profile. The wallet address often serves as the unique user identifier.
6. **Session Management:** The app can use cookies, tokens (like JWTs), or smart contract events to manage sessions while interacting with the blockchain.

This method ensures secure, decentralized authentication without traditional usernames or passwords. Wallets like Nami and Vespr streamline this process by providing user-friendly interfaces for signing and verifying data.

We'll be testing with both those wallets, and the screenshots you're going to see in the documentation will be from one of those two wallets (and especially on mobile phones, with Vespr).

Swim Lane Diagram

This diagram explains how this API backend interacts with the blockchain and the L4VA Smart Contracts

