**VoteSecure - JoyID Integration Setup**

This guide explains how to integrate the CKB Service Bridge with the VoteSecure organizer interface.

**Architecture**

votesecure/

├─ package.json

├─ .gitignore

├─ index.html

├─ src/

│ ├─ blockchain.js (VoteSecure blockchain logic)

│ └─ ckbServiceBridge.js (JoyID + CKB integration)

├─ web/

│ ├─ voter.html

│ ├─ organizer.html (Updated)

│ ├─ voter.js

│ ├─ organizer.js (Updated)

│ └─ votesecure.css

└─ node\_modules/

**Step 1: Install Dependencies**

Create or update your package.json:

{

"name": "votesecure",

"version": "1.0.0",

"type": "module",

"scripts": {

"dev": "vite",

"build": "vite build",

"preview": "vite preview"

},

"dependencies": {

"@joyid/ckb": "^0.3.0",

"@ckb-lumos/lumos": "^0.22.0"

},

"devDependencies": {

"vite": "^5.0.0"

}

}

Install:

npm install

**Step 2: Create ckbServiceBridge.js**

Place the ckbServiceBridge.js file in src/ directory. This file:

* Imports JoyID and Lumos libraries
* Provides wallet connection functions
* Exposes window.CKBService API for vanilla JS

**Key Features:**

* ✅ ES6 Module format
* ✅ Browser-compatible
* ✅ Exposes global API via window.CKBService
* ✅ Handles JoyID connection
* ✅ Manages CKB transactions

**Step 3: Update HTML Loading Order**

In organizer.html, ensure scripts are loaded in this order:

<!-- Load as ES Module (contains imports) -->

<script type="module" src="../src/ckbServiceBridge.js"></script>

<!-- Load blockchain logic -->

<script src="../src/blockchain.js"></script>

<!-- Load organizer UI -->

<script src="organizer.js"></script>

**Important:** The type="module" attribute is CRITICAL for ckbServiceBridge.js because it uses ES6 imports.

**Step 4: Configuration**

Edit src/ckbServiceBridge.js to configure your network:

// Configuration

const DEBUG\_LOG = true; // Enable console logging

const USE\_MAINNET = false; // false = testnet, true = mainnet

const USE\_PRIVATE\_NODE = true; // Use your private node

// Update these if using private node

const RPC\_URL = USE\_PRIVATE\_NODE

? 'http://192.168.178.94:8112' // Your testnet node

: 'https://testnet.ckb.dev/rpc'; // Public testnet

**Step 5: Development Server**

Run a local development server (required for ES modules):

# Using Vite (recommended)

npm run dev

# Or using Python

python -m http.server 8080

# Or using Node

npx serve

Then open: http://localhost:5173/web/organizer.html (or your port)

**Step 6: Testing the Integration**

**Check Service Status**

Open browser console (F12) and verify:

// Should see:

// ✓ CKB Service Bridge loaded successfully

// Network: testnet

// RPC: http://192.168.178.94:8112

// Indexer: http://192.168.178.94:8112

// Check if service is ready

console.log(window.CKBService);

// Should show: { connectJoyID: ƒ, signAndSendTransaction: ƒ, ... }

**Connect Wallet**

1. Click "Connect JoyID" button
2. JoyID popup should appear
3. Authenticate with your method (passkey/email)
4. Wallet info should display with balance

**Debug Interface**

Access debugging tools in console:

// Check current state

window.VoteSecureOrganizer.currentOrganizer

// { address: "ckb1...", balance: "100.00000000", network: "testnet" }

// Manually refresh balance

await window.VoteSecureOrganizer.refreshBalance()

// Show notification

window.VoteSecureOrganizer.showNotification('Test message', 'success')

**Troubleshooting**

**1. "CKBService is not defined"**

**Problem:** Module hasn't loaded yet

**Solution:**

* Ensure type="module" on script tag
* Check browser console for import errors
* Verify file paths are correct
* Make sure you're running a dev server (not file://)

**2. "Cannot use import statement outside a module"**

**Problem:** Script loaded without type="module"

**Solution:**

<!-- WRONG -->

<script src="../src/ckbServiceBridge.js"></script>

<!-- CORRECT -->

<script type="module" src="../src/ckbServiceBridge.js"></script>

**3. CORS Errors**

**Problem:** Loading from file:// protocol

**Solution:** Always use a development server:

npm run dev

# or

python -m http.server 8080

**4. JoyID Popup Doesn't Appear**

**Problem:** Network configuration or popup blocked

**Solution:**

* Check browser allows popups
* Verify network settings in ckbServiceBridge.js
* Check console for errors
* Ensure internet connection (JoyID needs to reach servers)

**5. Balance Shows 0.00000000**

**Problem:** Node connection or no funds

**Solution:**

* Verify node is running: curl http://192.168.178.94:8112
* Check if address has funds on explorer
* Try public node by setting USE\_PRIVATE\_NODE = false

**6. Service Status Stays "Loading"**

**Problem:** Module failed to load or initialize

**Solution:**

// Check in console:

console.log(window.CKBService);

// If undefined, check:

// 1. Network tab for failed loads

// 2. Console for import errors

// 3. File paths in HTML

**Network Configuration**

**Using Public Testnet**

const USE\_MAINNET = false;

const USE\_PRIVATE\_NODE = false;

**Using Private Testnet Node**

const USE\_MAINNET = false;

const USE\_PRIVATE\_NODE = true;

// Update RPC\_URL to your node

**Using Mainnet (Production)**

const USE\_MAINNET = true;

const USE\_PRIVATE\_NODE = false;

// ⚠️ WARNING: Real CKB tokens!

**API Reference**

**window.CKBService**

The global API exposed by the bridge:

// Connect JoyID wallet

const wallet = await window.CKBService.connectJoyID();

// Returns: { address, balance, network }

// Get balance

const balance = await window.CKBService.getSpendableCapacityShannons(address);

// Returns: BigInt (shannons)

// Convert shannons to CKB

const ckb = window.CKBService.shannons2CKB(balance);

// Returns: String (e.g., "100.00000000")

// Send transaction

const txHash = await window.CKBService.signAndSendTransaction(

fromAddress,

toAddress,

amountCKB

);

// Returns: String (transaction hash)

// Configuration

console.log(window.CKBService.config);

// { USE\_MAINNET, USE\_PRIVATE\_NODE, RPC\_URL, ... }

**File Structure Details**

**ckbServiceBridge.js**

* **Type:** ES6 Module
* **Imports:** @joyid/ckb, @ckb-lumos/lumos
* **Exports:** window.CKBService (global)
* **Purpose:** Bridge between TypeScript/React and vanilla JS

**organizer.js**

* **Type:** Vanilla JavaScript
* **Depends on:** window.CKBService, window.VoteSecureBlockchain
* **Purpose:** UI logic for election creation

**organizer.html**

* **Loads:** ckbServiceBridge.js (module), blockchain.js, organizer.js
* **Purpose:** Organizer interface

**Best Practices**

1. **Always use a dev server** - Never open HTML files directly
2. **Check console regularly** - Catch errors early
3. **Test on testnet first** - Never test with real funds
4. **Handle errors gracefully** - Network issues are common
5. **Keep dependencies updated** - Check for security updates

**Next Steps**

1. ✅ Install dependencies
2. ✅ Create ckbServiceBridge.js
3. ✅ Update organizer.html
4. ✅ Update organizer.js
5. ✅ Configure network settings
6. ✅ Start dev server
7. ✅ Test wallet connection
8. 🎯 Create your first election!

**Support**

If you encounter issues:

1. Check browser console (F12) for errors
2. Verify all files are in correct locations
3. Ensure dev server is running
4. Check network connectivity
5. Review this guide's troubleshooting section

**Production Deployment**

When ready for production:

# Build optimized bundle

npm run build

# Output will be in dist/ folder

# Deploy dist/ to your web server

**Security Notes:**

* Never expose private keys
* Use environment variables for sensitive config
* Enable HTTPS in production
* Audit all dependencies regularly

**Questions?** Check the console logs - they provide detailed information about service initialization and connection status.