**Walkers API Documentation**



**Walkers API Documentation**

[Verify Cardano Wallet Ownership 3](#_mdvczsrk5365)

[Obtaining Input Data 3](#_nsvetug89coz)

[API Call 4](#_fhbh66qt6gpz)

[Endpoint 4](#_3ajy8wbsz2pl)

[Method 4](#_hr1yy51jle2v)

[Body 4](#_r2oe3khpk8xi)

[Returns 4](#_sy83p78y1gcn)

[Examples 4](#_y9wka8b95ee5)

[Suggested Development Flow 5](#_3zbw70m15gc2)

[Obtain a User’s Cardano Assets 7](#_z484dk2cyjzr)

[Obtaining Input Data 7](#_n7y4ip4u176x)

[API Call 7](#_gpyiayp7ho0f)

[Endpoint 7](#_xmh69qm07elm)

[Method 7](#_hue0dy3bn5hq)

[Returns 7](#_4wuqkt2wuw8w)

[Examples 8](#_79te7ts3spil)

[Exchange Walker Points for Cardano Native Assets 9](#_u04n9zpg8hv8)

[Obtaining Input Data 9](#_nohlo1qbietp)

[API Call 9](#_hrhcu8nu0gg8)

# 

# Verify Cardano Wallet Ownership

## Obtaining Input Data

*First, create a new WalletConnector, the JavaScript class provided by Yepple.*

const walletConnector = new WalletConnector();

*Next, get the available wallets. On Cardano, wallets are identified by a string.*

const installedWallets = await walletConnector.getAvailableWallets();

// Example output

[

{ id: ‘eternl’, name: ‘Eternl’, icon: ‘https://…’ },

{ id: ‘nami’, name: ‘Nami’, icon: ‘https://…’ },

{ id: ‘flint’, name: ‘Flint’, icon: ‘https://…’ }

]

*Provide a “Connect Wallet” button for a User to click with each option from above and get the selected “id” of the wallet*

await wallet.connect(walletId);

// Example: const connected = await walletConnector.connect(‘eternl’);

*Obtain the stake address*

const stakeAddress = await walletConnector.getStakeAddress();

*Select an arbitrary payload that the user will sign*

const payload = ‘Sign into Walkers app!’;

*Obtain a wallet signature for the payload*

const walletSignature = await walletConnector.signMessage(payload);

Note: Currently, hardware wallets on Cardano do not support wallet signatures as there is an inherent risk with user hardware wallets for any dApps. Users are encouraged to use online wallets for connecting with the Walkers application.

## API Call

### Endpoint

https://client-api.yepple.io/v1/apps/walkers/wallet/verify

### Method

POST

### Body

| Name | Required | Type | Description | Example |
| --- | --- | --- | --- | --- |
| stakeAddress | Yes | string | The wallet’s stake address | “stake1234” |
| walletSignature | Yes | N/A | The cryptographic wallet signature generated by the user. | See examples above and below |
| payload | Yes | string | Arbitrary signature payload (displayed to user) | “Sign in to Walkers App!” |

### Returns

200 - Success

Body - { verified: boolean; } // Whether or not the signature was valid for the payload

All Other Statuses - Error

Body - { message: string; } // a message describing the error

### Examples

Using cURL:

curl -X POST [https://client-api.yepple.io/v1/apps/walkers/wallet/](https://client-api.yepple.io/v1/apps/walkers/assets/stake1234)verify -H “Content-Type: application/json”

Using Node.js:

await fetch('[https://client-api.yepple.io/v1/apps/walkers/wallet/](https://client-api.yepple.io/v1/apps/walkers/assets/:stakeAddress)verify, {

method: ‘POST’,

headers: {

‘Content-Type’: ‘application/json’

},

body: JSON.stringify({

stakeAddress,

walletSignature,

payload

})

})).then(async res => {

if (!res.ok) {

const errBody = await res.json();

console.log(`(${res.status}) Unable to verify wallet: ${errBody.message}`);

return;

}

return res.json();

}).then(jsonRes => {

console.log(‘The wallet verification was ‘ + (jsonRes.verified ? ‘successful!’ : ‘unsuccessful!’));

});

## Suggested Development Flow

The utility of wallet verification is that it allows users within an ecosystem to obtain in-game rewards based on what they are holding in their wallets. The above documentation allows any system to gather the wallet signature of a user and verify its integrity. The method in which this verification is used depends on the system implementing it. In general the following data flow and best practices are recommended:

Data Flow

1. User connects to platform
2. User logs in to existing account
3. To begin the process of adding a wallet to the account, the user clicks a button to initiate a wallet signature
4. The wallet signature is obtained and sent to the system’s server
5. The system’s server uses the Yepple API to verify the integrity of the wallet signature. Once this is done, the system can be confident that the user owns the wallet
6. The wallet is added to the user’s account in the system’s database

Best Practices

1. We recommend that you only let a wallet be owned by one account at a time. Letting the same wallet be used by multiple accounts could lead to duplicate rewards for the same assets

# Obtain a User’s Cardano Assets

## Obtaining Input Data

**Use wallet connector to get the user’s Stake Address (stakeAddress)**

*First, create a new WalletConnector, the vanilla JavaScript class provided by Yepple.*

const walletConnector = new WalletConnector();

*Next, get the available wallets. On Cardano, wallets are identified by a string.*

const installedWallets = await walletConnector.getAvailableWallets();

// Example output

[

{ id: ‘eternl’, name: ‘Eternl’, icon: ‘https://…’ },

{ id: ‘nami’, name: ‘Nami’, icon: ‘https://…’ },

{ id: ‘flint’, name: ‘Flint’, icon: ‘https://…’ }

]

*Provide a “Connect Wallet” button for a User to click with each option from above and get the selected “id” of the wallet*

await wallet.connect(walletId);

// Example: const connected = await wallet.connect(‘eternl’);

const stakeAddress = await wallet.getStakeAddress();

## API Call

### Endpoint

https://client-api.yepple.io/v1/apps/walkers/assets/:stakeAddress

### Method

GET

### Returns

200 - Success

Body - {

[policyId: string]: {

[assetId: string]: number;

}

}

Anything other than 200 - Error

Body - { message: string; } // a message describing the error

### Examples

Using cURL:

curl -X GET [https://client-api.yepple.io/v1/apps/walkers/assets/](https://client-api.yepple.io/v1/apps/walkers/assets/stake1234)stake1ux54zadu58skngn3z4fydrx6jf8368zm606dqnqvqfzs6ssgy0gza

Using Node.js:

const assets = await fetch('[https://client-api.yepple.io/v1/apps/walkers/assets/](https://client-api.yepple.io/v1/apps/walkers/assets/:stakeAddress)stake1ux54zadu58skngn3z4fydrx6jf8368zm606dqnqvqfzs6ssgy0gza).then(async res => {

if (!res.ok) {

const errBody = await res.json();

console.log(`(${res.status}) Unable to obtain assets: ${errBody.message}`);

}

return res.json();

});

# Exchange Walker Points for Cardano Native Assets

## Obtaining Input Data

Determine the amount of input tokens and the native asset to be exchanged.

Trades will be entered into the database, and the Cardano transaction processor will regularly scan for exchange requests.

*Coming Soon*

## API Call

*Coming Soon*