Why do you want to use Scatter ?

To make a trnasaction on a blockchain, we need to have the user's private key to sign the trasanction with. One of the downside of web beig so open is that we can't trust a third party with such sensitive peice of information. But then how do we authorize apps to trasact on our behalf without compromising our private keys.

This where Scatter comes into picture. User's define there private, public key and their personal information like firstname, last name, address in Scatter app running locally on there machine. ScatterJS is a web client library that allows a website to seek permission on user behalf and perform transactoins, without knowing the user's private key.

I like me, if you are coming from a Web2.0 background, the best way to understand Scatter will be a quick comparison with OAuth. When we wan't to login into a website using our google/facebook accounts, we don't share our original password with the site. Instead, we allow the website to link to our account with a defined set of permissions, and we trust Google and Facebook to take care of security for us.

Just like OAuth is the defacto for identity management, authorisation and authentication for web 2.0, currently Scatter app is for Ethereum and EOS blockchains.

Similarities with OAuth

- Acts as third party authority that brokers identity management and authority between user and an app
- Like OAuth, user controls what information an permissions they allow for an app

Difference with OAuth

- Unlike AOuth, user can always see which app they have linked.
- For each action (transaction), user explicitly reviews and approves in Scatter app.
- No central authority like Google or Facebook that holds and manages the user identity and permission. In this case, user has way more control.

For e.g. when we need to identity a user on a website and seek authority to act on behalf of user (like tweet, post on Facebook, get google data), we use scatter for doing the same on blockchains.

Unlike OAuth, there is no central authority like Google or Facebook that holds and manages the user identity and permission. Scatter run on each users machine and each user has way more control about which apps can see their details. Also each transaction that we perform on behalf of user, has to be explictly approved by the user in Scatter app.

Installing dependcies

```
npm install --save eosjs
npm install --save scatterjs-core
npm install --save scatterjs-plugin-eosjs
```

Intializing scatter and plugins

```
import ScatterJS from 'scatterjs-core';
import ScatterEOS from 'scatterjs-plugin-eosjs';
import Eos from 'eosjs';
ScatterJS.plugins( new ScatterEOS() );
```

Scatter interaction flow

1. App connects with Scatter useing Scatterjs
    - This fails if user does not has Scatter installed or if Scatter is locked on user's machine
2. Link app with Scatter (login user) -
    - Define a set of permissions that we seek from scatter on behalf of user
    - User sees prompt in Scatter to grant these perissions to our app
3. Perform transactions
    - Every transaction will need to be signed by user via the scatter app

Step 1 : Connect to scatter

In this step we try to connect to the Scatter instance running on user's machine. This will fail if user does not have scatter installed or if hasn't unlocked Scatter app.

```
export const connect = appName => (new Promise((resolve, reject)=> {
    ScatterJS.scatter.connect(appName).then(connected => {
        const
```

```
            onSuccess = () => {
                scatter = ScatterJS.scatter;
                resolve();
            },
            onError = () => reject({
                message: "Scatter not found. Please install and unlock
scatter"
            });

        connected ? onSuccess() : onError();
    });
}));
```

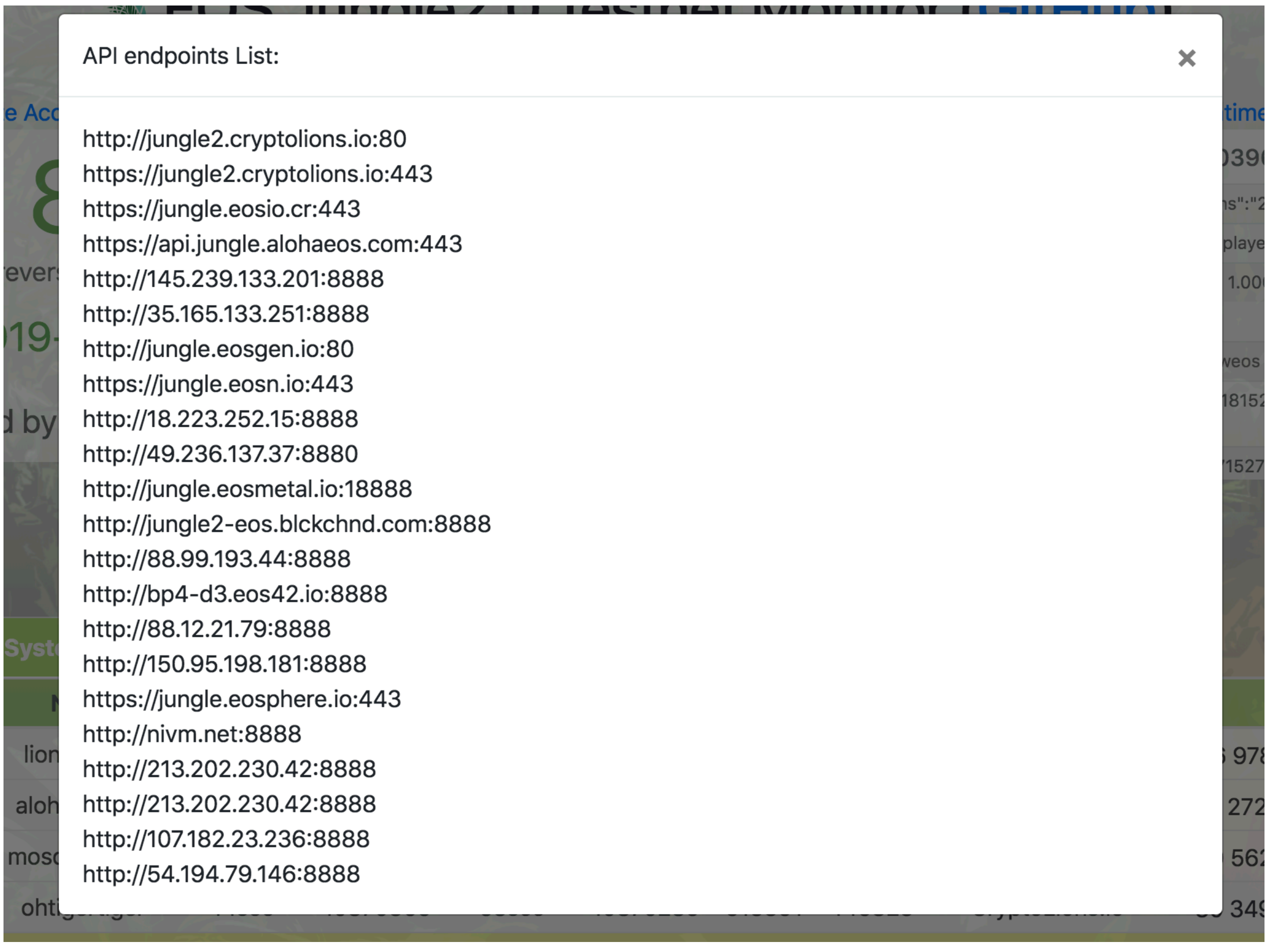
Step 2 : Get endpoints and chain-id of networks that we want to connect to

Now we seek user's permission to view who they are on the designated network. The network could be Ethereum, Tron, EOS mainnet of a testnet like jungle. For this example we will use **Jungle Testnet**.

First we need to define this networks that we want to connect to on behalf of user :

```
// We can have as many networks as we like here
// Based on wether user has network defined in there Scatter app instance,
our app will be  granted permssion for the network

const networks = [{
    blockchain:'eos',

 chainId:'e70aaab8997e1dfce58fbfac80cbbb8fecec7b99cf982a9444273cbc64c41473',
    host:'jungle.eosmetal.io',
    port:18888,
    protocol:'http'
}];
```

In this example we have asked permission of **Jungle Testnet**. For each network that you want user's account on, you can create simlare stuctuer in array. To get these details for Jungle net go to https://monitor.jungletestnet.io and click on **API endpoints**, you will see details like

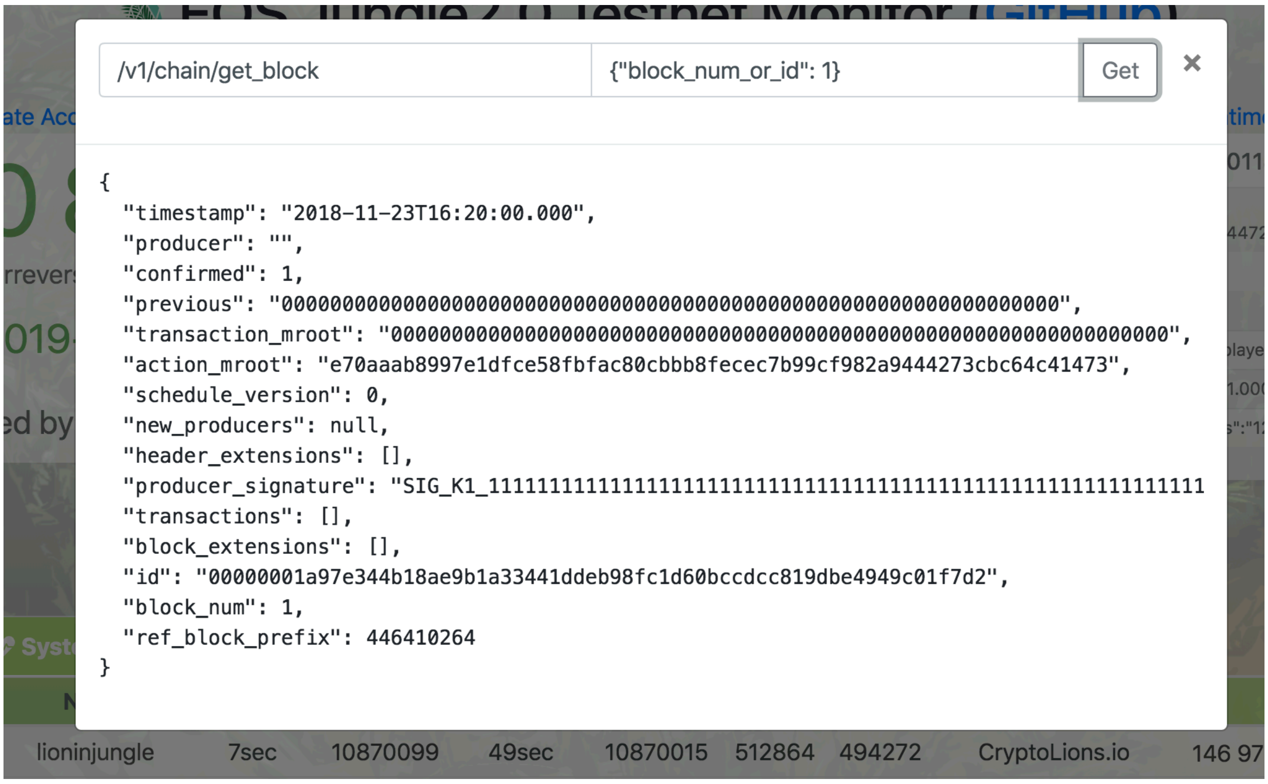


API endpoints List:

×

```
http://jungle2.cryptolions.io:80
https://jungle2.cryptolions.io:443
https://jungle.eosio.cr:443
https://api.jungle.alohaeos.com:443
http://145.239.133.201:8888
http://35.165.133.251:8888
http://jungle.eosgen.io:80
https://jungle.eosn.io:443
http://18.223.252.15:8888
http://49.236.137.37:8880
http://jungle.eosmetal.io:18888
http://jungle2-eos.blckchnd.com:8888
http://88.99.193.44:8888
http://bp4-d3.eos42.io:8888
http://88.12.21.79:8888
http://150.95.198.181:8888
https://jungle.eosphere.io:443
http://nivm.net:8888
http://213.202.230.42:8888
http://213.202.230.42:8888
http://107.182.23.236:8888
http://54.194.79.146:8888
```

Apart from these details we also need the **chainId**, to get this click on **API** link and get information for **/v1/chain/get_block**.



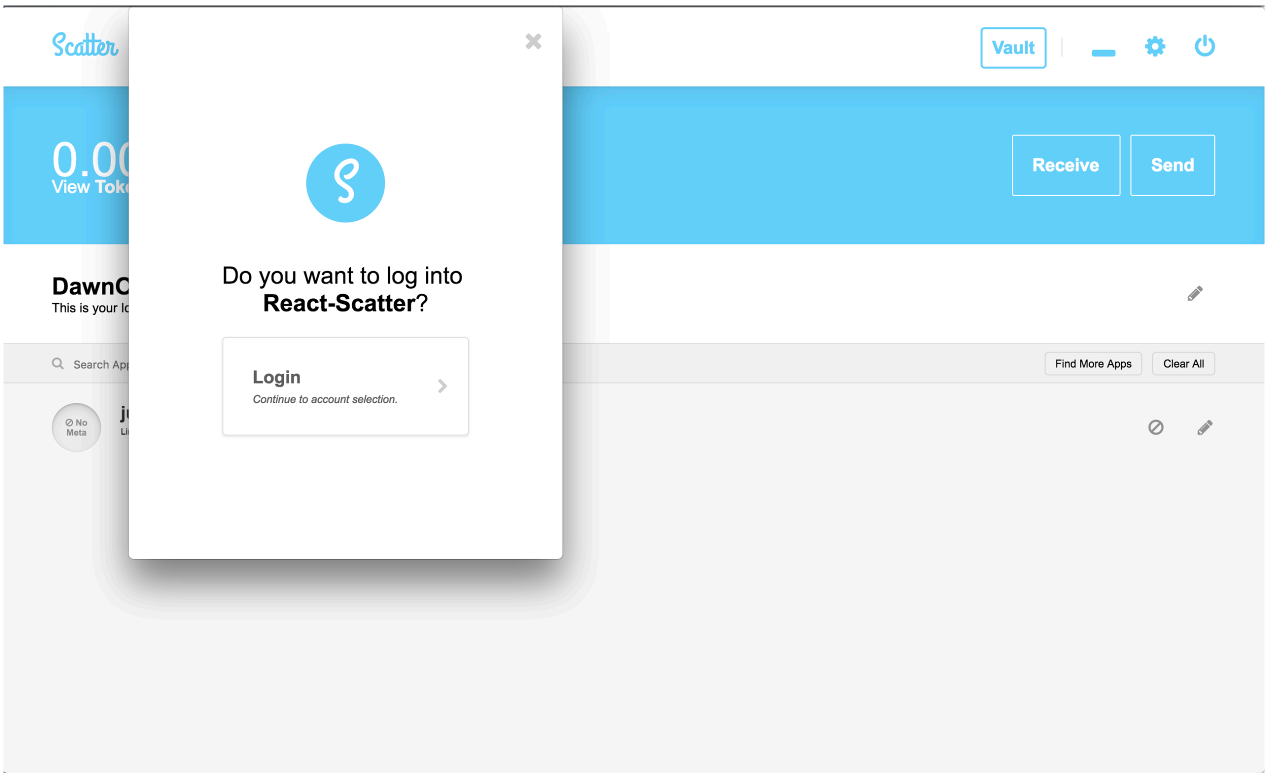
/v1/chain/get_block | {"block_num_or_id": 1} | Get ×

```
{
  "timestamp": "2018-11-23T16:20:00.000",
  "producer": "",
  "confirmed": 1,
  "previous": "0000000000000000000000000000000000000000000000000000000000000000",
  "transaction_mroot": "0000000000000000000000000000000000000000000000000000000000000000",
  "action_mroot": "e70aaab8997e1dfce58fbfac80cbbb8fecec7b99cf982a9444273cbc64c41473",
  "schedule_version": 0,
  "new_producers": null,
  "header_extensions": [],
  "producer_signature": "SIG_K1_11111111111111111111111111111111111111111111111111111111111
  "transactions": [],
  "block_extensions": [],
  "id": "00000001a97e344b18ae9b1a33441ddeb98fc1d60bccdcc819dbe4949c01f7d2",
  "block_num": 1,
  "ref_block_prefix": 446410264
}
```

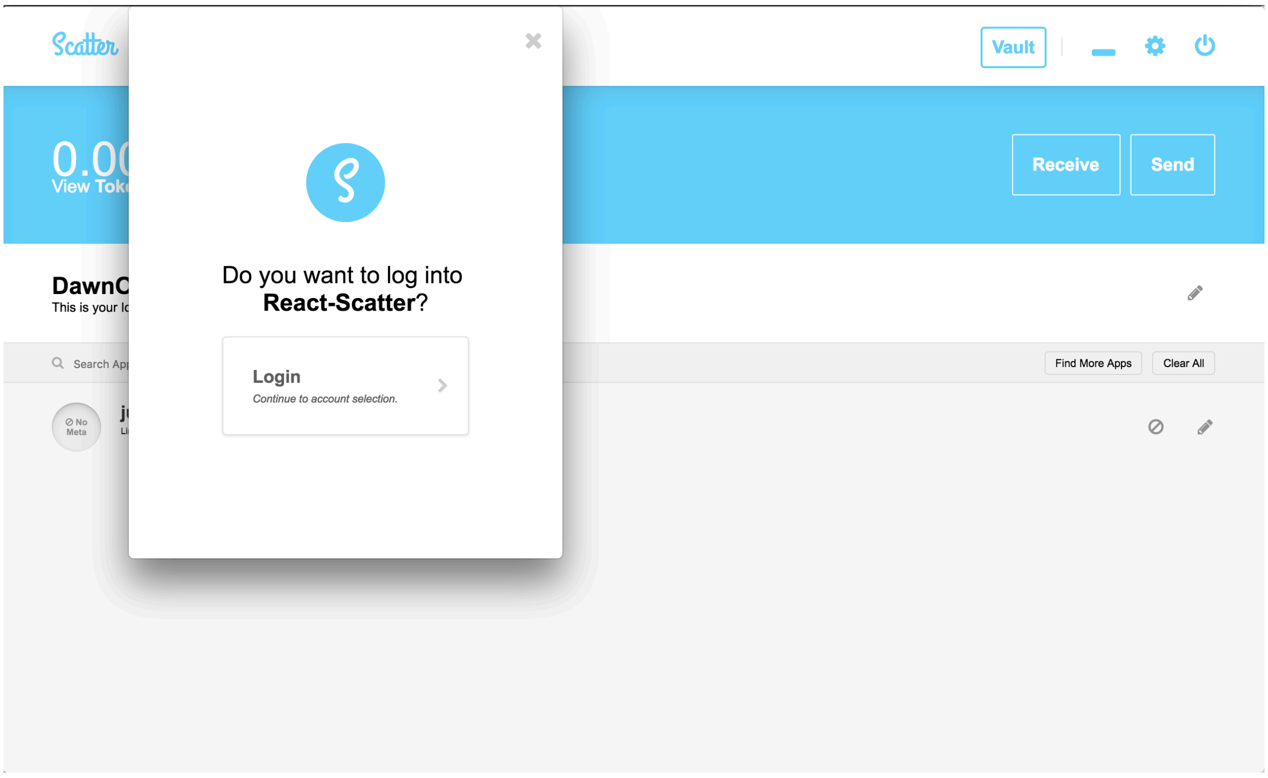
Step 3 : Ask user to allow us to access there details on chosen networks

```
export const login = ()=> {
    // Can have more required fields like firstname, lastname, address
    const requiredFields = { accounts:networks };

    return scatter.getIdentity(requiredFields).then(() => {
        // Get EOS chains from the networks defined in user's scatter app
and save this object in memory for future reference
        userAccount = scatter.identity.accounts.find(x => x.blockchain ===
'eos');

        const eosOptions = { expireInSeconds: 60 };

        // Create an EOS connection using which we can request user to sign
transaction using Scatter
        userEosConnection = scatter.eos(network, Eos, eosOptions);
        return {
            name: userAccount.name,
            authority: userAccount.authority,
            publicKey: userAccount.publicKey
        };
    });
};
```

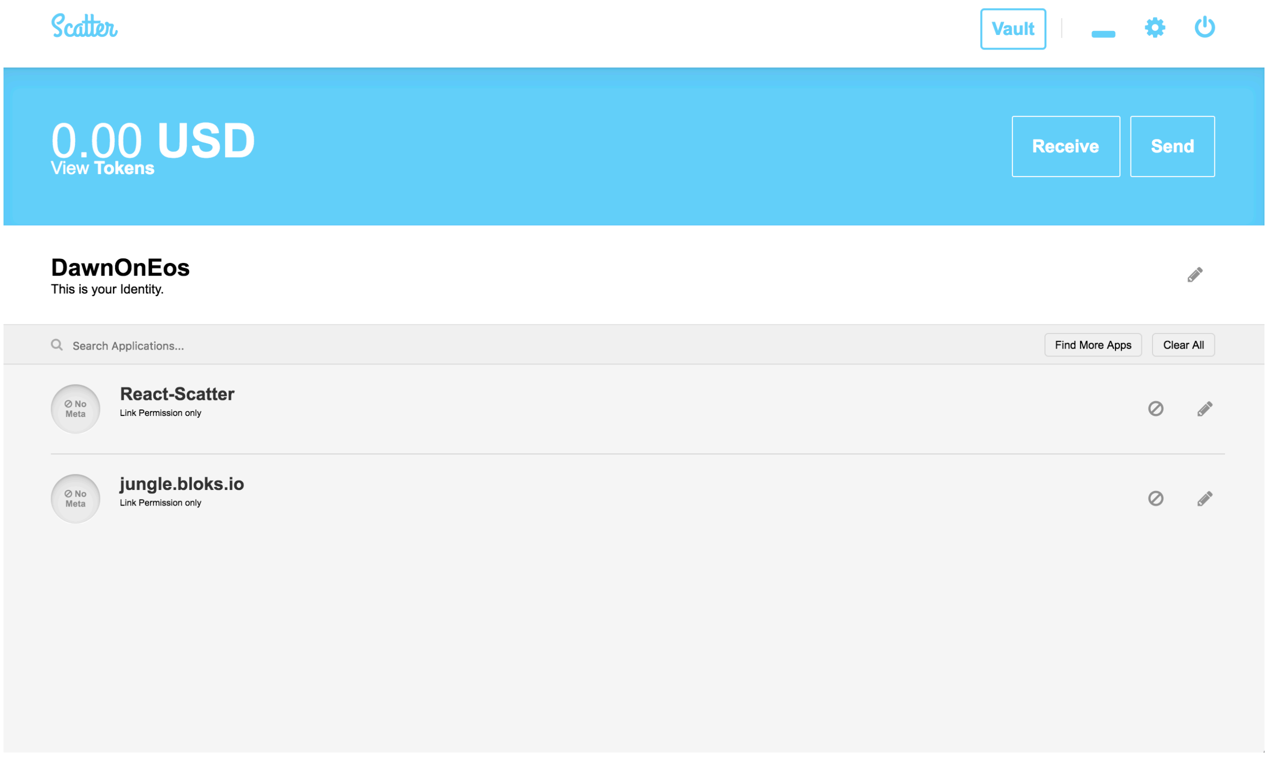
When above code runs, use sees a prompt in scatter app :



Now user can select which public key they want for our app to get :



Once user has approved linking our app to scatter, user can see our app on his home screen and unlink at anytime they want to.



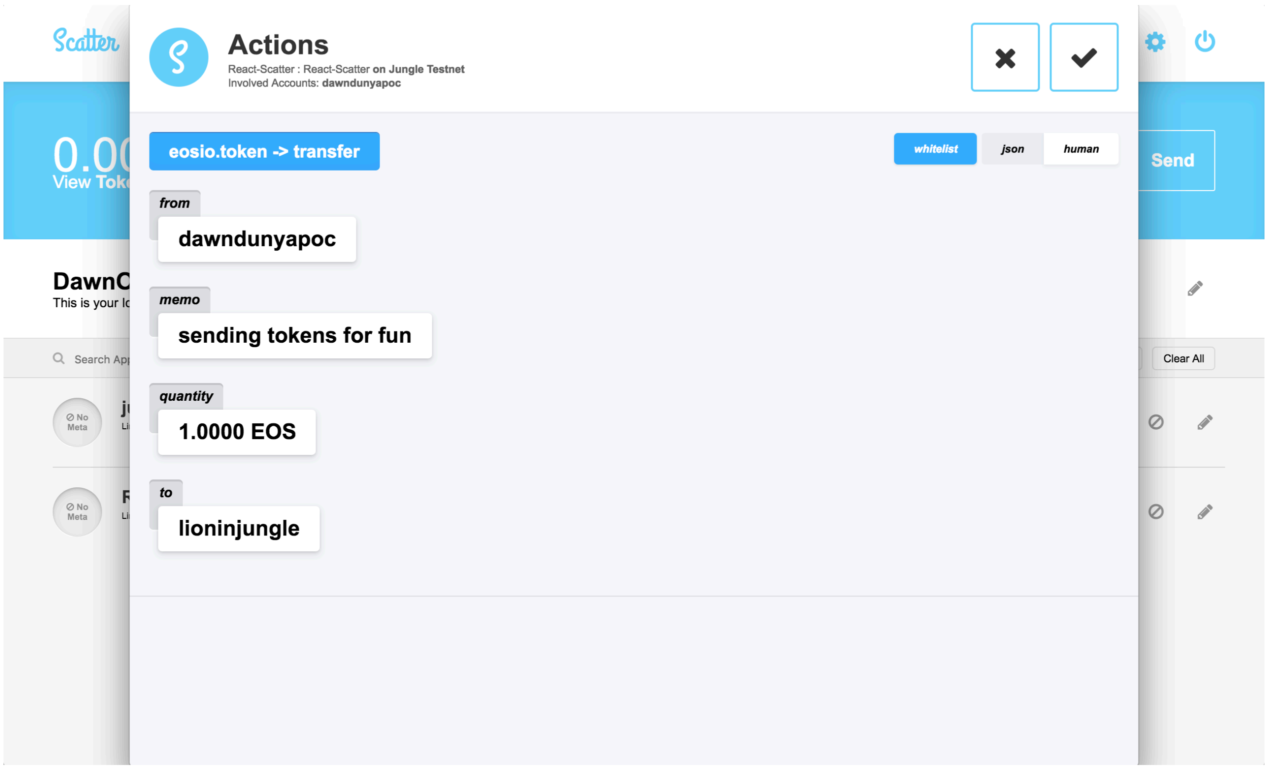
Now we have the user's account name, public key on the jungle testnet.

Step 4 : Perform transactions on the network

Even though we still do not have the user's private key, we can still intiate transactions using the ScatterJS , at which point user will se a prompt in Scatter app with details of the trasaction. Use can aprove/reject the transaction as he likes.

```
export const sendTokens = ({toAccount, amount, memo}) => {
    const transactionOptions = { authorization:
[`${userAccount.name}@${userAccount.authority}`] };
    return userEosConnection.transfer(
        userAccount.name,
        toAccount,
        amount,
        memo,
        transactionOptions
    ).then(trx => {
        return trx.transaction_id;
    }).catch(error => {
        console.error(error);
    });
};
```

This is how user's sees the request to sign this transaction :



Actions

React-Scatter : React-Scatter on Jungle Testnet
Involved Accounts: dawndunyapoc

eosio.token -> transfer

whitelist json human

from
dawndunyapoc

memo
sending tokens for fun

quantity
1.0000 EOS

to
lioninjungle

Once user accepts the action, our transaction is pushed to the chain. Thats it.

Resources :

- Setting up scatter with Jungle testnet : https://www.youtube.com/watch?v=6Yf-cHg4k90
- Jungle testnet : https://monitor.jungletestnet.io/#home
- ScatterJS : https://github.com/GetScatter/scatter-js