# EVTJS

General purpose API Binding for the everiToken blockchain. Supports `node` and `browser`.

## Install

For NodeJS, use `npm`:

```
npm install evtjs --save
```

Or use `yarn`:

```
yarn add evtjs
```

For browser, you may download the source code and compile it for browsers in the root directory:

```
npm run build_browser
```

Then you'll find produced `evt.js` in `dist` folder.

You can also download our release package and reference it to use the library in browser.

## Basic Usage

```
// set network endpoint
const network = {
    host: 'testnet1.everitoken.io', // For everiToken Aurora 2.0
    port: 8888,                     // defaults to 8888
    protocol: 'https'               // the TestNet of everiToken uses SSL
};

// get APICaller instance
const apiCaller = EVT({
    // keyProvider should be string of private key (aka. wit, can generate from
everiSigner)
    // you can also pass a function that return that string (or even
Promise<string> for a async function)
    endpoint: network,
    keyProvider: 'xxxxxxxxxxxxxxxxxxxxxxxxxx'
});

// call API
var response = await apiCaller.getInfo();
```

```
// or if `await` is supported in current environment
apiCaller.getInfo()
    .then(res => {
        // TODO
    })
    .catch((e) => {
        // TODO
    });
```

## EvtKey Usage

`EvtKey` is a class for everiToken's key management.

### randomPrivateKey()

You can get a random private key by `EVT.EvtKey.randomPrivateKey`:

```
// randomPrivateKey returns a promise so we should use await or 'then'
let key = await EVT.EvtKey.randomPrivateKey();

// now key is the private key, that is , a wit.
```

### privateToPublic(privateKey)

After generating a private key, you can convert it to a public key by `privateToPublic`.

**Parameters**

- `privateKey`: the private key you want to convert.

```
let publicKey = EVT.EvtKey.privateToPublic(key);
```

### seedPrivateKey(seed)

Get a private key from a specific `seed`.

**Parameters**

- `seed`: The `seed` is a string. For private key's safety, the `seed` must be random enough and must have at least 32 bytes' length.

```
let privateKey = EVT.EvtKey.seedPrivateKey('AVeryVeryVeryLongRandomSeedHere');
```

### isValidPrivateKey(key) / isValidPublicKey(key)

You can use `isValidPrivateKey` or `isValidPublicKey` to check a key.

**Parameters**

- `key`: The private / public key you want to check.

```
assert(EVT.EvtKey.isValidPrivateKey('5J1by7KRQujRdXrurEsvEr2zQGcdPaMJRjewER6XsAR2e
Ccpt3D'), 'should be a valid private');

assert(EVT.EvtKey.isValidPublicKey('EVT6Qz3wuRjyN6gaU3P3XRxpnEZnM4oPxortemaWDwFRvs
v2FxgND'), 'should be a valid public');
```

### random32BytesAsHex() => Promise<string>

You may generate a 32-byte-long hex string and it is promised to be safe in cryptography.

> This is a `async` function. Please use `then` or `await` for the result accordingly.

### randomName128() => Promise<string>

Produces a safe string with a length of 21. This is suitable for use in ABI structure where it requires a `name128` type such as `proposalName` for a suspended transaction.

> This is a `async` function. Please use `then` or `await` for the result accordingly.

## APICaller Usage

### Initialization

Before calling APICaller, you must initialize a instance of APICaller and pass a `keyProvider` for it. You can use `EvtKey` to generate a valid one.

A APICaller object could be created like this:

```
// get APICaller instance
var apiCaller = EVT(args);
```

**Parameters**

- `args`: a object, the following fields are required:
  - `keyProvider`: keyProvider should be string representing private key, or a function which returns the private key or a `Promise` that will resolves with the private key for a async function
  - `endpoint`: a object to specify the endpoint of the node to be connected

Here are several example of `keyProvider`:

```
// keyProvider is the private key
let keyProvider = '5J1by7KRQujRdXrurEsvEr2zQGcdPaMJRjewER6XsAR2eCcpt3D';

// keyProvider is a function
let keyProvider = function() {
    return '5J1by7KRQujRdXrurEsvEr2zQGcdPaMJRjewER6XsAR2eCcpt3D';
}

// keyProvider is a async function
let keyProvider = function() {
    return new Promise((res, rej) => {
        res('5J1by7KRQujRdXrurEsvEr2zQGcdPaMJRjewER6XsAR2eCcpt3D');
    });
}
```

A example of `endpoint`:

```
// endpoint is a object
// example:
let endpoint = {
    host: 'testnet1.everitoken.io', // For everiToken Aurora 2.0
    port: 8888,                     // defaults to 8888
    protocol: 'https'               // the TestNet of everiToken uses SSL
};
```

### getInfo()

get basic information from block chain.

```
let info = await apiCaller.getInfo();
```

**Example Response**

```
{
    "server_version": "3813c635",
    "chain_id":
"bb248d6319e51ad38502cc8ef8fe607eb5ad2cd0be2bdc0e6e30a506761b8636",
    "evt_api_version": "1.2.0",
    "head_block_num": 145469,
    "last_irreversible_block_num": 145468,
    "last_irreversible_block_id":
"0002383c51a24696917c8e6ba24557a138510d7f73196d0b11d447fd7f4b6eb7",
    "head_block_id":
"0002383d5378cb6ba3d261c2df459e2413a211e8211a11a22cd614b18a293bb5",
    "head_block_time": "2018-06-05T04:56:29",
    "head_block_producer": "evt",
    "recent_slots": "",
    "participation_rate": "0.00000000000000000"
}
```

Some important return values are as follow:

- `evt_api_version`: This is important and *must be verified before any other call* .

  Format: [ major.minor.modification ]. When `major` is changed then you could not use the chain's API. It is not compatible with you.

- `head_block_*` and `last_irreversible_block_*`: Represents information for last block / last irreversible block. This is used by other calls automatically by the library.

### getOwnedTokens(publicKeys)

Get the list of tokens which is owned by `publicKeys`.

> Make sure you have history_plugin enabled on connected node

**Parameters**

- `publicKey`: an array or a single value which represents public keys you want to query

**Response**

The response is an array representing the list of tokens belonging to public keys provided. Each token is identified by the `name` and the `domain` it belongs to.

A example:

```
[
    {
        "name": "T39823",
        "domain": "test"
    }
]
```

### getManagedGroups(publicKeys)

Get the list of groups which is managed by `publicKeys`.

> Make sure you have history_plugin enabled on connected node

**Parameters**

- `publicKey`: an array or a single value which represents public keys you want to query

**Response**

The response is an array representing the list of groups managed by public keys provided. Each group is identified by the `name`.

A example:

```
[
    {
        "name": "testgroup"
    }
]
```

## getCreatedDomains(publicKeys)

Get the list of domains which is created by `publicKeys`.

> Make sure you have history_plugin enabled on connected node

**Parameters**

- `publicKey`: an array or a single value which represents public keys you want to query

**Response**

The response is an array representing the list of domains created by public keys provided. Each domain is identified by the `name`.

A example:

```
[
    {
        "name": "testdomain"
    }
]
```

## getActions(params)

Get the list of actions. Supports filtering by `domain`, `key`, and `action name`.

> Make sure you have history_plugin enabled on connected node

**Parameters**

- `params`: a object with the follow members:
    - `domain`: The domain to filter the result. It is required. Some special domains are supported, such as `fungible`
    - `key`: The key to filter the result. The value of it is the same as you provided one in `pushTransaction`. Optional.

  - `names`: an array to filter the action name. Optional.
  - `skip`: The count to skip in the result. This is for paging.
  - `take`: The count to take after skipping. This is for paging.

**Response**

The response is an array representing the list of actions.

A example:

```
[{
    "name": "newfungible",
    "domain": "fungible",
    "key": "EVT",
    "trx_id": "f0c789933e2b381e88281e8d8e750b561a4d447725fb0eb621f07f219fe2f738",
    "data": {
      "sym": "5,EVT",
      "creator": "EVT8MGU4aKiVzqMtWi9zLpu8KuTHZWjQQrX475ycSxEkLd6aBpraX",
      "issue": {
        "name": "issue",
        "threshold": 1,
        "authorizers": [{
            "ref": "[A] EVT8MGU4aKiVzqMtWi9zLpu8KuTHZWjQQrX475ycSxEkLd6aBpraX",
            "weight": 1
          }
        ]
      },
      "manage": {
        "name": "manage",
        "threshold": 1,
        "authorizers": [{
            "ref": "[A] EVT8MGU4aKiVzqMtWi9zLpu8KuTHZWjQQrX475ycSxEkLd6aBpraX",
            "weight": 1
          }
        ]
      },
      "total_supply": "100000.00000 EVT"
    }
  }
]
```

### getFungibleBalance(address, [symbol])

Get the list of fungible balance of a user.

> Make sure you have history_plugin enabled on connected node

**Parameters**

- `address`: The address (public key) you want to query.

- `symbol`: The symbol you want to query, optional. For example: "5,EVT".

**Response**

The response is an array representing the list of balance.

A example:

```
[
  "2.00000 EVT", "1.00000 PEVT"
]
```

### getTransactionDetailById(id)

Get detail information about a transaction by its `id`.

> Make sure you have history_plugin enabled on connected node

**Parameters**

- `id`: The id to query. `pushTransaction` will return the id of the transaction it created.

**Response**

The response is the detail information about a transaction.

A example:

```
{
  "id": "f0c789933e2b381e88281e8d8e750b561a4d447725fb0eb621f07f219fe2f738",
  "signatures": [

"SIG_K1_K6hWsPBt7VfSrYDBZqCygWT8dbA6R3mpxKPjd3JUh18EQHfU55eVEkHgq8AR5odWjPXvYasZQ1
LoNdaLKKhagJXXuXp3Y2"
  ],
  "compression": "none",
  "packed_trx":
"bb72345b050016ed2e620001000a13e9b86a6e7100000000000000000000d0d550520646000000000
00000000000000000040beabb0010545565400000000003c7e3ff0060d848bd31bf53daf1d5fed7d82
c9b1121394ee15dcafb07e913a97000000008052e74c0100000000101000000003c7e3ff0060d848bd3
1bf53daf1d5fed7d82c9b1121394ee15dcafb07e913a970000000000000000000100000000094135c680
1000000010100000003c7e3ff0060d848bd31bf53daf1d5fed7d82c9b1121394ee15dcafb07e913a97
000000000000000010000e40b540200000005455654000000000",
  "transaction": {
    "expiration": "2018-06-28T05:31:39",
    "ref_block_num": 5,
    "ref_block_prefix": 1647242518,
    "delay_sec": 0,
    "actions": [{
        "name": "newfungible",
        "domain": "fungible",
        "key": "EVT",
        "data": {
          "sym": "5,EVT",
          "creator": "EVT8MGU4aKiVzqMtWi9zLpu8KuTHZWjQQrX475ycSxEkLd6aBpraX",
          "issue": {
            "name": "issue",
            "threshold": 1,
            "authorizers": [{
                "ref": "[A]
EVT8MGU4aKiVzqMtWi9zLpu8KuTHZWjQQrX475ycSxEkLd6aBpraX",
                "weight": 1
              }
            ]
          },
          "manage": {
            "name": "manage",
            "threshold": 1,
            "authorizers": [{
                "ref": "[A]
EVT8MGU4aKiVzqMtWi9zLpu8KuTHZWjQQrX475ycSxEkLd6aBpraX",
                "weight": 1
              }
            ]
          },
          "total_supply": "100000.00000 EVT"
        },
        "hex_data":
"05455654000000000000003c7e3ff0060d848bd31bf53daf1d5fed7d82c9b1121394ee15dcafb07e913a
970000000080 52e74c01000000001010000000 03c7e3ff0060d848bd31bf53daf1d5fed7d82c9b11213
94ee15dcafb07e913a97000000000000000001000000009 4135c680100000000101000000003c7e3ff0
060d848bd31bf53daf1d5fed7d82c9b1121394ee15dcafb07e913a970000000000000000010000e40b5
4020000000545565400000000"
      }
    ],
    "transaction_extensions": []
  }
}
```

## getDomainDetail(name)

Get detail information about a domain by its `name`.

> Make sure you have history_plugin enabled on connected node

**Parameters**

- `name`: The name of the domain you want to query

**Response**

The response is the detail information of the domain you queried.

A example:

```
{
    "name": "cookie",
    "creator": "EVT6MRyAjQq8ud7hVNYcfnVPJqcVpscN5So8BhtHuGYqET5GDW5CV",
    "issue_time": "2018-06-09T09:06:27",
    "issue": {
        "name": "issue",
        "threshold": 1,
        "authorizers": [{
                "ref": "[A]
EVT6MRyAjQq8ud7hVNYcfnVPJqcVpscN5So8BhtHuGYqET5GDW5CV",
                "weight": 1
            }
        ]
    },
    "transfer": {
        "name": "transfer",
        "threshold": 1,
        "authorizers": [{
                "ref": "[G] OWNER",
                "weight": 1
            }
        ]
    },
    "manage": {
        "name": "manage",
        "threshold": 1,
        "authorizers": [{
                "ref": "[A]
EVT6MRyAjQq8ud7hVNYcfnVPJqcVpscN5So8BhtHuGYqET5GDW5CV",
                "weight": 1
            }
        ]
    }
}
```

### getGroupDetail(name)

Get detail information about a group by its `name`.

> Make sure you have history_plugin enabled on connected node

**Parameters**

- `name`: The name of the group you want to query

**Response**

The response is the detail information of the group you queried.

A example:

```
{
    "name": "testgroup",
    "key": "EVT5RsxormWcjvVBvEdQFonu5RNG4js8Zvz9pTjABLZaYxo6NNbSJ",
    "root": {
        "threshold": 6,
        "weight": 0,
        "nodes": [{
                "threshold": 1,
                "weight": 3,
                "nodes": [{
                        "key":
"EVT6MRyAjQq8ud7hVNYcfnVPJqcVpscN5So8BhtHuGYqET5GDW5CV",
                        "weight": 1
                    }, {
                        "key":
"EVT8MGU4aKiVzqMtWi9zLpu8KuTHZWjQQrX475ycSxEkLd6aBpraX",
                        "weight": 1
                    }
                ]
            }, {
                "key": "EVT8MGU4aKiVzqMtWi9zLpu8KuTHZWjQQrX475ycSxEkLd6aBpraX",
                "weight": 3
            }, {
                "threshold": 1,
                "weight": 3,
                "nodes": [{
                        "key":
"EVT6MRyAjQq8ud7hVNYcfnVPJqcVpscN5So8BhtHuGYqET5GDW5CV",
                        "weight": 1
                    }, {
                        "key":
"EVT8MGU4aKiVzqMtWi9zLpu8KuTHZWjQQrX475ycSxEkLd6aBpraX",
                        "weight": 1
                    }
                ]
            }
        ]
    }
}
```

### getTransactionsDetailOfPublicKeys(publickeys, [skip], [take = 10])

Get detail information about transactions about provided `public keys`.

> Make sure you have history_plugin enabled on connected node

**Parameters**

- `publickeys`: The public keys you want to query about, can be an array or a single value (Required)

- `skip`: The count to skip before taking data from the list, for paging. Optional.
- `take`: The count to take, for paging. Optional.

**Response**

The response is an array consisting of the detail information of the transactions you queried.

A example:

```
[{
    "id": "0925740e3be034e4ac345461d6f5b95162a7cf1578a7ec3c7b9de0e9f0f84e3c",
    "signatures": [

"SIG_K1_K1G7PJcRaTgw8RBDVvHsj2SEPZTcV5S8KgdrSmpD1oUd6fgVdwD3jSqL7zSkaFAV2zDPsr4pYT
K1QkusALsEDGXk4PUC8y"
    ],
    "compression": "none",
    "packed_trx":
"9073345baf0105f3a965000100802bebd152e74c0000000000000000000000009f077d00000000000
00000000000000819e4701640000000000000000000000000000009f077d030000000000000000000000
00000307c000000000000000000000000000407c00000000000000000000000000000507c010003c7e
3ff0060d848bd31bf53daf1d5fed7d82c9b1121394ee15dcafb07e913a97000",
    "transaction": {
      "expiration": "2018-06-28T05:35:12",
      "ref_block_num": 431,
      "ref_block_prefix": 1705636613,
      "actions": [{
          "name": "issuetoken",
          "domain": "test",
          "key": ".issue",
          "data": {
            "domain": "test",
            "names": [
              "t1",
              "t2",
              "t3"
            ],
            "owner": [
              "EVT8MGU4aKiVzqMtWi9zLpu8KuTHZWjQQrX475ycSxEkLd6aBpraX"
            ]
          },
          "hex_data":
"0000000000000000000000000009f077d03000000000000000000000307c000000000000000000000
0000000000000407c0000000000000000000000000000507c010003c7e3ff0060d848bd31bf53daf1d
5fed7d82c9b1121394ee15dcafb07e913a970"
        }
      ],
      "transaction_extensions": []
    }
}]
```

### getFungibleSymbolDetail(name)

Get detail information about a fungible token symbol which has provided `name`.

> Make sure you have history_plugin enabled on connected node

**Parameters**

- `name`: The symbol name to query about, only the name, precision should not be included (required)

**Response**

The response is a abject containing the detail information of the symbols you queried.

A example:

```
{
  "sym": "5,EVT",
  "creator": "EVT8MGU4aKiVzqMtWi9zLpu8KuTHZWjQQrX475ycSxEkLd6aBpraX",
  "create_time": "2018-06-28T05:31:09",
  "issue": {
    "name": "issue",
    "threshold": 1,
    "authorizers": [{
        "ref": "[A] EVT8MGU4aKiVzqMtWi9zLpu8KuTHZWjQQrX475ycSxEkLd6aBpraX",
        "weight": 1
      }
    ]
  },
  "manage": {
    "name": "manage",
    "threshold": 1,
    "authorizers": [{
        "ref": "[A] EVT8MGU4aKiVzqMtWi9zLpu8KuTHZWjQQrX475ycSxEkLd6aBpraX",
        "weight": 1
      }
    ]
  },
  "total_supply": "100000.00000 EVT",
  "current_supply": "0.00000 EVT",
  "metas": []
}
```

### getRequiredKeysForSuspendedTransaction(proposalName, availableKeys)

This API is used for getting required keys for suspend transaction. Other than non-suspended transaction, this API will not throw exception when your keys don't satisfies the permission requirements for one action, instead returns the proper keys needed for authorizing the suspended transaction.

**Parameters**

- `proposalName`: The proposal name you want to sign.
- `availableKeys`: Array of public keys you own.

**Response**

The response is an array representing the list of required keys.

A example:

```
[
    "EVT6MRyAjQq8ud7hVNYcfnVPJqcVpscN5So8BhtHuGYqET5GDW5CV"
]
```

### getSuspendedTransactionDetail(proposalName)

Get detail information for specific `proposal`.

**Parameters**

- `proposalName`: The proposal name you want to query.

**Response**

The response is an array representing the detail information of a suspended transaction, including the keys which has signed on the transaction and signed signatures.

A example:

```
{
  "name": "suspend3",
  "proposer": "EVT546WaW3zFAxEEEkYKjDiMvg3CHRjmWX2XdNxEhi69RpdKuQRSK",
  "status": "proposed",
  "trx": {
    "expiration": "2018-07-03T07:34:14",
    "ref_block_num": 23618,
    "ref_block_prefix": 1259088709,
    "actions": [{
        "name": "newdomain",
        "domain": "test4",
        "key": ".create",
        "data": {
          "name": "test4",
          "creator": "EVT6MRyAjQq8ud7hVNYcfnVPJqcVpscN5So8BhtHuGYqET5GDW5CV",
          "issue": {
            "name": "issue",
            "threshold": 1,
            "authorizers": [{
                "ref": "[A]
EVT6MRyAjQq8ud7hVNYcfnVPJqcVpscN5So8BhtHuGYqET5GDW5CV",
                "weight": 1
              }
            ]
          },
          "transfer": {
            "name": "transfer",
            "threshold": 1,
            "authorizers": [{
                "ref": "[G] OWNER",
                "weight": 1
              }
            ]
          },
          "manage": {
            "name": "manage",
            "threshold": 1,
            "authorizers": [{
                "ref": "[A]
EVT6MRyAjQq8ud7hVNYcfnVPJqcVpscN5So8BhtHuGYqET5GDW5CV",
                "weight": 1
              }
            ]
          }
        },
        "hex_data":
"000000000000000000000000189f077d0002c0ded2bc1f1305fb0faac5e6c03ee3a1924234985427b
6167ca569d13df435cf000000008052e74c01000000010100000002c0ded2bc1f1305fb0faac5e6c03
ee3a1924234985427b6167ca569d13df435cf000000000000000100000000b298e982a401000000010
200000000000000000000000000000000000000000000000000000000000000000000000000000000
0000001000000000094135c6801000000010100000002c0ded2bc1f1305fb0faac5e6c03ee3a192423
4985427b6167ca569d13df435cf000000000000000100"
      }
    ],
    "transaction_extensions": []
  },
  "signed_keys": [
    "EVT6MRyAjQq8ud7hVNYcfnVPJqcVpscN5So8BhtHuGYqET5GDW5CV"
  ],
  "signatures": [

"SIG_K1_K1x3vANVU1H9zxKutyRUB4kHKqMLBCaohqPwEsit9oNL8j5SUgMxxgDFA7hwCz9DkrrpaLJSnd
qcxy3Rmy5qfQw21qHpiJ"
  ]
}
```

### pushTransaction(...actions)

Push a `transaction` to the chain. A `transaction` is composed of some `actions`. Generally a `action` is a interface to a writable API. Almost all the writable API are wrapped in transactions.

`...` is the syntax for `Rest Parameters` in JavaScript's `ES6`. It means you could pass as many parameters as you want to the function, and JavaScript will automatically convert them into an array. So you may use `pushTransaction` like this:

```
apiCaller.pushTransaction(
    new EVT.EvtAction(....), // the first action
    new EVT.EvtAction(....), // the second action
    new EVT.EvtAction(....), // the third action
    new EVT.EvtAction(....), // other actions
    ....                 // as more as you want
);
```

**Parameters**

Each `action` is either a `EvtAction` instance or a `abi` structure. `EvtAction` is preferred.

A EvtAction can be created like this:

```
new EVT.EvtAction(actionName, abiStructure, [domain], [key])
```

- `actionName`: Required, the name of the action you want to execute.
- `abiStructure`: Required, the ABI structure of this action.
- `domain` & `key`: See below.

You can find all the actions and ABI structure in everiToken here and here;

For each action you should provide `domain` and `key` which are two special values. Each action has its own requirement for these two values. You can see here for detail: https://github.com/everitoken/evt/blob/master/docs/API-References.md#post-v1chaintrx_json_to_digest

For the following actions, you may ignore the `domain` and `key` parameter of the constructor of `EvtAction` (will be filled in automatically):

- `newdomain`
- `updatedomain`
- `newgroup`
- `updategroup`
- `issuetoken`
- `transfer`
- `destroytoken`
- `newsuspend`
- `aprvsuspend`
- `cancelsuspend`
- `execsuspend`

Here is a example to use `pushTransaction`.

```
// this is a example about how to create a fungible token
let symbol = "ABC";

// pass EvtAction instance as a action
await apiCaller.pushTransaction(
    new EVT.EvtAction("newfungible", {
        sym: "5," + symbol,
        creator: publicKey,
        issue: { name: "issue", threshold: 1, authorizers: [ { ref: "[A] " +
publicKey, weight: 1  } ] },
        manage: { name: "manage", threshold: 1, authorizers: [ { ref: "[A] " +
publicKey, weight: 1  } ] },
        total_supply: "100000.00000 " + symbol
    })
);
```

**Response**

The response is a object containing a value named `transactionId` if succeed. Or a `Error` will be thrown.

## Action Examples

### Create Domain

```
await apiCaller.pushTransaction(
    new EVT.EvtAction("newdomain", {
        "name": testingTmpData.newDomainName,
        "creator": publicKey,
        "issue": {
            "name": "issue",
            "threshold": 1,
            "authorizers": [{
                "ref": "[A] " + publicKey,
                "weight": 1
            }]
        },
        "transfer": {
            "name": "transfer",
            "threshold": 1,
            "authorizers": [{
                "ref": "[G] OWNER",
                "weight": 1
            }]
        },
        "manage": {
            "name": "manage",
            "threshold": 1,
            "authorizers": [{
                "ref": "[A] " + publicKey,
                "weight": 1
            }]
        }
    })
);
```

### Issue Non-Fungible Tokens

```
await apiCaller.pushTransaction(
    new EVT.EvtAction("issuetoken", {
        "domain": testingTmpData.newDomainName,
        "names": [
            testingTmpData.addedTokenNamePrefix + "1",
            testingTmpData.addedTokenNamePrefix + "2",
            testingTmpData.addedTokenNamePrefix + "3"
        ],
        "owner": [
            Key.privateToPublic(wif)
        ]
    })
);
```

### Create Fungible Symbol

```
let newSymbol = randomString();
let publicKey = xxxxxxxxxxxxxxxxxx; // replace with your public key

let newTrxId = (await apiCaller.pushTransaction(
    new EVT.EvtAction("newfungible", {
        sym: "5," + newSymbol,
        creator: publicKey,
        issue: { name: "issue", threshold: 1, authorizers: [ { ref: "[A] " +
publicKey, weight: 1  } ] },
        manage: { name: "manage", threshold: 1, authorizers: [ { ref: "[A] " +
publicKey, weight: 1  } ] },
        total_supply: "100000.00000 " + newSymbol
    })
)).transactionId;
```

### Create Group

```
await apiCaller.pushTransaction(
    new EVT.EvtAction("newgroup", {
        "name": testingTmpData.newGroupName,
        "group": {
            "name": testingTmpData.newGroupName,
            "key": Key.privateToPublic(wif),
            "root": {
                "threshold": 6,
                "weight": 0,
                "nodes": [
                    {
                        "threshold": 1,
                        "weight": 3,
                        "nodes": [
                            {
                                "key":
"EVT6MRyAjQq8ud7hVNYcfnVPJqcVpscN5So8BhtHuGYqET5GDW5CV",
                                "weight": 1
                            },
                            {
                                "key":
"EVT8MGU4aKiVzqMtWi9zLpu8KuTHZWjQQrX475ycSxEkLd6aBpraX",
                                "weight": 1
                            }
                        ]
                    },
                    {
                        "key":
"EVT8MGU4aKiVzqMtWi9zLpu8KuTHZWjQQrX475ycSxEkLd6aBpraX",
                        "weight": 3
                    },
                    {
                        "threshold": 1,
                        "weight": 3,
                        "nodes": [
                            {
                                "key":
"EVT6MRyAjQq8ud7hVNYcfnVPJqcVpscN5So8BhtHuGYqET5GDW5CV",
                                "weight": 1
                            },
                            {
                                "key":
"EVT8MGU4aKiVzqMtWi9zLpu8KuTHZWjQQrX475ycSxEkLd6aBpraX",
                                "weight": 1
                            }
                        ]
                    }
                ]
            }
        }
    })
);
```