DAO DEPOSIT

OVERVIEW

DAO DEPOSIT is an Algorand-powered protocol to implement decentralized fundraising via an Algorand TEAL (Transaction Execution Approval Language) “smart contract” and “inner transactions.” The protocol stores and disperses funds via an “application address”, which is an Algorand account that is controlled entirely by the smart contract.

SDK NOTE:

Pipeline is an Algorand connector sdk developed by HEADLINE INC. The Pipeline code examples provided both assemble and send transactions, as well as prompt signing of transactions via MyAlgo wallet, WalletConnect, or AlgoSigner.

FLOW

Deploy

The individual or organization that is raising funds modifies the provided TEAL contract to hard-code a receiver address and fundraising goal in microAlgos.

```
let teal = modifyTeal(recipient, amount)

Pipeline.deployTeal(teal, clearTeal, [1, 1, 2, 6],
["create"]).then(data => {
     console.log(data);
 })
```

After deploying the contract, the recipient and investors opt-in to the contract:

```
Pipeline.optIn(appId, ["register"]).then(data => {
     console.log(data);
})
```

Fund

Investors then contribute funds to the application address, which is accessed by calling an algosdk function:

```
const appAddress = algosdk.getApplicationAddress(parseInt(appId))

Pipeline.appCallWithTxn(appId, ["fund"], appAddress, famt, "funding",
0, [appAddress]).then(data => {
      console.log(data)
})
```

In order for the contribution to be counted, the payment transaction must be the second in a group, in which the first transaction is an app call with the argument `"fund"`. As investors contribute to the app, their individual cumulative contributions are tabulated and stored in their "local app states" under the key `"amt"`. Contribution attempts will fail if the amount contributed pushes the balance of the app address above the funding goal.

Withdraw

When the funding goal has been reached, the recipient enters an app call with the argument `"withdraw".` The application account then sends the balance (minus fees) to the specified recipient address and set the global `"ready"` value to 1 (true).

```
Pipeline.appCall(appId, ["withdraw"], [appAddress,
feeAddress]).then(data => {
      console.log(data)
})
```

Deposit

After withdrawing the funds, the fundraiser may then deposit funds to the application account for dispersal to investors by sending a group transaction with an app call with the argument `"deposit"` along with a payment transaction to the application account. The app will then increment its global state value with the key `"depositAmount."`

```
Pipeline.appCallWithTxn(appId, ["deposit"], appAddress,
depositAmt, "depositing", 0, [appAddress]).then(data => {
      console.log(data)
})
```

Redeem

Investors may attempt to redeem their dividend by sending an app call with the argument "redeem."

```
Pipeline.appCall(appId, ["redeem"], [appAddress]).then(data => {
      console.log(data)
})
```

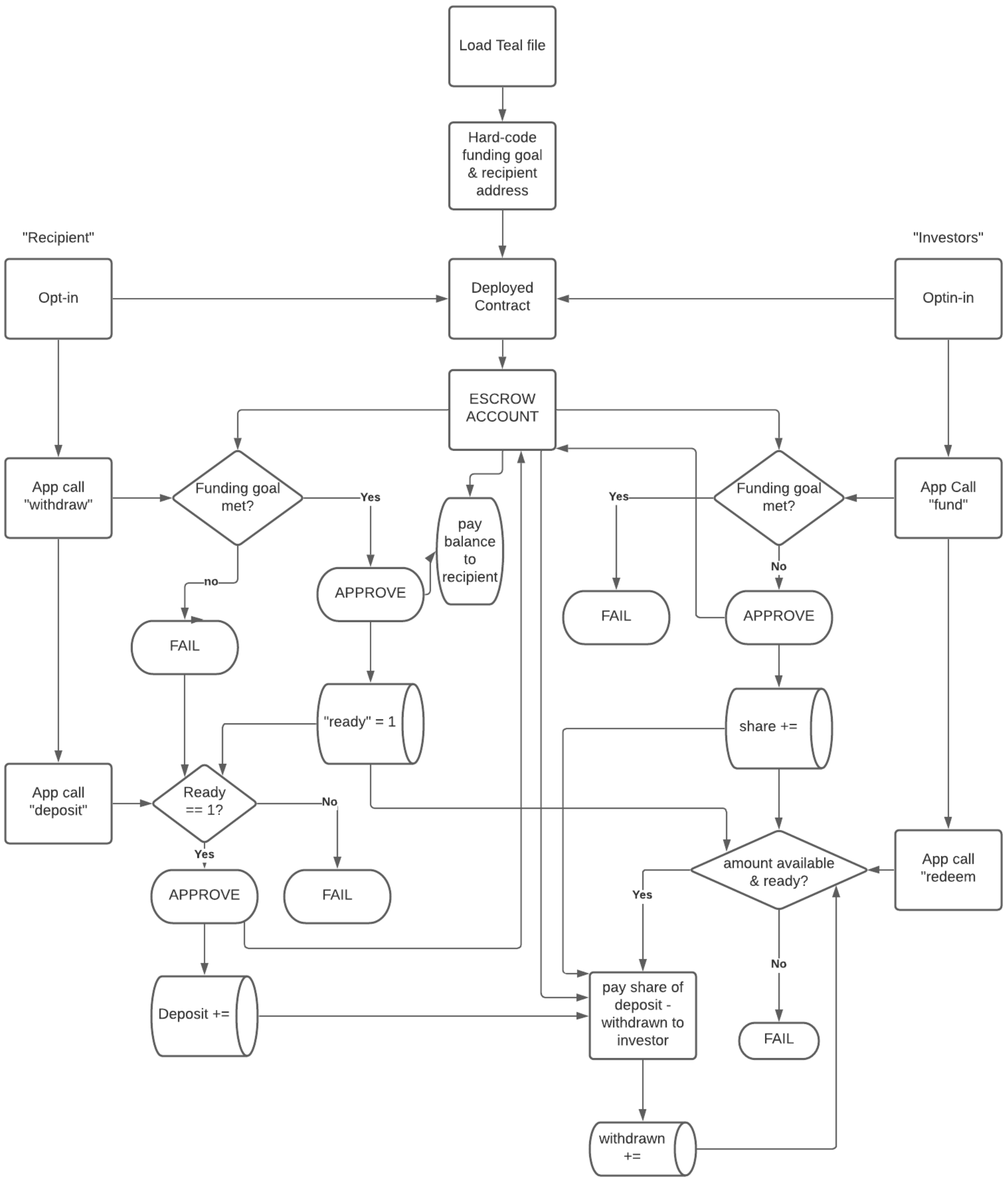
The amount available to redeem is calculating by factoring in the share percentage, amount of total dividend ever deposited, and amount of total dividend ever redeemed. The TEAL function for this calculation is provided below, which also subtracts the minimum balance and fee:

```
//calculate amount available to withdraw

int 0
byte "amt"
app_local_get
byte "depositAmount"
app_global_get
*
int  3000000
/
int 0
byte "withdrawn"
app_local_get
-
int 1
min_balance
-
int 2000
-
store 0
// check to make sure that amount available is > 10000 microalgos
load 0
int 10000
>=
bz failed
```

Formula:

$$amountAvailable = (\ totalDeposit \times share\ ) - withdrawn - (\ fee + minimumBalance\ )$$


Load Teal file
Hard-code funding goal & recipient address
"Recipient"
"Investors"
Opt-in
Deployed Contract
Optin-in
ESCROW ACCOUNT
App call "withdraw"
Funding goal met?
Yes
no
pay balance to recipient
APPROVE
FAIL
"ready" = 1
App call "deposit"
Ready == 1?
No
Yes
APPROVE
FAIL
Deposit +=
Yes
Funding goal met?
App Call "fund"
No
FAIL
APPROVE
share +=
amount available & ready?
App call "redeem
Yes
No
pay share of deposit - withdrawn to investor
FAIL
withdrawn +=