

Executor DAO

The one DAO to rule them all. ExecutorDAO is designed to be completely modular and flexible, leveraging Clarity to the fullest extent. The core tenets of ExecutorDAO that make this possible are:

- 1. Proposals are smart contracts.
- 2. The core executes, the extensions give form.

ሄ Clarity-Innovation-Lab / executor-dao Public

3. Ownership control happens via sending context.

This is an early release. More information, design specifics, and unit tests coming soon.

1. Proposals are smart contracts

The way a conventional company operates is defined in its constitution. Changes are made by means of resolutions put forward by its members, the process of which is described by that same constitution. Translating this into a DAO was one of the aims when designing ExecutorDAO. Proposals are therefore expressed as smart contracts. Clarity is a beautifully expressive language. Instead of verbose "Legalese", we can describe the operation, duties, and members using concise logical statements. Proposals implement a specific trait and may be executed by the DAO when certain conditions are met. It makes ExecutorDAO extremely flexible and powerful.

2. The core executes, the extensions give form

ExecutorDAO initially consists of just one core contract. Its sole purpose is to execute proposals and to keep a list of authorised extensions. There are no other features: no token, no voting, no functions. The DAO is given form by means of so-called extension contracts. Extensions are contracts that can be enabled or disabled by proposals and add specific features to the DAO. They are allowed to assume the "sending context" of the DAO and can thus enact change. Since different groups and organisations have different needs, extensions are rather varied. Some example functionality that can be added to ExecutorDAO via an extension include:

- The issuance and management of a governance token.
- The ability to submit proposals.

- The ability to vote on proposals.
- The creation and management of a treasury.
- Salary payouts to specific members.
- · And more...

Since extensions become part of the DAO, they have privileged access to everything else included in the DAO. The trick that allows for extension interoperability is a common authorisation check. Privileged access is granted when the sending context is equal to that of the DAO or if the contract caller is an enabled DAO extension. It allows for extensions that depend on other extensions to be designed. They can be disabled and replaced at any time making ExecutorDAO fully polymorphic.

3. Ownership control happens via sending context

ExecutorDAO follows a single-address ownership model. The core contract is the de facto owner of external ownable contracts. (An ownable contract is to be understood as a contract that stores one privileged principal that may change internal state.) External contracts thus do not need to implement a complicated access model, as any proposal or extension may act upon it. Any ownable contract, even the ones that were deployed before ExecutorDAO came into use, can be owned and managed by the DAO. Proposals are executed in the sending context of the DAO and extensions can request it via a callback procedure.

Reference extensions

The ExecutorDAO code base comes with a few reference extension contracts. These are designated by a code that starts with "EDE" followed by an incrementing number of three digits.

EDE000: Governance Token

Implements a SIP010 governance token with locking capabilities. The DAO has full control over the minting, burning, transferring, and locking.

EDE001: Proposal Voting

Allows governance token holders to vote on proposals. (Note: *vote*, not *propose*.) One token equals one vote. Tokens used for voting are locked for the duration of the vote. They can then be reclaimed and used again.

EDE002: Proposal Submission

Allows governance token holders that own at least 1% of the supply to submit a proposal to be voted on via EDE001. Proposals that are made this way are subject to a delay of at least 144 blocks (~1 day) to 1004 blocks (~7 days) and run for 1440 blocks (~10 days). All these parameters can be changed by a proposal.

EDE003: Emergency Proposals

Manages a list of emergency team members that have the ability to submit emergency proposals to EDE001. Such proposals are not subject to a start delay and run for only 144 blocks (~1 day). This extension is subject to a sunset period after which it deactivates. The members, parameters, and sunset period can be changed by a proposal.

EDE004: Emergency Execute

Manages a list of executive team members that have the ability to signal for the immediate execution of a proposal. This extension is subject to a sunset period after which it deactivates. The members, parameters, and sunset period can be changed by a proposal.

EDE005: Dev Fund

An extension that functions as a development fund. It can hold a governance token balance and manages monthly developer payouts. The developers that receive a payout as well as the amounts can be changed by a proposal.

Reference proposals

ExecutorDAO also comes with some reference and example proposals. These are designated by a code that starts with "EDP" followed by an incrementing number of three digits. The numbers do not to coincide with extension numbering.

EDP000: Bootstrap

A bootstrapping proposal that is meant to be executed when the ExecutorDAO is first deployed. It initialises boot extensions, sets various parameters on them, and mints initial governance tokens.

EDP001: Dev Fund

Enables the EDE005 Dev Fund extension, mints an amount of tokens for it equal to 30% of the current supply, and awards an allowance to two principals.

EDO002: Kill Emergency Execute

Immediately disables EDE004 Emergency Execute if it passes.

EDE003: Allowlist Escrow NFT

A simple example on how ExecutorDAO can add third-party smart contracts to be allowlisted.

Testing

Unit tests coming soon, for now you can try it out manually in a clarinet console session. Execute the bootstrap proposal and have at it:

```
(contract-call? .executor-dao construct .edp000-bootstrap) □
```

To propose edp001-dev-fund, run the following commands one by one:

```
ſŌ
;; Submit the proposal via extension EDE002, starting
;; the voting process at block-height + 144.
(contract-call? .ede002-proposal-submission propose .edp001-dev-fund (+ block-height u144) .ede000-governance-token)
;; Advance the chain 144 blocks.
::advance_chain_tip 144
;; Vote YES with 100 tokens.
(contract-call? .ede001-proposal-voting vote u100 true .edp001-dev-fund .ede000-governance-token)
;; (Optional) take a look at the current proposal data.
(contract-call? .ede001-proposal-voting get-proposal-data .edp001-dev-fund)
;; Advance the chain tip until after the proposal concludes.
::advance_chain_tip 1440
;; Conclude the proposal vote, thus executing it.
(contract-call? .ede001-proposal-voting conclude .edp001-dev-fund)
;; Check that the ede005-dev-fund contract now
;; indeed holds governance tokens.
::get_assets_maps
;; Reclaim tokens used for voting.
(contract-call? .ede001-proposal-voting reclaim-votes .edp001-dev-fund .ede000-governance-token)
```

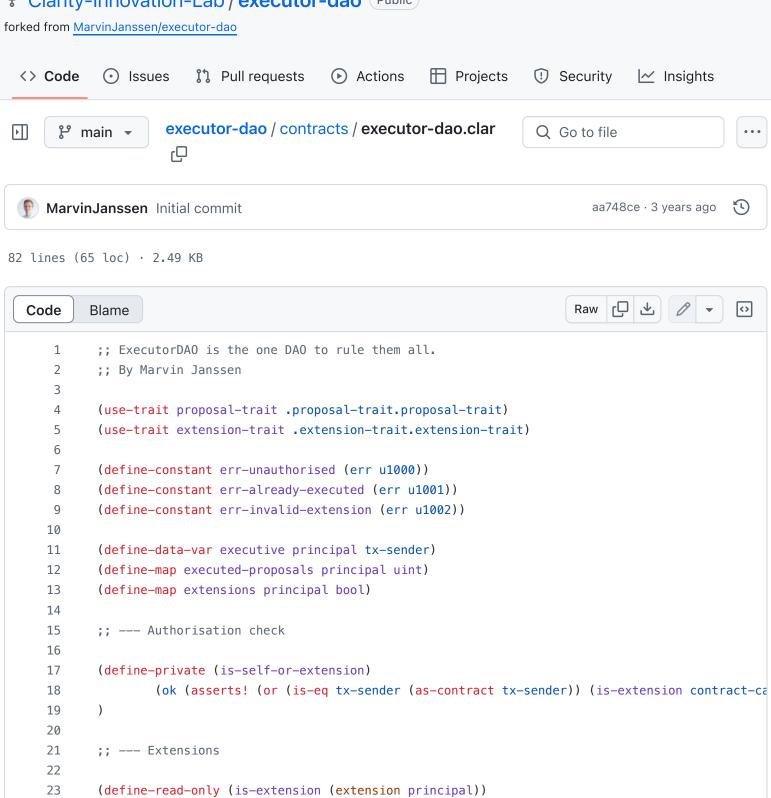
Error space

ExecutorDAO reserves different uint ranges for the main components.

- 1000–1999 : ExecutorDAO errors.
- 2000–2999 : Proposal errors.
- 3000-3999 : Extension errors.

License

MIT license, all good as long as the copyright and permission notice are included. Although I ask developers that adopt ExecutorDAO in one way or another to make the adapter open source. (The client code that interfaces with the DAO.)



(default-to false (map-get? extensions extension))

(define-public (set-extension (extension principal) (enabled bool))

(ok (map-set extensions extension enabled))

(define-private (set-extensions-iter (item {extension: principal, enabled: bool}))

(man_set extensions (get extension item) (get enabled item))

(print {event: "extension", extension: extension, enabled: enabled})

(print {event: "extension", extension: (get extension item), enabled: (get er

(try! (is-self-or-extension))

242526

2728

29

30

31

32

33 34

35 36

37

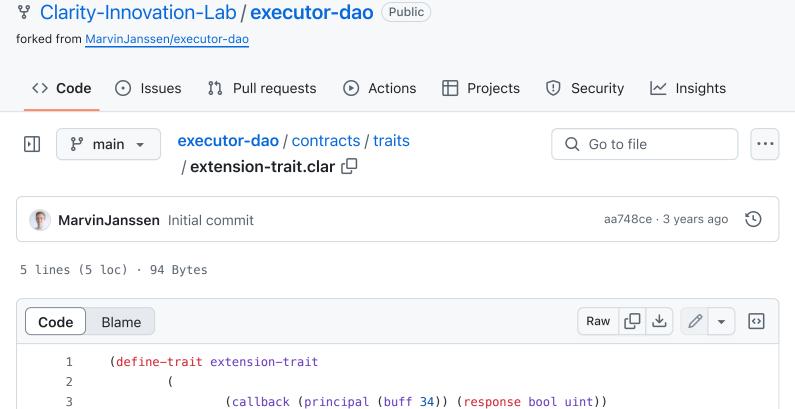
30

)

(begin

(begin

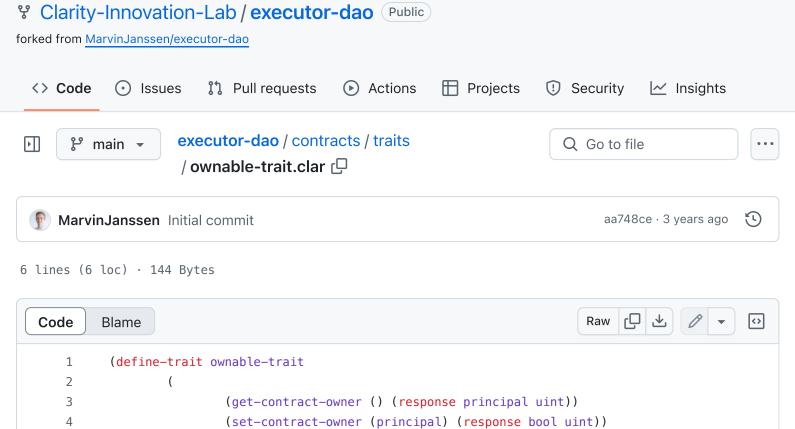
```
)
39
      )
40
41
      (define-public (set-extensions (extension-list (list 200 {extension: principal, enabled: bool
42
43
             (begin
                    (try! (is-self-or-extension))
44
                    (ok (map set-extensions-iter extension-list))
45
             )
46
      )
47
48
      ;; --- Proposals
49
50
51
      (map-get? executed-proposals (contract-of proposal))
52
53
      )
54
      55
56
             (begin
57
                    (try! (is-self-or-extension))
                    (asserts! (map-insert executed-proposals (contract-of proposal) block-height)
58
                    (print {event: "execute", proposal: proposal})
59
                    (as-contract (contract-call? proposal execute sender))
60
61
             )
      )
62
63
64
      ;; --- Bootstrap
65
66
      67
             (let ((sender tx-sender))
                    (asserts! (is-eq sender (var-get executive)) err-unauthorised)
68
69
                    (var-set executive (as-contract tx-sender))
                    (as-contract (execute proposal sender))
70
71
             )
72
      )
73
74
      ;; --- Extension requests
75
      (define-public (request-extension-callback (extension <extension-trait>) (memo (buff 34)))
76
77
             (let ((sender tx-sender))
                    (asserts! (is-extension contract-caller) err-invalid-extension)
78
                    (asserts! (is-eq contract-caller (contract-of extension)) err-invalid-extensi
79
                    (as-contract (contract-call? extension callback sender memo))
80
81
             )
      )
82
```



প Clarity-Innovation-Lab / executor-dao Public forked from MarvinJanssen/executor-dao Projects <> Code Issues ?? Pull requests Actions Security ✓ Insights executor-dao / contracts / traits F ያ main ▼ Q Go to file / governance-token-trait.clar 📮 aa748ce · 3 years ago MarvinJanssen Initial commit

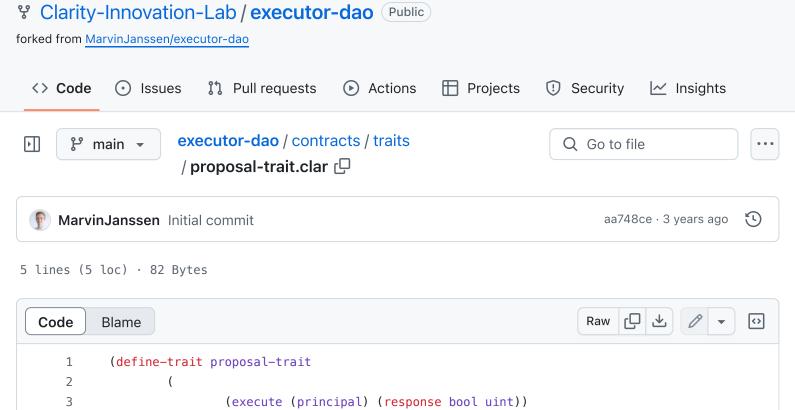
12 lines (12 loc) · 490 Bytes

```
Raw 🕒 😃
                                                                                                    (>)
Code
        Blame
   1
          (define-trait governance-token-trait
   2
   3
                          (edg-get-balance (principal) (response uint uint))
                          (edg-has-percentage-balance (principal uint) (response bool uint))
   4
   5
                          (edg-transfer (uint principal principal) (response bool uint))
                          (edg-lock (uint principal) (response bool uint))
   6
   7
                          (edg-unlock (uint principal) (response bool uint))
   8
                          (edg-get-locked (principal) (response uint uint))
   9
                          (edg-mint (uint principal) (response bool uint))
                          (edg-burn (uint principal) (response bool uint))
  10
                  )
  11
  12
         )
```

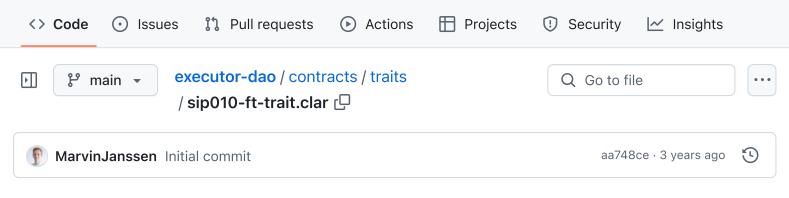


6

)



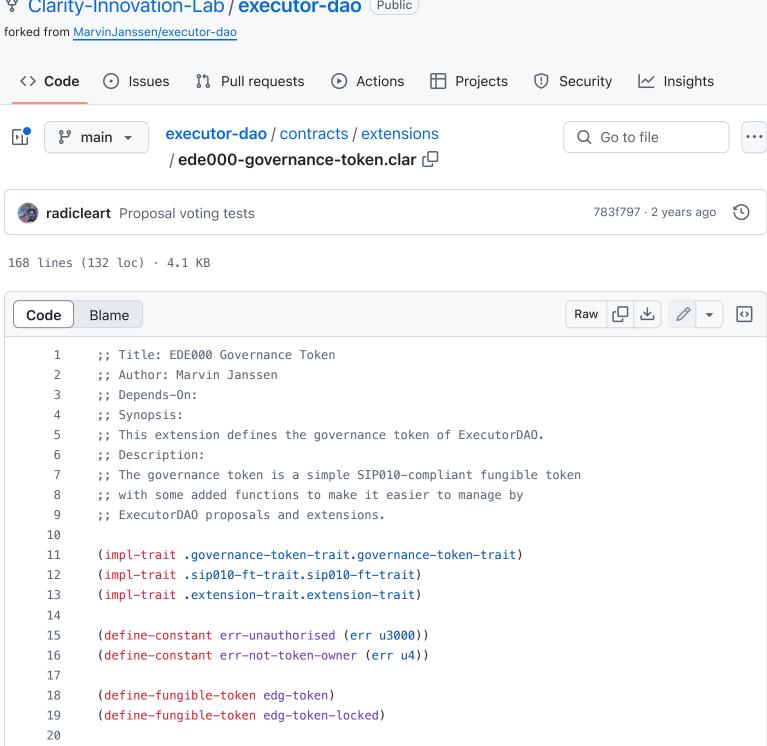
forked from MarvinJanssen/executor-dao



24 lines (18 loc) · 812 Bytes

```
Raw 🕒 😃
Code
        Blame
   1
          (define-trait sip010-ft-trait
   2
                          ;; Transfer from the caller to a new principal
   3
                          (transfer (uint principal principal (optional (buff 34))) (response bool uint
   4
   5
                          ;; the human readable name of the token
   6
   7
                          (get-name () (response (string-ascii 32) uint))
   8
                          ;; the ticker symbol, or empty if none
   9
                          (get-symbol () (response (string-ascii 32) uint))
  10
  11
                          ;; the number of decimals used, e.g. 6 would mean 1_000_000 represents 1 toke
  12
                          (get-decimals () (response uint uint))
  13
  14
                          ;; the balance of the passed principal
  15
  16
                          (get-balance (principal) (response uint uint))
  17
                          ;; the current total supply (which does not need to be a constant)
  18
  19
                          (get-total-supply () (response uint uint))
  20
                          ;; an optional URI that represents metadata of this token
  21
  22
                          (get-token-uri () (response (optional (string-utf8 256)) uint))
                  )
  23
  24
         )
```

♥ Clarity-Innovation-Lab / executor-dao Public



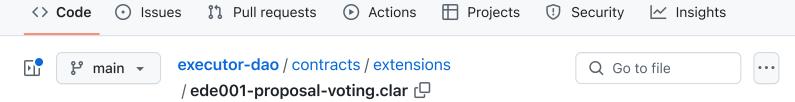
```
21
       (define-data-var token-name (string-ascii 32) "ExecutorDAO Governance Token")
       (define-data-var token-symbol (string-ascii 10) "EDG")
22
       (define-data-var token-uri (optional (string-utf8 256)) none)
23
       (define-data-var token-decimals uint u6)
24
25
26
       ;; --- Authorisation check
27
       (define-public (is-dao-or-extension)
28
               (ok (asserts! (or (is-eq tx-sender .executor-dao) (contract-call? .executor-dao is-ex
29
       )
30
31
       ;; --- Internal DAO functions
32
33
34
       ;; governance-token-trait
35
       (define-public (edg-transfer (amount uint) (sender principal) (recipient principal))
36
37
               (begin
30
                       (tryl (ic-dan-or-extension))
```

```
39
                        (ft-transfer? edg-token amount sender recipient)
40
               )
       )
41
42
43
       (define-public (edg-lock (amount uint) (owner principal))
                (begin
44
45
                        (try! (is-dao-or-extension))
                        (try! (ft-burn? edg-token amount owner))
46
                        (ft-mint? edg-token-locked amount owner)
47
               )
48
49
       )
50
       (define-public (edg-unlock (amount uint) (owner principal))
51
                (begin
52
53
                        (try! (is-dao-or-extension))
                        (try! (ft-burn? edg-token-locked amount owner))
54
                        (ft-mint? edg-token amount owner)
55
               )
56
       )
57
58
59
       (define-public (edg-mint (amount uint) (recipient principal))
60
                        (try! (is-dao-or-extension))
61
                        (ft-mint? edg-token amount recipient)
62
63
               )
       )
64
65
66
       (define-public (edg-burn (amount uint) (owner principal))
67
                (begin
                        (try! (is-dao-or-extension))
68
69
                        (ft-burn? edg-token amount owner)
70
71
               )
72
       )
73
74
       ;; Other
75
76
       (define-public (set-name (new-name (string-ascii 32)))
77
                (begin
                        (try! (is-dao-or-extension))
78
79
                        (ok (var-set token-name new-name))
               )
80
       )
81
82
       (define-public (set-symbol (new-symbol (string-ascii 10)))
83
84
                (begin
                        (try! (is-dao-or-extension))
85
86
                        (ok (var-set token-symbol new-symbol))
87
               )
       )
88
89
       (define-public (set-decimals (new-decimals uint))
90
                (begin
91
92
                        (try! (is-dao-or-extension))
93
                        (ok (var-set token-decimals new-decimals))
94
```

```
95
        )
 96
 97
        (define-public (set-token-uri (new-uri (optional (string-utf8 256))))
 98
                (begin
 99
                        (try! (is-dao-or-extension))
100
                        (ok (var-set token-uri new-uri))
                )
101
        )
102
103
104
        (define-private (edg-mint-many-iter (item {amount: uint, recipient: principal}))
105
                (ft-mint? edg-token (get amount item) (get recipient item))
        )
106
107
        (define-public (edg-mint-many (recipients (list 200 {amount: uint, recipient: principal})))
108
109
110
                         (try! (is-dao-or-extension))
                         (ok (map edg-mint-many-iter recipients))
111
112
                )
113
        )
114
115
        ;; --- Public functions
116
117
        ;; sip010-ft-trait
118
119
        (define-public (transfer (amount uint) (sender principal) (recipient principal) (memo (option
                (begin
120
121
                         (asserts! (or (is-eq tx-sender sender) (is-eq contract-caller sender)) err-nd
122
                         (ft-transfer? edg-token amount sender recipient)
123
                )
124
        )
125
        (define-read-only (get-name)
126
                (ok (var-get token-name))
127
        )
128
129
        (define-read-only (get-symbol)
130
131
                (ok (var-get token-symbol))
        )
132
133
134
        (define-read-only (get-decimals)
                (ok (var-get token-decimals))
135
        )
136
137
        (define-read-only (get-balance (who principal))
138
139
                (ok (+ (ft-get-balance edg-token who) (ft-get-balance edg-token-locked who)))
        )
140
141
142
        (define-read-only (get-total-supply)
                (ok (+ (ft-get-supply edg-token) (ft-get-supply edg-token-locked)))
143
144
        )
145
        (define-read-only (get-token-uri)
146
147
                (ok (var-get token-uri))
148
        )
149
150
        ;; governance-token-trait
```

```
151
        (define-read-only (edg-get-balance (who principal))
152
153
                (get-balance who)
154
        )
155
        (define-read-only (edg-has-percentage-balance (who principal) (factor uint))
156
                (ok (>= (* (unwrap-panic (get-balance who)) factor) (* (unwrap-panic (get-total-supp))
157
        )
158
159
        (define-read-only (edg-get-locked (owner principal))
160
                (ok (ft-get-balance edg-token-locked owner))
161
        )
162
163
        ;; --- Extension callback
164
165
        (define-public (callback (sender principal) (memo (buff 34)))
166
                (ok true)
167
        )
168
```

forked from MarvinJanssen/executor-dao



MarvinJanssen feat: simplify ede001 and ede002, remove governance token trait 3d8291e · 2 years ago

140 lines (118 loc) · 4.93 KB

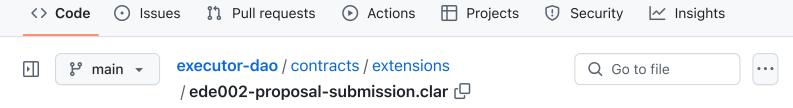
```
Raw 🕒 🕹 🧷
                                                                                                   <>
Code
        Blame
   1
         ;; Title: EDE001 Proposal Voting
   2
         ;; Author: Marvin Janssen
         ;; Depends-On: EDE000
   3
         ;; Synopsis:
   4
         ;; This extension is part of the core of ExecutorDAO. It allows governance token
   5
         ;; holders to vote on and conclude proposals.
   6
   7
         ;; Description:
         ;; Once proposals are submitted, they are open for voting after a lead up time
   8
         ;; passes. Any token holder may vote on an open proposal, where one token equals
   9
         ;; one vote. Members can vote until the voting period is over. After this period
  10
         ;; anyone may trigger a conclusion. The proposal will then be executed if the
  11
         ;; votes in favour exceed the ones against.
  12
  13
         (impl-trait .extension-trait.extension-trait)
  14
  15
         (use-trait proposal-trait.proposal-trait)
  16
  17
         (define-constant err-unauthorised (err u3000))
         (define-constant err-proposal-already-executed (err u3002))
  18
  19
         (define-constant err-proposal-already-exists (err u3003))
         (define-constant err-unknown-proposal (err u3004))
  20
         (define-constant err-proposal-already-concluded (err u3005))
  21
         (define-constant err-proposal-inactive (err u3006))
  22
         (define-constant err-proposal-not-concluded (err u3007))
  23
  24
         (define-constant err-no-votes-to-return (err u3008))
  25
         (define-constant err-end-block-height-not-reached (err u3009))
  26
         (define-constant err-disabled (err u3010))
  27
         (define-map proposals
  28
  29
                  principal
                  {
  30
  31
                          votes-for: uint,
  32
                          votes-against: uint,
                          start-block-height: uint,
  33
  34
                          end-block-height: uint,
  35
                          concluded: bool,
  36
                          passed: bool,
  37
                          proposer: principal
  30
```

```
)
39
40
      (define-map member-total-votes {proposal: principal, voter: principal} uint)
41
42
      ;; --- Authorisation check
43
44
      (define-public (is-dao-or-extension)
45
              (ok (asserts! (or (is-eq tx-sender .executor-dao) (contract-call? .executor-dao is-ex
46
      )
47
48
49
      ;; --- Internal DAO functions
50
51
      ;; Proposals
52
      53
54
55
                      (try! (is-dao-or-extension))
                      (asserts! (is-none (contract-call? .executor-dao executed-at proposal)) err-
56
57
                      (print {event: "propose", proposal: proposal, proposer: tx-sender})
58
                      (ok (asserts! (map-insert proposals (contract-of proposal) (merge {votes-for})
              )
59
60
      )
61
62
      ;; --- Public functions
63
64
      ;; Proposals
65
      (define-read-only (get-proposal-data (proposal principal))
66
              (map-get? proposals proposal)
67
      )
68
69
70
      ;; Votes
71
72
      (define-read-only (get-current-total-votes (proposal principal) (voter principal))
73
              (default-to u0 (map-get? member-total-votes {proposal: proposal, voter: voter}))
74
      )
75
76
      (define-public (vote (amount uint) (for bool) (proposal principal))
77
              (let
78
                      (
79
                              (proposal-data (unwrap! (map-get? proposals proposal) err-unknown-prd
80
                      (asserts! (>= block-height (get start-block-height proposal-data)) err-propos
81
                      (asserts! (< block-height (get end-block-height proposal-data)) err-proposal-
82
                      (map-set member-total-votes {proposal: proposal, voter: tx-sender}
83
                              (+ (get-current-total-votes proposal tx-sender) amount)
84
85
                      )
                      (map-set proposals proposal
86
87
                              (if for
                                      (merge proposal-data {votes-for: (+ (get votes-for proposal-d)
88
                                      (merge proposal-data {votes-against: (+ (get votes-against pr
89
                              )
90
91
                      (print {event: "vote", proposal: proposal, voter: tx-sender, for: for, amount
92
                      (contract-call? .ede000-governance-token edg-lock amount tx-sender)
93
94
```

```
95
       )
 96
 97
       ;; Conclusion
 98
99
       100
               (let
101
                       (
102
                              (proposal-data (unwrap! (map-get? proposals (contract-of proposal)) €
                              (passed (> (get votes-for proposal-data) (get votes-against proposal-
103
104
                       )
105
                       (asserts! (not (get concluded proposal-data)) err-proposal-already-concluded)
                       (asserts! (>= block-height (get end-block-height proposal-data)) err-end-block
106
107
                       (map-set proposals (contract-of proposal) (merge proposal-data {concluded: tr
108
                       (print {event: "conclude", proposal: proposal, passed: passed})
                       (and passed (try! (contract-call? .executor-dao execute proposal tx-sender)))
109
110
                       (ok passed)
               )
111
112
       )
113
       ;; Reclamation
114
115
116
       (let
117
118
                       (
119
                              (proposal-principal (contract-of proposal))
120
                              (proposal-data (unwrap! (map-get? proposals proposal-principal) err-d
121
                              (votes (unwrap! (map-get? member-total-votes {proposal: proposal-prir
122
                       )
123
                       (asserts! (get concluded proposal-data) err-proposal-not-concluded)
124
                       (map-delete member-total-votes {proposal: proposal-principal, voter: tx-sende
125
                       (contract-call? .ede000-governance-token edg-unlock votes tx-sender)
126
               )
       )
127
128
129
       (define-public (reclaim-and-vote (amount uint) (for bool) (proposal principal) (reclaim-from
130
               (begin
131
                       (try! (reclaim-votes reclaim-from))
                       (vote amount for proposal)
132
               )
133
134
       )
135
136
       ;; --- Extension callback
137
       (define-public (callback (sender principal) (memo (buff 34)))
138
139
               (ok true)
140
       )
```

Clarity-Innovation-Lab / executor-dao Public

forked from MarvinJanssen/executor-dao



MarvinJanssen feat: simplify ede001 and ede002, remove governance token trait 3d8291e · 2 years ago

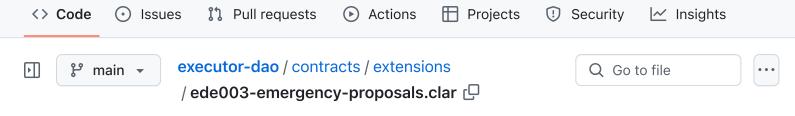
95 lines (77 loc) · 3.45 KB

```
Raw 🖵 🕹
                                                                                                   <>
Code
        Blame
   1
         ;; Title: EDE002 Proposal Submission
   2
         ;; Author: Marvin Janssen
         ;; Depends-On: EDE000, EDE001
   3
         ;; Synopsis:
   4
         ;; This extension part of the core of ExecutorDAO. It allows governance token
   5
         ;; holders to submit proposals when they hold at least n% percentage of the
   6
   7
         ;; token supply.
         ;; Description:
   8
         ;; Proposals may be submitted by anyone that holds at least n% of governance
   9
         ;; tokens. Any submission is subject to a pre-defined start delay before voting
  10
         ;; can begin, and will then run for a pre-defined duration. The percentage,
  11
         ;; start delay, and proposal duration can all by changed by means of a future
  12
  13
         ;; proposal.
  14
  15
         (impl-trait .extension-trait.extension-trait)
         (use-trait proposal-trait.proposal-trait)
  16
  17
         (define-constant err-unauthorised (err u3100))
  18
  19
         (define-constant err-not-governance-token (err u3101))
         (define-constant err-insufficient-balance (err u3102))
  20
         (define-constant err-unknown-parameter (err u3103))
  21
         (define-constant err-proposal-minimum-start-delay (err u3104))
  22
         (define-constant err-proposal-maximum-start-delay (err u3105))
  23
  24
         (define-map parameters (string-ascii 34) uint)
  25
  26
         (map-set parameters "propose-factor" u100000) ;; 1% initially required to propose (100/n*1000)
  27
         (map-set parameters "proposal-duration" u1440) ;; ~10 days based on a ~10 minute block time.
  28
         (map-set parameters "minimum-proposal-start-delay" u144) ;; ~1 day minimum delay before voting
  29
         (map-set parameters "maximum-proposal-start-delay" u1008) ;; ∼7 days maximum delay before vot
  30
  31
  32
         ;; --- Authorisation check
  33
  34
         (define-public (is-dao-or-extension)
  35
                  (ok (asserts! (or (is-eq tx-sender .executor-dao) (contract-call? .executor-dao is-ex
  36
         )
  37
  30
                Internal DAO functions
```

```
39
40
      ;; Parameters
41
42
       (define-public (set-parameter (parameter (string-ascii 34)) (value uint))
43
               (begin
44
                      (try! (is-dao-or-extension))
45
                      (try! (get-parameter parameter))
46
                      (ok (map-set parameters parameter value))
              )
47
      )
48
49
      (define-private (set-parameters-iter (item {parameter: (string-ascii 34), value: uint}) (prev
50
51
52
                      (try! previous)
                      (try! (get-parameter (get parameter item)))
53
                      (ok (map-set parameters (get parameter item) (get value item)))
54
55
               )
      )
56
57
58
       (define-public (set-parameters (parameter-list (list 200 {parameter: (string-ascii 34), value
59
               (begin
60
                      (try! (is-dao-or-extension))
61
                      (fold set-parameters-iter parameter-list (ok true))
              )
62
63
      )
64
       ;; --- Public functions
65
66
67
      ;; Parameters
68
69
       (define-read-only (get-parameter (parameter (string-ascii 34)))
70
               (ok (unwrap! (map-get? parameters parameter) err-unknown-parameter))
      )
71
72
73
      ;; Proposals
74
75
       (begin
76
77
                      (asserts! (>= start-block-height (+ block-height (try! (get-parameter "minimu
78
                      (asserts! (<= start-block-height (+ block-height (try! (get-parameter "maximu
                      (asserts! (unwrap-panic (contract-call? .ede000-governance-token edg-has-per
79
80
                      (contract-call? .ede001-proposal-voting add-proposal
81
                              proposal
                              {
82
83
                                      start-block-height: start-block-height,
                                      end-block-height: (+ start-block-height (try! (get-parameter
84
85
                                      proposer: tx-sender
                              }
86
                      )
87
              )
88
      )
89
90
91
       ;; --- Extension callback
92
      (define-public (callback (sender principal) (memo (buff 34)))
93
94
               (ok true)
```

Clarity-Innovation-Lab / executor-dao Public

forked from MarvinJanssen/executor-dao



MarvinJanssen Initial commit

aa748ce · 3 years ago

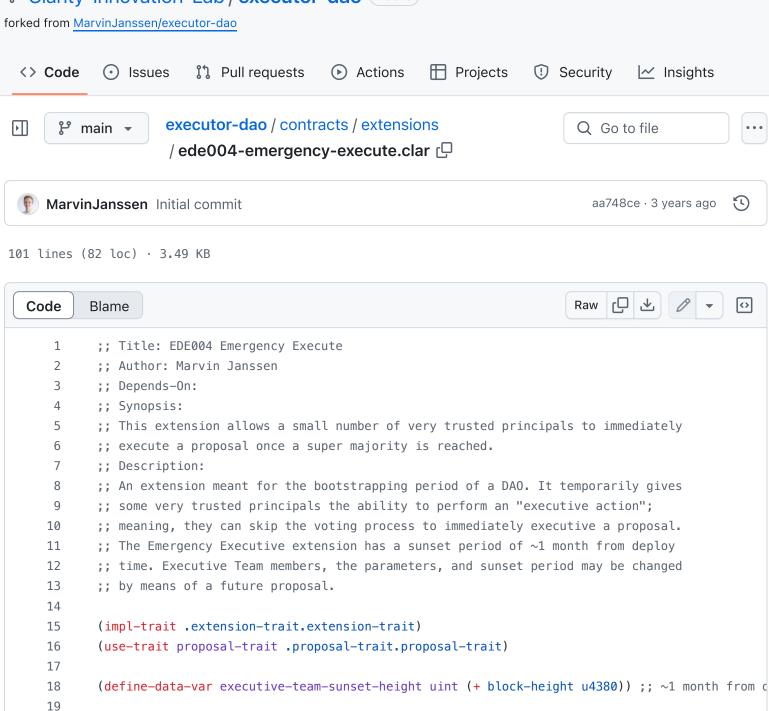
(1)

82 lines (67 loc) \cdot 2.64 KB

```
Raw 🖵 🕹
                                                                                                   <>
Code
        Blame
   1
         ;; Title: EDE003 Emergency Proposals
   2
         ;; Author: Marvin Janssen
         ;; Depends-On: EDE001
   3
   4
         ;; Synopsis:
         ;; This extension allows for the creation of emergency proposals by a few trusted
   5
         ;; principals.
   6
   7
         ;; Description:
         ;; Emergency proposals have a voting period of roughly 1 day, instead of the
   8
         ;; normal proposal duration. Only a list of trusted principals, designated as the
   9
         ;; "emergency team", can create emergency proposals. The emergency proposal
  10
         ;; extension has a ~3 month sunset period, after which no more emergency
  11
         ;; proposals can be made. The emergency team members, sunset period, and
  12
         ;; emergency vote duration can be changed by means of a future proposal.
  13
  14
  15
         (impl-trait .extension-trait.extension-trait)
         (use-trait proposal-trait.proposal-trait)
  16
  17
         (define-data-var emergency-proposal-duration uint u144) ;; ~1 day
  18
  19
         (define-data-var emergency-team-sunset-height uint (+ block-height u13140)) ;; ~3 months from
  20
  21
         (define-constant err-unauthorised (err u3000))
         (define-constant err-not-emergency-team-member (err u3001))
  22
         (define-constant err-sunset-height-reached (err u3002))
  23
         (define-constant err-sunset-height-in-past (err u3003))
  24
  25
  26
         (define-map emergency-team principal bool)
  27
         ;; --- Authorisation check
  28
  29
         (define-public (is-dao-or-extension)
  30
                  (ok (asserts! (or (is-eq tx-sender .executor-dao) (contract-call? .executor-dao is-ex
  31
         )
  32
  33
  34
         ;; --- Internal DAO functions
  35
  36
         (define-public (set-emergency-proposal-duration (duration uint))
  37
                  (begin
  30
                          (tryl (is-dan-or-extension))
```

```
39
                        (ok (var-set emergency-proposal-duration duration))
40
               )
       )
41
42
43
       (define-public (set-emergency-team-sunset-height (height uint))
               (begin
44
                        (try! (is-dao-or-extension))
45
                        (asserts! (> height block-height) err-sunset-height-in-past)
46
                        (ok (var-set emergency-team-sunset-height height))
47
               )
48
49
       )
50
       (define-public (set-emergency-team-member (who principal) (member bool))
51
               (begin
52
53
                       (try! (is-dao-or-extension))
                        (ok (map-set emergency-team who member))
54
               )
55
       )
56
57
       ;; --- Public functions
58
59
60
       (define-read-only (is-emergency-team-member (who principal))
               (default-to false (map-get? emergency-team who))
61
       )
62
63
64
       (define-public (emergency-propose (proposal proposal-trait>))
               (begin
65
66
                        (asserts! (is-emergency-team-member tx-sender) err-not-emergency-team-member)
67
                        (asserts! (< block-height (var-get emergency-team-sunset-height)) err-sunset-
                        (contract-call? .ede001-proposal-voting add-proposal proposal
68
69
70
                                        start-block-height: block-height,
71
                                        end-block-height: (+ block-height (var-get emergency-proposal)
72
                                        proposer: tx-sender
73
                                }
                       )
74
75
               )
       )
76
77
       ;; --- Extension callback
78
79
       (define-public (callback (sender principal) (memo (buff 34)))
80
               (ok true)
81
       )
82
```

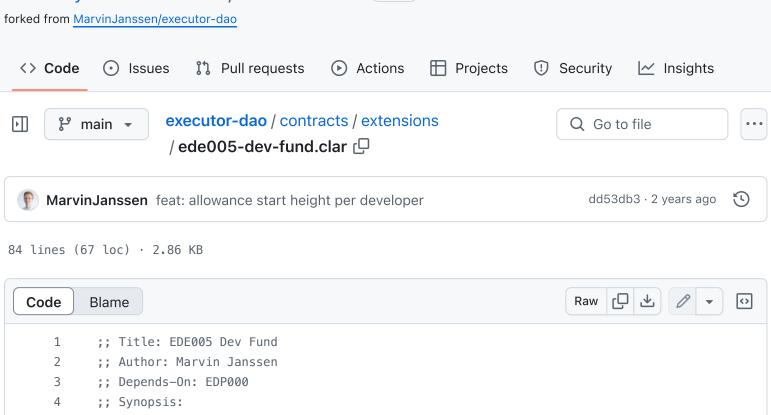
Clarity-Innovation-Lab / executor-dao Public



```
(define-constant err-unauthorised (err u3000))
20
       (define-constant err-not-executive-team-member (err u3001))
21
       (define-constant err-already-executed (err u3002))
22
       (define-constant err-sunset-height-reached (err u3003))
23
       (define-constant err-sunset-height-in-past (err u3004))
24
25
26
       (define-map executive-team principal bool)
       (define-map executive-action-signals {proposal: principal, team-member: principal} bool)
27
       (define-map executive-action-signal-count principal uint)
28
29
30
       (define-data-var executive-signals-required uint u1) ;; signals required for an executive act
31
32
       ;; --- Authorisation check
33
34
       (define-public (is-dao-or-extension)
35
               (ok (asserts! (or (is-eq tx-sender .executor-dao) (contract-call? .executor-dao is-ex
36
       )
37
30
              Internal DAO functions
```

```
39
40
       (define-public (set-executive-team-sunset-height (height uint))
41
               (begin
42
                       (try! (is-dao-or-extension))
                       (asserts! (> height block-height) err-sunset-height-in-past)
43
44
                       (ok (var-set executive-team-sunset-height height))
              )
45
      )
46
47
       (define-public (set-executive-team-member (who principal) (member bool))
48
49
               (begin
                       (try! (is-dao-or-extension))
50
51
                       (ok (map-set executive-team who member))
52
               )
      )
53
54
55
       (define-public (set-signals-required (new-requirement uint))
56
               (begin
57
                       (try! (is-dao-or-extension))
58
                       (ok (var-set executive-signals-required new-requirement))
              )
59
      )
60
61
62
      ;; --- Public functions
63
64
       (define-read-only (is-executive-team-member (who principal))
65
               (default-to false (map-get? executive-team who))
66
      )
67
      (define-read-only (has-signalled (proposal principal) (who principal))
68
69
               (default-to false (map-get? executive-action-signals {proposal: proposal, team-member
      )
70
71
72
       (define-read-only (get-signals-required)
73
               (var-get executive-signals-required)
      )
74
75
76
      (define-read-only (get-signals (proposal principal))
77
               (default-to u0 (map-get? executive-action-signal-count proposal))
78
      )
79
80
       81
               (let
                       (
82
83
                               (proposal-principal (contract-of proposal))
                               (signals (+ (get-signals proposal-principal) (if (has-signalled propo
84
                       )
85
                       (asserts! (is-executive-team-member tx-sender) err-not-executive-team-member)
86
87
                       (asserts! (< block-height (var-get executive-team-sunset-height)) err-sunset-
                       (and (>= signals (var-get executive-signals-required))
88
                               (try! (contract-call? .executor-dao execute proposal tx-sender))
89
90
91
                       (map-set executive-action-signals {proposal: proposal-principal, team-member:
                       (map-set executive-action-signal-count proposal-principal signals)
92
93
                       (ok signals)
94
```

```
95
96
97 ;; --- Extension callback
98
99 (define-public (callback (sender principal) (memo (buff 34)))
100 (ok true)
101
```



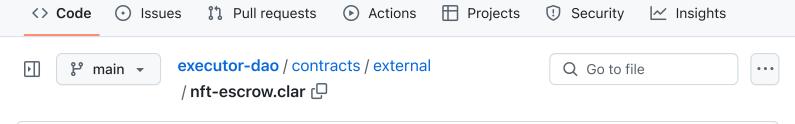
```
;; A simple pre-seeded dev fund that can pay out developers on a monthly basis.
5
       ;; Description:
6
7
       ;; Initialised by EDP001 Dev Fund. Developers can be awarded a monthly allowance
       ;; and can claim it from this extension. Principals can be added and removed, and
8
       ;; allowances can be changed via future proposals.
9
10
       (impl-trait .extension-trait.extension-trait)
11
12
       (define-constant one-month-time u4380) ;; 43,800 minutes / 10 minute average block time.
13
14
       (define-constant err-unauthorised (err u3000))
15
       (define-constant err-no-allowance (err u3001))
16
       (define-constant err-already-claimed (err u3002))
17
18
       (define-map monthly-developer-allowances principal {start-height: uint, allowance: uint})
19
       (define-map claim-counts principal uint)
20
21
       ;; --- Authorisation check
22
23
       (define-public (is-dao-or-extension)
24
               (ok (asserts! (or (is-eq tx-sender .executor-dao) (contract-call? .executor-dao is-ex
25
26
       )
27
       ;; --- Internal DAO functions
28
29
       (define-public (set-developer-allowance (start-height uint) (allowance uint) (who principal))
30
31
               (begin
32
                       (try! (is-dao-or-extension))
                       (ok (map-set monthly-developer-allowances who {start-height: start-height, al
33
34
               )
       )
35
36
       (define-private (set-developer-allowances-iter (item {start-height: uint, allowance: uint, wh
37
              (man_set month)y_developer_allowances (get who item) {start_height: (get start_height
30
```

```
)
39
40
       (define-public (set-developer-allowances (developers (list 200 {start-height: uint, allowance
41
               (begin
42
43
                        (try! (is-dao-or-extension))
                        (ok (fold set-developer-allowances-iter developers true))
44
               )
45
       )
46
47
       (define-public (transfer (amount uint) (recipient principal) (memo (optional (buff 34))))
48
49
               (begin
                       (try! (is-dao-or-extension))
50
                        (as-contract (contract-call? .ede000-governance-token transfer amount tx-send
51
               )
52
53
       )
54
       ;; --- Public functions
55
56
       (define-read-only (get-developer-allowance (who principal))
57
               (map-get? monthly-developer-allowances who)
58
       )
59
60
       (define-read-only (get-developer-claim-count (who principal))
61
62
               (default-to u0 (map-get? claim-counts who))
       )
63
64
       (define-public (claim (memo (optional (buff 34))))
65
               (let
66
                        (
67
                                (entry (unwrap! (get-developer-allowance tx-sender) err-no-allowance)
68
69
                                (claim-count (get-developer-claim-count tx-sender))
                                (start-height (get start-height entry))
70
                                (max-claims (/ (- block-height start-height) one-month-time))
71
                                (developer tx-sender)
72
73
                        )
                        (asserts! (< claim-count max-claims) err-already-claimed)</pre>
74
                        (map-set claim-counts tx-sender max-claims)
75
76
                        (as-contract (contract-call? .ede000-governance-token transfer (* (- max-clai
77
               )
       )
78
79
80
       ;; --- Extension callback
81
       (define-public (callback (sender principal) (memo (buff 34)))
82
               (ok true)
83
       )
84
```

쑥 Clarity-Innovation-Lab / executor-dao Public

radicleart Change is/set allowed to is/set allowlisted

forked from MarvinJanssen/executor-dao



0ca9017 · 2 years ago

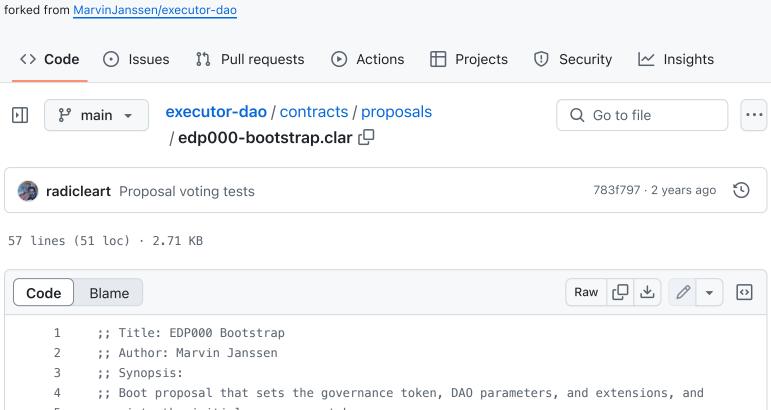
(1)

82 lines (67 loc) · 2.86 KB

```
(>)
Code
         Blame
   1
          ;; An example external contract to show how the ExecutorDAO is able to
   2
          ;; add external contracts to an allowlist. These contracts may not be aware of the DAO. See
          ;; edp003-allowlist-escrow-nft for more details.
   3
   4
   5
          (impl-trait .ownable-trait.ownable-trait)
   6
   7
          (define-constant err-not-contract-owner (err u100))
          (define-constant err-not-allowlisted (err u101))
          (define-constant err-unknown-escrow (err u102))
   9
          (define-constant err-wrong-nft (err u103))
  10
          (define-constant err-not-nft-owner (err u104))
  11
  12
          (define-data-var contract-owner principal tx-sender)
  13
  14
          (define-map nft-allowlist principal bool)
          (define-map nfts-in-escrow {token-id: uint, recipient: principal} {owner: principal, price: |
  15
  17
          (define-trait sip009-transferable
  18
                  (
  19
                          (transfer (uint principal principal) (response bool uint))
                  )
  20
          )
  21
  22
          (define-private (is-owner)
  23
  24
                  (ok (asserts! (is-eq (var-get contract-owner) tx-sender) err-not-contract-owner))
  25
  26
  27
          (define-read-only (get-contract-owner)
                  (ok (var-get contract-owner))
  28
          )
  29
  30
          (define-public (set-contract-owner (new-owner principal))
  31
  32
                  (begin
                          (try! (is-owner))
  33
  34
                          (ok (var-set contract-owner new-owner))
                  )
  35
  36
          )
  37
  30
          (define-read-only (is-allowlisted (nft principal))
```

```
(default-to false (map-get? nft-allowlist nft))
39
40
       )
41
42
       (define-public (set-allowlisted (nft principal) (enabled bool))
43
               (begin
                       (try! (is-owner))
44
                       (ok (map-set nft-allowlist nft enabled))
45
               )
46
       )
47
48
       (define-read-only (get-escrow (token-id uint) (recipient principal))
49
50
               (map-get? nfts-in-escrow {token-id: token-id, recipient: recipient})
       )
51
52
       (define-private (send-nft (token-id uint) (recipient principal) (nft <sip009-transferable>))
53
               (as-contract (contract-call? nft transfer token-id tx-sender recipient))
54
       )
55
56
       (define-public (place-in-escrow (token-id uint) (recipient principal) (amount uint) (nft <sip
57
58
               (begin
                       (asserts! (is-allowlisted (contract-of nft)) err-not-allowlisted)
59
                       (map-set nfts-in-escrow {token-id: token-id, recipient: recipient} {owner: tx
60
                       (contract-call? nft transfer token-id tx-sender (as-contract tx-sender))
61
               )
62
       )
63
64
       (define-public (pay-and-redeem (token-id uint) (nft <sip009-transferable>))
65
66
               (let ((escrow (unwrap! (get-escrow token-id tx-sender) err-unknown-escrow)))
67
                       (asserts! (is-eq (contract-of nft) (get asset escrow)) err-wrong-nft)
                       (map-delete nfts-in-escrow {token-id: token-id, recipient: tx-sender})
68
69
                       (try! (stx-transfer? (get price escrow) tx-sender (get owner escrow)))
                       (send-nft token-id tx-sender nft)
70
               )
71
       )
72
73
       (define-public (cancel-escrow (token-id uint) (recipient principal) (nft <sip009-transferable
74
               (let ((escrow (unwrap! (get-escrow token-id recipient) err-unknown-escrow)))
75
76
                       (asserts! (is-eq (get owner escrow) tx-sender) err-not-nft-owner)
                       (asserts! (is-eq (contract-of nft) (get asset escrow)) err-wrong-nft)
77
78
                       (map-delete nfts-in-escrow {token-id: token-id, recipient: recipient})
79
                       (send-nft token-id tx-sender nft)
               )
80
81
       )
```

악 Clarity-Innovation-Lab / executor-dao Public



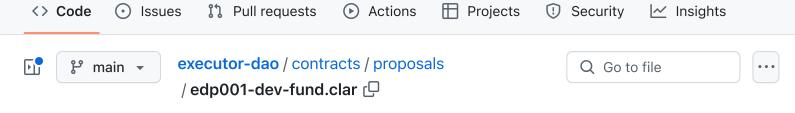
```
;; mints the initial governance tokens.
5
       ;; Description:
6
7
       ;; Mints the initial supply of governance tokens and enables the the following
       ;; extensions: "EDE000 Governance Token", "EDE001 Proposal Voting",
8
       ;; "EDE002 Proposal Submission", "EDE003 Emergency Proposals",
9
       ;; "EDE004 Emergency Execute".
10
11
       (impl-trait .proposal-trait.proposal-trait)
12
13
       (define-public (execute (sender principal))
14
15
               (begin
                       ;; Enable genesis extensions.
16
17
                       (try! (contract-call? .executor-dao set-extensions
                                (list
18
19
                                        {extension: .ede000-governance-token, enabled: true}
                                        {extension: .ede001-proposal-voting, enabled: true}
20
                                        {extension: .ede002-proposal-submission, enabled: true}
21
                                        {extension: .ede003-emergency-proposals, enabled: true}
22
                                        {extension: .ede004-emergency-execute, enabled: true}
23
                                )
24
25
                       ))
26
                       ;; Set emergency team members.
27
                       (try! (contract-call? .ede003-emergency-proposals set-emergency-team-member
28
29
                       (try! (contract-call? .ede003-emergency-proposals set-emergency-team-member
30
                       ;; Set executive team members.
31
32
                       (try! (contract-call? .ede004-emergency-execute set-executive-team-member 'ST
                       (try! (contract-call? .ede004-emergency-execute set-executive-team-member 'ST
33
34
                       (try! (contract-call? .ede004-emergency-execute set-executive-team-member 'ST
35
                       (try! (contract-call? .ede004-emergency-execute set-executive-team-member 'ST
36
                       (try! (contract-call? .ede004-emergency-execute set-signals-required u3)) ;;
37
30
                       .. Mint initial token cumply
```

```
,, nime interac concil supply:
                       (try! (contract-call? .ede000-governance-token edg-mint-many
39
                                (list
40
                                        {amount: u1000, recipient: sender}
41
42
                                        {amount: u1000, recipient: 'ST1SJ3DTE5DN7X54YDH5D64R3BCB6A2A0
                                        {amount: u1000, recipient: 'ST2CY5V39NHDPWSXMW9QDT3HC3GD6Q6XX
43
                                        {amount: u1000, recipient: 'ST2JHG361ZXG51QTKY2NQCVBPPRRE2KZE
44
45
                                        {amount: u1000, recipient: 'ST2NEB84ASENDXKYGJPQW86YXQCEFEX2Z
                                        {amount: u1000, recipient: 'ST2REHHS5J3CERCRBEPMGH7921Q6PYKA/
46
                                        {amount: u1000, recipient: 'ST3AM1A56AK2C1XAFJ4115ZSV26EB49BV
47
                                        {amount: u1000, recipient: 'ST3NBRSFKX28FQ2ZJ1MAKX58HKHSDGNV5
48
                                        {amount: u1000, recipient: 'ST3PF13W7Z0RRM42A8VZRVFQ75SV1K26F
49
                                        ;;{amount: u1000, recipient: 'STNHKEPYEPJ8ET55ZZ0M5A34J0R3N5F
50
                               )
51
52
                       ))
53
                       (print "ExecutorDAO has risen.")
54
55
                       (ok true)
               )
56
57
       )
```

Clarity-Innovation-Lab / executor-dao Public

MarvinJanssen feat: allowance start height per developer

forked from MarvinJanssen/executor-dao



dd53db3 · 2 years ago

30 lines (27 loc) · 1.3 KB

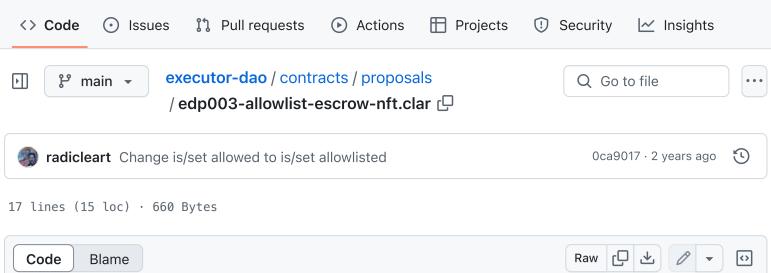
```
<>
Code
        Blame
   1
          ;; Title: EDP001 Dev Fund
   2
          ;; Author: Marvin Janssen
   3
         ;; Synopsis:
          ;; This proposal creates a simple dev fund that pays developers on a monthly basis.
   4
          ;; Description:
   5
         ;; If this proposal passes, it mints new governance tokens equal to 30% of the total
   6
          ;; supply and awards them to the EDE005 Dev Fund extension. It contains a number of
   7
         ;; principals and set allowances. Any principal with an allowance is able to claim
   8
         ;; an amount of tokens equal to the allowance on a (roughly) monthly basis.
   9
          ;; Principals can be added and removed, and allowances can be changed via future
  10
          ;; proposals.
  11
  12
         (impl-trait .proposal-trait.proposal-trait)
  13
  14
  15
         (define-constant dev-fund-percentage u30)
  16
  17
         (define-public (execute (sender principal))
                  (let
  18
  19
                          (
  20
                                  (total-supply (unwrap-panic (contract-call? .ede000-governance-token
                                  (dev-fund-amount (/ (* total-supply dev-fund-percentage) u100))
  21
  22
                          (try! (contract-call? .executor-dao set-extension .ede005-dev-fund true))
  23
                          (try! (contract-call? .ede005-dev-fund set-developer-allowances (list
  24
  25
                                  {who: 'ST2CY5V39NHDPWSXMW9QDT3HC3GD6Q6XX4CFRK9AG, start-height: block
  26
                                  {who: 'ST2JHG361ZXG51QTKY2NQCVBPPRRE2KZB1HR05NNC, start-height: block
                          )))
  27
                          (contract-call? .ede000-governance-token edg-mint dev-fund-amount .ede005-de√
  28
                  )
  29
         )
  30
```

13 lines (11 loc) · 422 Bytes

```
Raw 🖵 🕹
Code
        Blame
   1
         ;; Title: EDP002 Kill Emergency Execute
   2
         ;; Author: Marvin Janssen
   3
         ;; Synopsis:
         ;; This proposal disables extension "EDE004 Emergency Execute".
   4
   5
         ;; Description:
         ;; If this proposal passes, extension "EDE004 Emergency Execute" is immediately
   6
   7
         ;; disabled.
   8
         (impl-trait .proposal-trait.proposal-trait)
   9
  10
  11
         (define-public (execute (sender principal))
                  (contract-call? .executor-dao set-extension .ede004-emergency-execute false)
  12
  13
         )
```

쑥 Clarity-Innovation-Lab / executor-dao Public

forked from MarvinJanssen/executor-dao



```
1
       ;; Title: EDP003 Allowlist Escrow NFT
 2
       ;; Author: Marvin Janssen
 3
       ;; Synopsis:
       ;; An example proposal to illustrate how ExecutorDAO can manage external
 4
 5
       ;; ownable contracts.
       ;; Description:
 6
 7
       ;; ExecutorDAO is well-equiped to manage external contracts feature have
       ;; some form of ownership. This proposal updates the allowlist of an
8
       ;; example escrow contract that is owned by the ExecutorDAO contract.
9
       ;; Note that the ExecutorDAO contract must be the owner of nft-escrow
10
       ;; for this proposal to be executed.
11
12
       (impl-trait .proposal-trait.proposal-trait)
13
14
       (define-public (execute (sender principal))
15
               (contract-call? .nft-escrow set-allowlisted .some-nft true)
16
17
       )
```