

 10 Branches

 0 Tags



 Go to file


Go to file










 Code



This branch is **17 commits ahead of**, **3 commits behind** [MarvinJanssen/executor-dao:main](#).








 **MarvinJanssen** Merge pull request **#10** from Clarity-Innovation-Lab/feat/simplify-... 

a8f218e · 2 years ago 

 .vscode	Initial commit	3 years ago
 contracts	feat: simplify ede001 and ede0...	2 years ago
 cvm	Initial commit	3 years ago
 settings	Initial commit	3 years ago
 tests	feat: simplify ede001 and ede0...	2 years ago
 .gitignore	Initial commit	3 years ago
 Clarinet.toml	feat: simplify ede001 and ede0...	2 years ago
 LICENSE	Initial commit	3 years ago
 README.md	Change is/set allowed to is/set ...	2 years ago

About

The one DAO to rule them all. A modular DAO written in Clarity for the Stacks blockchain.

-  Readme
-  MIT license
-  Activity
-  Custom properties
-  1 star
-  1 watching
-  1 fork

Report repository

Releases

No releases published

Packages

No packages published

Languages

-  README
-  MIT license



ExecutorDAO

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.
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:

```
;; 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.)



main ▾

executor-dao / [contracts](#) / **executor-dao.clar**



Go to file



MarvinJanssen Initial commit

aa748ce · 3 years ago



82 lines (65 loc) · 2.49 KB

Code

Blame

Raw



```

1  ;; ExecutorDAO is the one DAO to rule them all.
2  ;; By Marvin Janssen
3
4  (use-trait proposal-trait .proposal-trait.proposal-trait)
5  (use-trait extension-trait .extension-trait.extension-trait)
6
7  (define-constant err-unauthorised (err u1000))
8  (define-constant err-already-executed (err u1001))
9  (define-constant err-invalid-extension (err u1002))
10
11 (define-data-var executive principal tx-sender)
12 (define-map executed-proposals principal uint)
13 (define-map extensions principal bool)
14
15 ;; --- Authorisation check
16
17 (define-private (is-self-or-extension)
18   (ok (asserts! (or (is-eq tx-sender (as-contract tx-sender)) (is-extension contract-ca
19 )
20
21 ;; --- Extensions
22
23 (define-read-only (is-extension (extension principal))
24   (default-to false (map-get? extensions extension)))
25 )
26
27 (define-public (set-extension (extension principal) (enabled bool))
28   (begin
29     (try! (is-self-or-extension))
30     (print {event: "extension", extension: extension, enabled: enabled})
31     (ok (map-set extensions extension enabled))
32   )
33 )
34
35 (define-private (set-extensions-iter (item {extension: principal, enabled: bool}))
36   (begin
37     (print {event: "extension", extension: (get extension item), enabled: (get er
38     (map-set extensions (get extension item) (get enabled item))

```



```

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

```

forked from [MarvinJanssen/executor-dao](#)

[Code](#) [Issues](#) [Pull requests](#) [Actions](#) [Projects](#) [Security](#) [Insights](#)

 [main](#) [executor-dao / contracts / traits](#)
[/ extension-trait.clar](#) 

 Go to file



MarvinJanssen Initial commit

aa748ce · 3 years ago



5 lines (5 loc) · 94 Bytes

Code

Blame



Raw



```
1  (define-trait extension-trait
2      (
3          (callback (principal (buff 34)) (response bool uint))
4      )
5  )
```


forked from [MarvinJanssen/executor-dao](#)

[Code](#) [Issues](#) [Pull requests](#) [Actions](#) [Projects](#) [Security](#) [Insights](#)

 [main](#) [executor-dao / contracts / traits](#)
[/ governance-token-trait.clar](#) 

 Go to file



MarvinJanssen Initial commit

aa748ce · 3 years ago



12 lines (12 loc) · 490 Bytes

Code

Blame



Raw



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


forked from [MarvinJanssen/executor-dao](#)

[Code](#) [Issues](#) [Pull requests](#) [Actions](#) [Projects](#) [Security](#) [Insights](#)

 [main](#) [executor-dao / contracts / traits](#)
[/ ownable-trait.clar](#) 

 Go to file



MarvinJanssen Initial commit

aa748ce · 3 years ago



6 lines (6 loc) · 144 Bytes

Code

Blame

Raw



```
1      (define-trait ownable-trait
2          (
3              (get-contract-owner () (response principal uint))
4              (set-contract-owner (principal) (response bool uint))
5          )
6      )
```


forked from [MarvinJanssen/executor-dao](#)

[Code](#) [Issues](#) [Pull requests](#) [Actions](#) [Projects](#) [Security](#) [Insights](#)

 [main](#) [executor-dao / contracts / traits](#)
[/ proposal-trait.clar](#) 

 Go to file



MarvinJanssen Initial commit

aa748ce · 3 years ago



5 lines (5 loc) · 82 Bytes


Code


Blame

Raw



```
1      (define-trait proposal-trait
2          (
3              (execute (principal) (response bool uint))
4          )
5      )
```



[main](#)
[executor-dao / contracts / traits](#)

[/ sip010-ft-trait.clar](#)


Go to file



MarvinJanssen Initial commit

aa748ce · 3 years ago



24 lines (18 loc) · 812 Bytes

Code

Blame



Raw



```

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

```


 [main](#) **executor-dao** / [contracts](#) / [extensions](#)
/ **ede000-governance-token.clar** 

 Go to file



radicleart Proposal voting tests

783f797 · 2 years ago



168 lines (132 loc) · 4.1 KB

Code

Blame

Raw



```

1  ;; Title: EDE000 Governance Token
2  ;; Author: Marvin Janssen
3  ;; Depends-On:
4  ;; Synopsis:
5  ;; This extension defines the governance token of ExecutorDAO.
6  ;; Description:
7  ;; The governance token is a simple SIP010-compliant fungible token
8  ;; with some added functions to make it easier to manage by
9  ;; ExecutorDAO proposals and extensions.
10
11  (impl-trait .governance-token-trait.governance-token-trait)
12  (impl-trait .sip010-ft-trait.sip010-ft-trait)
13  (impl-trait .extension-trait.extension-trait)
14
15  (define-constant err-unauthorised (err u3000))
16  (define-constant err-not-token-owner (err u4))
17
18  (define-fungible-token edg-token)
19  (define-fungible-token edg-token-locked)
20
21  (define-data-var token-name (string-ascii 32) "ExecutorDAO Governance Token")
22  (define-data-var token-symbol (string-ascii 10) "EDG")
23  (define-data-var token-uri (optional (string-utf8 256)) none)
24  (define-data-var token-decimals uint u6)
25
26  ;; --- Authorisation check
27
28  (define-public (is-dao-or-extension)
29    (ok (asserts! (or (is-eq tx-sender .executor-dao) (contract-call? .executor-dao is-ex
30  )
31  )
32  ;; --- Internal DAO functions
33
34  ;; governance-token-trait
35
36  (define-public (edg-transfer (amount uint) (sender principal) (recipient principal))
37    (begin
38      (try! (is-dao-or-extension))

```



```

38      (try! (is-dao-or-extension))
39      (ft-transfer? edg-token amount sender recipient)
40    )
41  )
42
43  (define-public (edg-lock (amount uint) (owner principal))
44    (begin
45      (try! (is-dao-or-extension))
46      (try! (ft-burn? edg-token amount owner))
47      (ft-mint? edg-token-locked amount owner)
48    )
49  )
50
51  (define-public (edg-unlock (amount uint) (owner principal))
52    (begin
53      (try! (is-dao-or-extension))
54      (try! (ft-burn? edg-token-locked amount owner))
55      (ft-mint? edg-token amount owner)
56    )
57  )
58
59  (define-public (edg-mint (amount uint) (recipient principal))
60    (begin
61      (try! (is-dao-or-extension))
62      (ft-mint? edg-token amount recipient)
63    )
64  )
65
66  (define-public (edg-burn (amount uint) (owner principal))
67    (begin
68      (try! (is-dao-or-extension))
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
78      (try! (is-dao-or-extension))
79      (ok (var-set token-name new-name))
80    )
81  )
82
83  (define-public (set-symbol (new-symbol (string-ascii 10)))
84    (begin
85      (try! (is-dao-or-extension))
86      (ok (var-set token-symbol new-symbol))
87    )
88  )
89
90  (define-public (set-decimals (new-decimals uint))
91    (begin
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
108     (define-public (edg-mint-many (recipients (list 200 {amount: uint, recipient: principal})))
109         (begin
110             (try! (is-dao-or-extension))
111             (ok (map edg-mint-many-iter recipients))
112         )
113     )
114
115     ;; --- Public functions
116
117     ;; sip010-ft-trait
118
119     (define-public (transfer (amount uint) (sender principal) (recipient principal) (memo (optional string-utf8 256)))
120         (begin
121             (asserts! (or (is-eq tx-sender sender) (is-eq contract-caller sender)) err-no-sender)
122             (ft-transfer? edg-token amount sender recipient)
123         )
124     )
125
126     (define-read-only (get-name)
127         (ok (var-get token-name))
128     )
129
130     (define-read-only (get-symbol)
131         (ok (var-get token-symbol))
132     )
133
134     (define-read-only (get-decimals)
135         (ok (var-get token-decimals))
136     )
137
138     (define-read-only (get-balance (who principal))
139         (ok (+ (ft-get-balance edg-token who) (ft-get-balance edg-token-locked who)))
140     )
141
142     (define-read-only (get-total-supply)
143         (ok (+ (ft-get-supply edg-token) (ft-get-supply edg-token-locked)))
144     )
145
146     (define-read-only (get-token-uri)
147         (ok (var-get token-uri))
148     )
149
150     ;; governance-token-trait

```

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

 [main](#) **executor-dao / contracts / extensions**
/ede001-proposal-voting.clar 

 Go to file



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



140 lines (118 loc) · 4.93 KB

Code [Blame](#)

Raw



```

1  ;; Title: EDE001 Proposal Voting
2  ;; Author: Marvin Janssen
3  ;; Depends-On: EDE000
4  ;; Synopsis:
5  ;; This extension is part of the core of ExecutorDAO. It allows governance token
6  ;; holders to vote on and conclude proposals.
7  ;; Description:
8  ;; Once proposals are submitted, they are open for voting after a lead up time
9  ;; passes. Any token holder may vote on an open proposal, where one token equals
10 ;; one vote. Members can vote until the voting period is over. After this period
11 ;; anyone may trigger a conclusion. The proposal will then be executed if the
12 ;; votes in favour exceed the ones against.
13
14 (impl-trait .extension-trait.extension-trait)
15 (use-trait proposal-trait .proposal-trait.proposal-trait)
16
17 (define-constant err-unauthorised (err u3000))
18 (define-constant err-proposal-already-executed (err u3002))
19 (define-constant err-proposal-already-exists (err u3003))
20 (define-constant err-unknown-proposal (err u3004))
21 (define-constant err-proposal-already-concluded (err u3005))
22 (define-constant err-proposal-inactive (err u3006))
23 (define-constant err-proposal-not-concluded (err u3007))
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
28 (define-map proposals
29   principal
30   {
31     votes-for: uint,
32     votes-against: uint,
33     start-block-height: uint,
34     end-block-height: uint,
35     concluded: bool,
36     passed: bool,
37     proposer: principal
38   }

```

```

39 )
40
41 (define-map member-total-votes {proposal: principal, voter: principal} uint)
42
43 ;; --- Authorisation check
44
45 (define-public (is-dao-or-extension)
46   (ok (asserts! (or (is-eq tx-sender .executor-dao) (contract-call? .executor-dao is-ex
47 )
48
49 ;; --- Internal DAO functions
50
51 ;; Proposals
52
53 (define-public (add-proposal (proposal <proposal-trait>) (data {start-block-height: uint, end
54   (begin
55     (try! (is-dao-or-extension))
56     (asserts! (is-none (contract-call? .executor-dao executed-at proposal)) err-p
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
66 (define-read-only (get-proposal-data (proposal principal))
67   (map-get? proposals proposal)
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-pro
80     )
81     (asserts! (>= block-height (get start-block-height proposal-data)) err-propos
82     (asserts! (< block-height (get end-block-height proposal-data)) err-proposal-
83     (map-set member-total-votes {proposal: proposal, voter: tx-sender}
84       (+ (get-current-total-votes proposal tx-sender) amount)
85     )
86     (map-set proposals proposal
87       (if for
88         (merge proposal-data {votes-for: (+ (get votes-for proposal-c
89         (merge proposal-data {votes-against: (+ (get votes-against pr
90       )
91     )
92     (print {event: "vote", proposal: proposal, voter: tx-sender, for: for, amount
93     (contract-call? .ede000-governance-token edg-lock amount tx-sender)
94   )

```

```

95     )
96
97 ;; Conclusion
98
99 (define-public (conclude (proposal <proposal-trait>))
100     (let
101         (
102             (proposal-data (unwrap! (map-get? proposals (contract-of proposal)) e
103             (passed (> (get votes-for proposal-data) (get votes-against proposal-
104             )
105             (asserts! (not (get concluded proposal-data)) err-proposal-already-concluded)
106             (asserts! (>= block-height (get end-block-height proposal-data)) err-end-bloc
107             (map-set proposals (contract-of proposal) (merge proposal-data {concluded: tr
108             (print {event: "conclude", proposal: proposal, passed: passed})
109             (and passed (try! (contract-call? .executor-dao execute proposal tx-sender)))
110             (ok passed)
111         )
112     )
113
114 ;; Reclamation
115
116 (define-public (reclaim-votes (proposal <proposal-trait>))
117     (let
118         (
119             (proposal-principal (contract-of proposal))
120             (proposal-data (unwrap! (map-get? proposals proposal-principal) err-u
121             (votes (unwrap! (map-get? member-total-votes {proposal: proposal-prin
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))
132         (vote amount for proposal)
133     )
134 )
135
136 ;; --- Extension callback
137
138 (define-public (callback (sender principal) (memo (buff 34)))
139     (ok true)
140 )

```

 [main](#) **executor-dao / contracts / extensions**
/ede002-proposal-submission.clar 

 Go to file



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



95 lines (77 loc) · 3.45 KB

Code

Blame

Raw



```


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

```

```

38 ;; 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
50 (define-private (set-parameters-iter (item {parameter: (string-ascii 34), value: uint}) (prev
51   (begin
52     (try! previous)
53     (try! (get-parameter (get parameter item)))
54     (ok (map-set parameters (get parameter item) (get value item)))
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
65 ;; --- Public functions
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 (define-public (propose (proposal <proposal-trait>) (start-block-height uint))
76   (begin
77     (asserts! (>= start-block-height (+ block-height (try! (get-parameter "minimu
78     (asserts! (<= start-block-height (+ block-height (try! (get-parameter "maximu
79     (asserts! (unwrap-panic (contract-call? .ede000-governance-token edg-has-perc
80     (contract-call? .ede001-proposal-voting add-proposal
81       proposal
82       {
83         start-block-height: start-block-height,
84         end-block-height: (+ start-block-height (try! (get-parameter
85         proposer: tx-sender
86       }
87     )
88   )
89 )
90
91 ;; --- Extension callback
92
93 (define-public (callback (sender principal) (memo (buff 34)))
94   (ok true)

```


 [main](#) **executor-dao / contracts / extensions**
/ede003-emergency-proposals.clar 

 Go to file



MarvinJanssen Initial commit

aa748ce · 3 years ago



82 lines (67 loc) · 2.64 KB

Code

Blame

Raw



```

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

```

```

38      (try! (is-dao-or-extension))
39      (ok (var-set emergency-proposal-duration duration))
40    )
41  )
42
43  (define-public (set-emergency-team-sunset-height (height uint))
44    (begin
45      (try! (is-dao-or-extension))
46      (asserts! (> height block-height) err-sunset-height-in-past)
47      (ok (var-set emergency-team-sunset-height height))
48    )
49  )
50
51  (define-public (set-emergency-team-member (who principal) (member bool))
52    (begin
53      (try! (is-dao-or-extension))
54      (ok (map-set emergency-team who member))
55    )
56  )
57
58  ;; --- Public functions
59
60  (define-read-only (is-emergency-team-member (who principal))
61    (default-to false (map-get? emergency-team who))
62  )
63
64  (define-public (emergency-propose (proposal <proposal-trait>))
65    (begin
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-
68      (contract-call? .ede001-proposal-voting add-proposal proposal
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
78  ;; --- Extension callback
79
80  (define-public (callback (sender principal) (memo (buff 34)))
81    (ok true)
82  )

```

 [main](#) **executor-dao / contracts / extensions**
/ede004-emergency-execute.clar 

 Go to file



MarvinJanssen Initial commit

aa748ce · 3 years ago



101 lines (82 loc) · 3.49 KB

Code

Blame

Raw



```

1  ;; Title: EDE004 Emergency Execute
2  ;; Author: Marvin Janssen
3  ;; Depends-On:
4  ;; Synopsis:
5  ;; This extension allows a small number of very trusted principals to immediately
6  ;; execute a proposal once a super majority is reached.
7  ;; Description:
8  ;; An extension meant for the bootstrapping period of a DAO. It temporarily gives
9  ;; some very trusted principals the ability to perform an "executive action";
10 ;; meaning, they can skip the voting process to immediately executive a proposal.
11 ;; The Emergency Executive extension has a sunset period of ~1 month from deploy
12 ;; time. Executive Team members, the parameters, and sunset period may be changed
13 ;; by means of a future proposal.
14
15 (impl-trait .extension-trait.extension-trait)
16 (use-trait proposal-trait .proposal-trait.proposal-trait)
17
18 (define-data-var executive-team-sunset-height uint (+ block-height u4380)) ;; ~1 month from c
19
20 (define-constant err-unauthorised (err u3000))
21 (define-constant err-not-executive-team-member (err u3001))
22 (define-constant err-already-executed (err u3002))
23 (define-constant err-sunset-height-reached (err u3003))
24 (define-constant err-sunset-height-in-past (err u3004))
25
26 (define-map executive-team principal bool)
27 (define-map executive-action-signals {proposal: principal, team-member: principal} bool)
28 (define-map executive-action-signal-count principal uint)
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
38   ;; --- Internal DAO functions



```

```

39
40 (define-public (set-executive-team-sunset-height (height uint))
41   (begin
42     (try! (is-dao-or-extension))
43     (asserts! (> height block-height) err-sunset-height-in-past)
44     (ok (var-set executive-team-sunset-height height))
45   )
46 )
47
48 (define-public (set-executive-team-member (who principal) (member bool))
49   (begin
50     (try! (is-dao-or-extension))
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
68 (define-read-only (has-signalled (proposal principal) (who principal))
69   (default-to false (map-get? executive-action-signals {proposal: proposal, team-member:
70   }
71   ))
72 )
73
74 (define-read-only (get-signals-required)
75   (var-get executive-signals-required)
76 )
77
78 (define-read-only (get-signals (proposal principal))
79   (default-to u0 (map-get? executive-action-signal-count proposal))
80 )
81
82 (define-public (executive-action (proposal <proposal-trait>))
83   (let
84     (
85       (proposal-principal (contract-of proposal))
86       (signals (+ (get-signals proposal-principal) (if (has-signalled propo
87       )
88       (asserts! (is-executive-team-member tx-sender) err-not-executive-team-member)
89       (asserts! (< block-height (var-get executive-team-sunset-height)) err-sunset-
90       (and (>= signals (var-get executive-signals-required))
91       (try! (contract-call? .executor-dao execute proposal tx-sender))
92       )
93       (map-set executive-action-signals {proposal: proposal-principal, team-member:
94       (map-set executive-action-signal-count proposal-principal signals)
95       (ok signals)
96     )
97   )

```

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

 [main](#) **executor-dao** / [contracts](#) / [extensions](#)
/ **ede005-dev-fund.clar** 

 Go to file



MarvinJanssen feat: allowance start height per developer

dd53db3 · 2 years ago



84 lines (67 loc) · 2.86 KB

Code

Blame

Raw



```



1  ;; Title: EDE005 Dev Fund
2  ;; Author: Marvin Janssen
3  ;; Depends-On: EDP000
4  ;; Synopsis:
5  ;; A simple pre-seeded dev fund that can pay out developers on a monthly basis.
6  ;; Description:
7  ;; Initialised by EDP001 Dev Fund. Developers can be awarded a monthly allowance
8  ;; and can claim it from this extension. Principals can be added and removed, and
9  ;; allowances can be changed via future proposals.
10
11  (impl-trait .extension-trait.extension-trait)
12
13  (define-constant one-month-time u4380) ;; 43,800 minutes / 10 minute average block time.
14
15  (define-constant err-unauthorised (err u3000))
16  (define-constant err-no-allowance (err u3001))
17  (define-constant err-already-claimed (err u3002))
18
19  (define-map monthly-developer-allowances principal {start-height: uint, allowance: uint})
20  (define-map claim-counts principal uint)
21
22  ;; --- Authorisation check
23
24  (define-public (is-dao-or-extension)
25    (ok (asserts! (or (is-eq tx-sender .executor-dao) (contract-call? .executor-dao is-ex
26  )
27
28  ;; --- Internal DAO functions
29
30  (define-public (set-developer-allowance (start-height uint) (allowance uint) (who principal))
31    (begin
32      (try! (is-dao-or-extension))
33      (ok (map-set monthly-developer-allowances who {start-height: start-height, al
34    )
35  )
36
37  (define-private (set-developer-allowances-iter (item {start-height: uint, allowance: uint, wh
38    (map-set monthly-developer-allowances (get who item) {start-height: (get start-height

```

```

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

```


 [main](#) **executor-dao / contracts / external**
/ nft-escrow.clar 

 Go to file



 **radicleart** Change is/set allowed to is/set allowlisted

0ca9017 · 2 years ago



82 lines (67 loc) · 2.86 KB

Code [Blame](#)

[Raw](#)



```


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
3  ;; edp003-allowlist-escrow-nft for more details.
4
5  (impl-trait .ownable-trait.ownable-trait)
6
7  (define-constant err-not-contract-owner (err u100))
8  (define-constant err-not-allowlisted (err u101))
9  (define-constant err-unknown-escrow (err u102))
10 (define-constant err-wrong-nft (err u103))
11 (define-constant err-not-nft-owner (err u104))
12
13 (define-data-var contract-owner principal tx-sender)
14 (define-map nft-allowlist principal bool)
15 (define-map nfts-in-escrow {token-id: uint, recipient: principal} {owner: principal, price: u
16
17 (define-trait sip009-transferable
18   (
19     (transfer (uint principal principal) (response bool uint))
20   )
21 )
22
23 (define-private (is-owner)
24   (ok (asserts! (is-eq (var-get contract-owner) tx-sender) err-not-contract-owner))
25 )
26
27 (define-read-only (get-contract-owner)
28   (ok (var-get contract-owner))
29 )
30
31 (define-public (set-contract-owner (new-owner principal))
32   (begin
33     (try! (is-owner))
34     (ok (var-set contract-owner new-owner))
35   )
36 )
37
38 (define-read-only (is-allowlisted (nft principal))

```

```

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

```

 [main](#) **executor-dao / contracts / proposals**
/ edp000-bootstrap.clar 

 Go to file



radicleart Proposal voting tests

783f797 · 2 years ago



57 lines (51 loc) · 2.71 KB

Code [Blame](#)

Raw



```

1  ;; Title: EDP000 Bootstrap
2  ;; Author: Marvin Janssen
3  ;; Synopsis:
4  ;; Boot proposal that sets the governance token, DAO parameters, and extensions, and
5  ;; mints the initial governance tokens.
6  ;; Description:
7  ;; Mints the initial supply of governance tokens and enables the the following
8  ;; extensions: "EDE000 Governance Token", "EDE001 Proposal Voting",
9  ;; "EDE002 Proposal Submission", "EDE003 Emergency Proposals",
10 ;; "EDE004 Emergency Execute".
11
12 (impl-trait .proposal-trait.proposal-trait)
13
14 (define-public (execute (sender principal))
15   (begin
16     ;; Enable genesis extensions.
17     (try! (contract-call? .executor-dao set-extensions
18       (list
19         {extension: .ede000-governance-token, enabled: true}
20         {extension: .ede001-proposal-voting, enabled: true}
21         {extension: .ede002-proposal-submission, enabled: true}
22         {extension: .ede003-emergency-proposals, enabled: true}
23         {extension: .ede004-emergency-execute, enabled: true}
24       )
25     ))
26
27     ;; Set emergency team members.
28     (try! (contract-call? .ede003-emergency-proposals set-emergency-team-member 'ST
29     (try! (contract-call? .ede003-emergency-proposals set-emergency-team-member 'ST
30
31     ;; Set executive team members.
32     (try! (contract-call? .ede004-emergency-execute set-executive-team-member 'ST
33     (try! (contract-call? .ede004-emergency-execute set-executive-team-member 'ST
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
38     ;; Mint initial token supply

```

```

38 // mint initial token supply
39 (try! (contract-call? .ede000-governance-token edg-mint-many
40     (list
41         {amount: u1000, recipient: sender}
42         {amount: u1000, recipient: 'ST1SJ3DTE5DN7X54YDH5D64R3BCB6A2AC
43         {amount: u1000, recipient: 'ST2CY5V39NHDPWSXMW9QDT3HC3GD6Q6XX
44         {amount: u1000, recipient: 'ST2JHG361ZXG51QTKY2NQCVBPPRRE2KZE
45         {amount: u1000, recipient: 'ST2NEB84ASENDXKYGJPQW86YXQCEFEX2Z
46         {amount: u1000, recipient: 'ST2REHHS5J3CERCRCBEPMGH7921Q6PYKAA
47         {amount: u1000, recipient: 'ST3AM1A56AK2C1XAFJ4115ZSV26EB49BV
48         {amount: u1000, recipient: 'ST3NBRSFKX28FQ2ZJ1MAKX58HKHSDGNV5
49         {amount: u1000, recipient: 'ST3PF13W7Z0RRM42A8VZRVFQ75SV1K26F
50         ;;{amount: u1000, recipient: 'STNHKEPYEPJ8ET55ZZ0M5A34J0R3N5F
51     )
52 ))
53
54 (print "ExecutorDAO has risen.")
55 (ok true)
56 )
57 )

```

 [main](#) **executor-dao / contracts / proposals**
/ edp001-dev-fund.clar 

 Go to file



MarvinJanssen feat: allowance start height per developer

dd53db3 · 2 years ago



30 lines (27 loc) · 1.3 KB

Code

Blame

Raw



```

1      ;; Title: EDP001 Dev Fund
2      ;; Author: Marvin Janssen
3      ;; Synopsis:
4      ;; This proposal creates a simple dev fund that pays developers on a monthly basis.
5      ;; Description:
6      ;; If this proposal passes, it mints new governance tokens equal to 30% of the total
7      ;; supply and awards them to the EDE005 Dev Fund extension. It contains a number of
8      ;; principals and set allowances. Any principal with an allowance is able to claim
9      ;; an amount of tokens equal to the allowance on a (roughly) monthly basis.
10     ;; Principals can be added and removed, and allowances can be changed via future
11     ;; proposals.
12
13     (impl-trait .proposal-trait.proposal-trait)
14
15     (define-constant dev-fund-percentage u30)
16
17     (define-public (execute (sender principal))
18         (let
19             (
20                 (total-supply (unwrap-panic (contract-call? .ede000-governance-token
21                 (dev-fund-amount (/ (* total-supply dev-fund-percentage) u100))
22             )
23             (try! (contract-call? .executor-dao set-extension .ede005-dev-fund true))
24             (try! (contract-call? .ede005-dev-fund set-developer-allowances (list
25                 {who: 'ST2CY5V39NHDPWSXMW9QDT3HC3GD6Q6XX4CFRK9AG, start-height: block
26                 {who: 'ST2JHG361ZXG51QTKY2NQCVBPPPRE2KZB1HR05NNC, start-height: block
27             )))
28             (contract-call? .ede000-governance-token edg-mint dev-fund-amount .ede005-dev
29         )
30     )

```


forked from [MarvinJanssen/executor-dao](#)

[Code](#) [Issues](#) [Pull requests](#) [Actions](#) [Projects](#) [Security](#) [Insights](#)



 main ▾

executor-dao / [contracts](#) / [proposals](#)

/ **edp002-kill-emergency-execute.clar** 

 Go to file



MarvinJanssen Initial commit

aa748ce · 3 years ago



13 lines (11 loc) · 422 Bytes

Code

Blame

Raw



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


 [main](#) **executor-dao / contracts / proposals**
/ edp003-allowlist-escrow-nft.clar 

 Go to file



 **radicleart** Change is/set allowed to is/set allowlisted

0ca9017 · 2 years ago



17 lines (15 loc) · 660 Bytes

Code [Blame](#)

[Raw](#)



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

