radicleart Switched data to event based - much simpler and lots of dead code can...

3fb66ba · 5 months ago

Executable File · 87 lines (70 loc) · 2.69 KB

Code Blame Raw

```
;; Bitcoin DAO
;; Author Mike Cohen - based on Marvin Janssen' Executor DAO
;; Synopsis:
;; bitcoin-dao is the core of the dao framework.
;; Description:
;; Valid extensions must be registered here. The DAO is bootstrapped
;; by calling construct with a bootstrap proposal.

(use-trait extension-trait .extension-trait.extension-trait)
(use-trait proposal-trait .proposal-trait.proposal-trait)

(define-constant err-unauthorised (err u1000))
(define-constant err-already-executed (err u1001))
(define-constant err-invalid-extension (err u1002))

(define-data-var executive principal tx-sender)
(define-map executed-proposals principal uint)
(define-map extensions principal bool)

;; --- Authorisation check

(define-private (is-self-or-extension)
        (ok (asserts! (or (is-eq tx-sender (as-contract tx-sender)) (is-extension contract-ca
)

;; --- Extensions

(define-read-only (is-extension (extension principal))
        (default-to false (map-get? extensions extension))
)

(define-public (set-extension (extension principal) (enabled bool))
        (begin
                (try! (is-self-or-extension))
                (print {event: "extension", extension: extension, enabled: enabled})
                (ok (map-set extensions extension enabled))
        )
```

```
)

(define-private (set-extensions-iter (item {extension: principal, enabled: bool}))
	(begin
		(print {event: "extension", extension: (get extension item), enabled: (get er
		(map-set extensions (get extension item) (get enabled item))
	)
)

(define-public (set-extensions (extension-list (list 200 {extension: principal, enabled: bool
	(begin
		(try! (is-self-or-extension))
		(ok (map set-extensions-iter extension-list))
	)
)

;; --- Proposals

(define-read-only (executed-at (proposal <proposal-trait>))
	(map-get? executed-proposals (contract-of proposal))
)

(define-public (execute (proposal <proposal-trait>) (sender principal))
	(begin
		(try! (is-self-or-extension))
		(asserts! (map-insert executed-proposals (contract-of proposal) block-height)
		(print {event: "execute", proposal: proposal})
		(as-contract (contract-call? proposal execute sender))
	)
)

;; --- Bootstrap

(define-public (construct (proposal <proposal-trait>))
	(let ((sender tx-sender))
		(asserts! (is-eq sender (var-get executive)) err-unauthorised)
		(var-set executive (as-contract tx-sender))
		(as-contract (execute proposal sender))
	)
)

;; --- Extension requests

(define-public (request-extension-callback (extension <extension-trait>) (memo (buff 34)))
	(let ((sender tx-sender))
		(asserts! (is-extension contract-caller) err-invalid-extension)
		(asserts! (is-eq contract-caller (contract-of extension)) err-invalid-extensi
		(as-contract (contract-call? extension callback sender memo))
	)
)
```

radicleart Switched data to event based - much simpler and lots of dead code can...

3fb66ba · 5 months ago

Executable File · 168 lines (132 loc) · 4.14 KB

Code Blame Raw

```
;; Title: BDE000 Governance Token
;; Author: Mike Cohen (based upon work of Marvin Janssen)
;; Depends-On:
;; Synopsis:
;; This extension defines the governance token of Bitcoin DAO.
;; Description:
;; The governance token is a simple SIP010-compliant fungible token
;; with some added functions to make it easier to manage by
;; Bitcoin DAO proposals and extensions.

(impl-trait .governance-token-trait.governance-token-trait)
(impl-trait .sip010-ft-trait.sip010-ft-trait)
(impl-trait .extension-trait.extension-trait)

(define-constant err-unauthorised (err u3000))
(define-constant err-not-token-owner (err u4))

(define-fungible-token bdg-token)
(define-fungible-token bdg-token-locked)

(define-data-var token-name (string-ascii 32) "{{token_name}}")
(define-data-var token-symbol (string-ascii 10) "{{symbol}}")
(define-data-var token-uri (optional (string-utf8 256)) (some u"{{token_uri}}"))
(define-data-var token-decimals uint u6)

;; --- Authorisation check

(define-public (is-dao-or-extension)
        (ok (asserts! (or (is-eq tx-sender .bitcoin-dao) (contract-call? .bitcoin-dao is-exte
)

;; --- Internal DAO functions

;; governance-token-trait

(define-public (bdg-transfer (amount uint) (sender principal) (recipient principal))
        (begin
```

```
                (try! (is-dao-or-extension))
                (ft-transfer? bdg-token amount sender recipient)
        )
)

(define-public (bdg-lock (amount uint) (owner principal))
        (begin
                (try! (is-dao-or-extension))
                (try! (ft-burn? bdg-token amount owner))
                (ft-mint? bdg-token-locked amount owner)
        )
)

(define-public (bdg-unlock (amount uint) (owner principal))
        (begin
                (try! (is-dao-or-extension))
                (try! (ft-burn? bdg-token-locked amount owner))
                (ft-mint? bdg-token amount owner)
        )
)

(define-public (bdg-mint (amount uint) (recipient principal))
        (begin
                (try! (is-dao-or-extension))
                (ft-mint? bdg-token amount recipient)
        )
)

(define-public (bdg-burn (amount uint) (owner principal))
        (begin
                (try! (is-dao-or-extension))
                (ft-burn? bdg-token amount owner)

        )
)

;; Other

(define-public (set-name (new-name (string-ascii 32)))
        (begin
                (try! (is-dao-or-extension))
                (ok (var-set token-name new-name))
        )
)

(define-public (set-symbol (new-symbol (string-ascii 10)))
        (begin
                (try! (is-dao-or-extension))
                (ok (var-set token-symbol new-symbol))
        )
)

(define-public (set-decimals (new-decimals uint))
        (begin
                (try! (is-dao-or-extension))
                (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 (bdg-mint-many-iter (item {amount: uint, recipient: principal}))
105             (ft-mint? bdg-token (get amount item) (get recipient item))
106     )
107
108     (define-public (bdg-mint-many (recipients (list 200 {amount: uint, recipient: principal})))
109             (begin
110                     (try! (is-dao-or-extension))
111                     (ok (map bdg-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 (optior
120             (begin
121                     (asserts! (or (is-eq tx-sender sender) (is-eq contract-caller sender)) err-no
122                     (ft-transfer? bdg-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 bdg-token who) (ft-get-balance bdg-token-locked who)))
140     )
141
142     (define-read-only (get-total-supply)
143             (ok (+ (ft-get-supply bdg-token) (ft-get-supply bdg-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 (bdg-get-balance (who principal))
153             (get-balance who)
154     )
155
156     (define-read-only (bdg-has-percentage-balance (who principal) (factor uint))
157             (ok (>= (* (unwrap-panic (get-balance who)) factor) (* (unwrap-panic (get-total-suppl
158     )
159
160     (define-read-only (bdg-get-locked (owner principal))
161             (ok (ft-get-balance bdg-token-locked owner))
162     )
163
164     ;; --- Extension callback
165
166     (define-public (callback (sender principal) (memo (buff 34)))
167             (ok true)
168     )
```

radicleart Switched data to event based - much simpler and lots of dead code can...

3fb66ba · 5 months ago

Executable File · 152 lines (130 loc) · 5.56 KB

Code Blame Raw

```
;; Title: BDE001 Snapshot Proposal Voting
;; Author: Mike Cohen (based upon work of Marvin Janssen)
;; Depends-On:
;; Synopsis:
;; This extension is a concept that allows all STX holders to
;; vote on proposals based on their STX balance.
;; Description:
;; This extension allows anyone with STX to vote on proposals. The maximum upper
;; bound, or voting power, depends on the amount of STX tokens the tx-sender
;; owned at the start block height of the proposal. The name "snapshot" comes
;; from the fact that the extension effectively uses the STX balance sheet
;; at a specific block heights to determine voting power.
;; Custom majority thresholds for voting are also possible on a per proposal basis.
;; A custom majority of 66% mean the percent of votes for must be greater than 66 for
;; the vote to carry.

(impl-trait .extension-trait.extension-trait)
(use-trait proposal-trait .proposal-trait.proposal-trait)

(define-constant err-unauthorised (err u3000))
(define-constant err-proposal-already-executed (err u3001))
(define-constant err-proposal-already-exists (err u3002))
(define-constant err-unknown-proposal (err u3003))
(define-constant err-proposal-already-concluded (err u3004))
(define-constant err-proposal-inactive (err u3005))
(define-constant err-insufficient-voting-capacity (err u3006))
(define-constant err-end-burn-height-not-reached (err u3007))
(define-constant err-not-majority (err u3008))
(define-constant err-exceeds-voting-cap (err u3009))

(define-constant custom-majority-upper u10000)
(define-constant vote-cap u140000000000)

(define-map proposals
        principal
        {
                votes-for: uint,
```

```
                votes-against: uint,
                start-height-stacks: uint,
                start-burn-height: uint,
                end-burn-height: uint,
                concluded: bool,
                passed: bool,
                custom-majority: (optional uint), ;; u10000 = 100%
                proposer: principal
        }
)

(define-map member-total-votes {proposal: principal, voter: principal} uint)

;; --- Authorisation check

(define-public (is-dao-or-extension)
        (ok (asserts! (or (is-eq tx-sender .bitcoin-dao) (contract-call? .bitcoin-dao is-exte
)

;; --- Internal DAO functions

;; Proposals

(define-public (add-proposal (proposal <proposal-trait>) (data {start-height-stacks: uint, st
        (begin
                (try! (is-dao-or-extension))
                (asserts! (is-none (contract-call? .bitcoin-dao executed-at proposal)) err-pr
                (asserts! (match (get custom-majority data) majority (> majority u5000) true)
                (print {event: "propose", proposal: proposal, proposer: tx-sender})
                (ok (asserts! (map-insert proposals (contract-of proposal) (merge {votes-for:
        )
)

;; --- Public functions

;; Proposals

(define-read-only (get-proposal-data (proposal principal))
        (map-get? proposals proposal)
)

;; Votes

(define-read-only (get-current-total-votes (proposal principal) (voter principal))
        (default-to u0 (map-get? member-total-votes {proposal: proposal, voter: voter}))
)

(define-read-only (get-historical-values (height uint) (who principal))
        (at-block (unwrap! (get-block-info? id-header-hash height) none)
                (some
                        {
                                user-balance: (stx-get-balance who),
                                voting-cap: vote-cap,
                                ;;voting-cap: (contract-call? 'SP000000000000000000002Q6VF78.
                        }
                )
```

```
94              )
95      )
96
97      (define-public (vote (amount uint) (for bool) (proposal principal))
98              (let
99                      (
100                             (proposal-data (unwrap! (map-get? proposals proposal) err-unknown-pro
101                             (new-total-votes (+ (get-current-total-votes proposal tx-sender) amou
102                             (historical-values (unwrap! (get-historical-values (get start-height-
103                     )
104                     (asserts! (>= burn-block-height (get start-burn-height proposal-data)) err-pr
105                     (asserts! (< burn-block-height (get end-burn-height proposal-data)) err-propo
106                     (asserts!
107                             (<= new-total-votes (get user-balance historical-values))
108                             err-insufficient-voting-capacity
109                     )
110                     (asserts!
111                             (< new-total-votes (get voting-cap historical-values))
112                             err-exceeds-voting-cap)
113
114                     (map-set member-total-votes {proposal: proposal, voter: tx-sender} new-total-
115                     (map-set proposals proposal
116                             (if for
117                                     (merge proposal-data {votes-for: (+ (get votes-for proposal-d
118                                     (merge proposal-data {votes-against: (+ (get votes-against pr
119                             )
120                     )
121                     (print {event: "vote", proposal: proposal, voter: tx-sender, for: for, amount
122                     (ok true)
123             )
124     )
125
126     ;; Conclusion
127
128     (define-public (conclude (proposal <proposal-trait>))
129             (let
130                     (
131                             (proposal-data (unwrap! (map-get? proposals (contract-of proposal)) e
132                             (passed
133                                     (match (get custom-majority proposal-data)
134                                             majority (> (* (get votes-for proposal-data) custom-m
135                                             (> (get votes-for proposal-data) (get votes-against p
136                                     )
137                             )
138                     )
139                     (asserts! (not (get concluded proposal-data)) err-proposal-already-concluded)
140                     (asserts! (>= burn-block-height (get end-burn-height proposal-data)) err-end-
141                     (map-set proposals (contract-of proposal) (merge proposal-data {concluded: tr
142                     (print {event: "conclude", proposal: proposal, passed: passed})
143                     (and passed (try! (contract-call? .bitcoin-dao execute proposal tx-sender)))
144                     (ok passed)
145             )
146     )
147
148     ;; --- Extension callback
149
```

```
150     (define-public (callback (sender principal) (memo (buff 34)))
151             (ok true)
152     )
```

radicleart Switched data to event based - much simpler and lots of dead code can...

3fb66ba · 5 months ago

Executable File · 154 lines (127 loc) · 5.92 KB

Code Blame Raw

```
;; Title: BDE002 Funded Custom End Proposal Submission
;; Author: Mike Cohen (based upon work of Marvin Janssen)
;; Depends-On: BDE001, BDE007
;; Synopsis:
;; This extension part of the core of Bitcoin DAO. It allows members to
;; bring proposals to the voting phase by funding them with a preset amount
;; of tokens.
;; Description:
;; The level of funding is determined by a DAO parameter and can be changed by proposal.
;; Any funder can reclaim their stx up to the point the proposal is fully funded and submitte
;; Proposals can also be marked as refundable in which case a funder can reclaim their stx
;; even after submission (during or after the voting period).
;; This extension provides the ability for the final funding transaction to set a
;; custom majority for voting. This changes the threshold from the
;; default of 50% to anything up to 100%.

(impl-trait .extension-trait.extension-trait)
(use-trait proposal-trait .proposal-trait.proposal-trait)

(define-constant err-unauthorised (err u3100))
(define-constant err-not-governance-token (err u3101))
(define-constant err-insufficient-balance (err u3102))
(define-constant err-unknown-parameter (err u3103))
(define-constant err-proposal-minimum-start-delay (err u3104))
(define-constant err-proposal-minimum-duration (err u3105))
(define-constant err-already-funded (err u3106))
(define-constant err-nothing-to-refund (err u3107))
(define-constant err-refund-not-allowed (err u3108))

(define-map refundable-proposals principal bool)
(define-map funded-proposals principal bool)
(define-map proposal-funding principal uint)
(define-map funding-per-principal {proposal: principal, funder: principal} uint)

(define-map parameters (string-ascii 30) uint)

(map-set parameters "funding-cost" u500000) ;; funding cost in uSTX. 5 STX in this case.
```

```
	(map-set parameters "minimum-proposal-start-delay" u6) ;; eg 6 = ~1 hour minimum delay before
	(map-set parameters "minimum-proposal-duration" u72) ;; eg 72 = ~1/2 days minimum duration of

	;; --- Authorisation check

	(define-public (is-dao-or-extension)
		(ok (asserts! (or (is-eq tx-sender .bitcoin-dao) (contract-call? .bitcoin-dao is-exte
	)

	;; --- Internal DAO functions

	;; Proposals

	(define-private (submit-proposal-for-vote (proposal <proposal-trait>) (start-height-stacks ui
		(contract-call? .bde001-proposal-voting add-proposal
			proposal
			{
				start-height-stacks: start-height-stacks,
				start-burn-height: start-burn-height,
				end-burn-height: (+ start-burn-height duration),
				custom-majority: custom-majority,
				proposer: tx-sender ;; change to original submitter
			}
		)
	)

	;; Parameters

	(define-public (set-parameter (parameter (string-ascii 30)) (value uint))
		(begin
			(try! (is-dao-or-extension))
			(try! (get-parameter parameter))
			(ok (map-set parameters parameter value))
		)
	)

	;; Refunds

	(define-public (set-refundable (proposal principal) (refundable bool))
		(begin
			(try! (is-dao-or-extension))
			(ok (map-set refundable-proposals proposal refundable))
		)
	)

	;; --- Public functions

	;; Parameters

	(define-read-only (get-parameter (parameter (string-ascii 30)))
		(ok (unwrap! (map-get? parameters parameter) err-unknown-parameter))
	)

	;; Funding status

	(define-read-only (is-proposal-funded (proposal principal))
```

```
94              (default-to false (map-get? funded-proposals proposal))
95      )
96
97      (define-read-only (get-proposal-funding (proposal principal))
98              (default-to u0 (map-get? proposal-funding proposal))
99      )
100
101     (define-read-only (get-proposal-funding-by-principal (proposal principal) (funder principal))
102             (default-to u0 (map-get? funding-per-principal {proposal: proposal, funder: funder}))
103     )
104
105     (define-read-only (can-refund (proposal principal) (funder principal))
106             (or
107                     (default-to false (map-get? refundable-proposals proposal))
108                     (and (not (is-proposal-funded proposal)) (is-eq funder tx-sender))
109             )
110     )
111
112     ;; Proposals
113
114     (define-public (fund (proposal <proposal-trait>) (start-delay uint) (duration uint) (amount u
115             (let
116                     (
117                             (proposal-principal (contract-of proposal))
118                             (current-total-funding (get-proposal-funding proposal-principal))
119                             (funding-cost (try! (get-parameter "funding-cost")))
120                             (difference (if (> funding-cost current-total-funding) (- funding-cos
121                             (funded (<= difference amount))
122                             (transfer-amount (if funded difference amount))
123                     )
124                     (asserts! (not (is-proposal-funded proposal-principal)) err-already-funded)
125                     (and (> transfer-amount u0) (try! (stx-transfer? transfer-amount tx-sender .b
126                     (map-set funding-per-principal {proposal: proposal-principal, funder: tx-send
127                     (map-set proposal-funding proposal-principal (+ current-total-funding transfe
128                     (asserts! funded (ok false))
129                     (asserts! (>= start-delay (try! (get-parameter "minimum-proposal-start-delay"
130                     (asserts! (>= duration (try! (get-parameter "minimum-proposal-duration"))) er
131                     (map-set funded-proposals proposal-principal true)
132                     (submit-proposal-for-vote proposal block-height (+ burn-block-height start-de
133             )
134     )
135
136     (define-public (refund (proposal principal) (funder (optional principal)))
137             (let
138                     (
139                             (recipient (default-to tx-sender funder))
140                             (refund-amount (get-proposal-funding-by-principal proposal recipient)
141                     )
142                     (asserts! (> refund-amount u0) err-nothing-to-refund)
143                     (asserts! (can-refund proposal recipient) err-refund-not-allowed)
144                     (map-set funding-per-principal {proposal: proposal, funder: recipient} u0)
145                     (map-set proposal-funding proposal (- (get-proposal-funding proposal) refund-
146                     (contract-call? .bde006-treasury stx-transfer refund-amount recipient none)
147             )
148     )
149
```

```
150     ;; --- Extension callback
151
152     (define-public (callback (sender principal) (memo (buff 34)))
153             (ok true)
154     )
```

radicleart Clarigne tests updated 6be7546 · 5 months ago

Executable File · 79 lines (65 loc) · 2.81 KB

Code Blame Raw

```
;; Title: BDE003 Core Proposals
;; Author: Mike Cohen (based upon work of Marvin Janssen)
;; Depends-On: BDE001
;; Synopsis:
;; This extension allows for the creation of core proposals by a few trusted
;; principals.
;; Description:
;; Only a list of trusted principals, designated as the
;; "core team", can create core proposals. The core proposal
;; extension has an optional ~3 month sunset period, after which no more core
;; proposals can be made - set it to 0 to disable. The core team members, sunset period, and
;; core vote duration can be changed by means of a future proposal.

(impl-trait .extension-trait.extension-trait)
(use-trait proposal-trait .proposal-trait.proposal-trait)

(define-data-var core-team-sunset-height uint u0) ;; does not expire by default - can be char

(define-constant err-unauthorised (err u3300))
(define-constant err-not-core-team-member (err u3301))
(define-constant err-sunset-height-reached (err u3302))
(define-constant err-sunset-height-in-past (err u3303))
(define-constant err-proposal-minimum-start-delay (err u3304))
(define-constant err-proposal-minimum-duration (err u3305))

(define-map core-team principal bool)

;; --- Authorisation check

(define-public (is-dao-or-extension)
        (ok (asserts! (or (is-eq tx-sender .bitcoin-dao) (contract-call? .bitcoin-dao is-exte
)

;; --- Internal DAO functions

(define-public (set-core-team-sunset-height (height uint))
        (begin
                (try! (is-dao-or-extension))
                (asserts! (> height burn-block-height) err-sunset-height-in-past)
```

```
                (ok (var-set core-team-sunset-height height))
        )
)

(define-public (set-core-team-member (who principal) (member bool))
        (begin
                (try! (is-dao-or-extension))
                (ok (map-set core-team who member))
        )
)

;; --- Public functions

(define-read-only (is-core-team-member (who principal))
        (default-to false (map-get? core-team who))
)

(define-public (core-propose (proposal <proposal-trait>) (start-burn-height uint) (duration u
        (begin
                (asserts! (is-core-team-member tx-sender) err-not-core-team-member)
                (asserts! (or (is-eq (var-get core-team-sunset-height) u0) (< burn-block-heig
                (asserts! (>= start-burn-height (+ burn-block-height u2)) err-proposal-minimu
                (asserts! (>= (+ start-burn-height duration) (+ burn-block-height u72)) err-p
                (contract-call? .bde001-proposal-voting add-proposal proposal
                        {
                                start-height-stacks: block-height,
                                start-burn-height: start-burn-height,
                                end-burn-height: (+ start-burn-height duration),
                                custom-majority: custom-majority,
                                proposer: tx-sender ;; change to original submitter
                        }
                )
        )
)

;; --- Extension callback

(define-public (callback (sender principal) (memo (buff 34)))
        (ok true)
)
```

radicleart Switched data to event based - much simpler and lots of dead code can...

3fb66ba · 5 months ago

Executable File · 101 lines (82 loc) · 3.6 KB

Code Blame Raw

```
;; Title: BDE004 Core Execute
;; Author: Mike Cohen (based on Marvin Janssen)
;; Depends-On:
;; Synopsis:
;; This extension allows a small number of very trusted principals to immediately
;; execute a proposal once a super majority is reached.
;; Description:
;; An extension meant for the bootstrapping period of a DAO. It temporarily gives
;; some very trusted principals the ability to perform an "executive action";
;; meaning, they can skip the voting process to immediately executive a proposal.
;; The Core execute extension has an optional sunset period of ~1 month from deploy
;; time, set it to 0 to disable. The core executive team, parameters, and sunset period may b
;; by means of a future proposal.

(impl-trait .extension-trait.extension-trait)
(use-trait proposal-trait .proposal-trait.proposal-trait)

(define-data-var executive-team-sunset-height uint u0) ;; does not expire by default - can be

(define-constant err-unauthorised (err u3400))
(define-constant err-not-executive-team-member (err u3401))
(define-constant err-already-executed (err u3402))
(define-constant err-sunset-height-reached (err u3403))
(define-constant err-sunset-height-in-past (err u3404))

(define-map executive-team principal bool)
(define-map executive-action-signals {proposal: principal, team-member: principal} bool)
(define-map executive-action-signal-count principal uint)

(define-data-var executive-signals-required uint u1) ;; signals required for an executive act

;; --- Authorisation check

(define-public (is-dao-or-extension)
        (ok (asserts! (or (is-eq tx-sender .bitcoin-dao) (contract-call? .bitcoin-dao is-exte
)

```

```
;; --- Internal DAO functions

(define-public (set-executive-team-sunset-height (height uint))
	(begin
		(try! (is-dao-or-extension))
		(asserts! (> height burn-block-height) err-sunset-height-in-past)
		(ok (var-set executive-team-sunset-height height))
	)
)

(define-public (set-executive-team-member (who principal) (member bool))
	(begin
		(try! (is-dao-or-extension))
		(ok (map-set executive-team who member))
	)
)

(define-public (set-signals-required (new-requirement uint))
	(begin
		(try! (is-dao-or-extension))
		(ok (var-set executive-signals-required new-requirement))
	)
)

;; --- Public functions

(define-read-only (is-executive-team-member (who principal))
	(default-to false (map-get? executive-team who))
)

(define-read-only (has-signalled (proposal principal) (who principal))
	(default-to false (map-get? executive-action-signals {proposal: proposal, team-member
)

(define-read-only (get-signals-required)
	(var-get executive-signals-required)
)

(define-read-only (get-signals (proposal principal))
	(default-to u0 (map-get? executive-action-signal-count proposal))
)

(define-public (executive-action (proposal <proposal-trait>))
	(let
		(
			(proposal-principal (contract-of proposal))
			(signals (+ (get-signals proposal-principal) (if (has-signalled propo
		)
		(asserts! (is-executive-team-member tx-sender) err-not-executive-team-member)
		(asserts! (or (is-eq (var-get executive-team-sunset-height) u0) (< burn-block
		(and (>= signals (var-get executive-signals-required))
			(try! (contract-call? .bitcoin-dao execute proposal tx-sender))
		)
		(map-set executive-action-signals {proposal: proposal-principal, team-member:
		(map-set executive-action-signal-count proposal-principal signals)
		(ok signals)
```

```
        )
    )

    ;; --- Extension callback

    (define-public (callback (sender principal) (memo (buff 34)))
        (ok true)
    )
```

radicleart Switched data to event based - much simpler and lots of dead code can...

3fb66ba · 5 months ago

Executable File · 157 lines (128 loc) · 4.95 KB

Code Blame Raw

```
;; Title: EDE006 Treasury
;; Author: Mike Cohen (based upon work of Marvin Janssen)
;; Depends-On:
;; Synopsis:
;; A treasury that can manage STX, SIP009, SIP010, and SIP013 tokens.
;; Description:
;; An extension contract that is meant to hold tokens on behalf of the
;; DAO. It can hold and transfer STX, SIP009, SIP010, and SIP013 tokens.
;; They can be deposited by simply transferring them to the contract.
;; Any extension or executing proposal can trigger transfers.
;; Technically, the ExecutorDAO core can hold and transfer tokens
;; directly. The treasury extension merely adds a bit of separation.

(impl-trait .extension-trait.extension-trait)

(define-constant err-unauthorised (err u3000))

;; --- Transferable traits

(define-trait sip009-transferable
	(
		(transfer (uint principal principal) (response bool uint))
	)
)

(define-trait sip010-transferable
	(
		(transfer (uint principal principal (optional (buff 34))) (response bool uint
	)
)

(define-trait sip013-transferable
	(
		(transfer (uint uint principal principal) (response bool uint))
		(transfer-memo (uint uint principal principal (buff 34)) (response bool uint)
	)
)
```

```
38
39      (define-trait sip013-transferable-many
40              (
41                      (transfer-many ((list 200 {token-id: uint, amount: uint, sender: principal, r
42                      (transfer-many-memo ((list 200 {token-id: uint, amount: uint, sender: princip
43              )
44      )
45
46      ;; --- Authorisation check
47
48      (define-public (is-dao-or-extension)
49              (ok (asserts! (or (is-eq tx-sender .bitcoin-dao) (contract-call? .bitcoin-dao is-exte
50      )
51
52      ;; --- Internal DAO functions
53
54      ;; STX
55
56      (define-public (stx-transfer (amount uint) (recipient principal) (memo (optional (buff 34))))
57              (begin
58                      (try! (is-dao-or-extension))
59                      (match memo to-print (print to-print) 0x)
60                      (as-contract (stx-transfer? amount tx-sender recipient))
61              )
62      )
63
64      (define-public (stx-transfer-many (transfers (list 200 {amount: uint, recipient: principal, m
65              (begin
66                      (try! (is-dao-or-extension))
67                      (as-contract (fold stx-transfer-many-iter transfers (ok true)))
68              )
69      )
70
71      ;; SIP009
72
73      (define-public (sip009-transfer (token-id uint) (recipient principal) (asset <sip009-transfer
74              (begin
75                      (try! (is-dao-or-extension))
76                      (as-contract (contract-call? asset transfer token-id tx-sender recipient))
77              )
78      )
79
80      (define-public (sip009-transfer-many (data (list 200 {token-id: uint, recipient: principal})))
81              (begin
82                      (as-contract (fold sip009-transfer-many-iter data asset))
83                      (ok true)
84              )
85      )
86
87      ;; SIP010
88
89      (define-public (sip010-transfer (amount uint) (recipient principal) (memo (optional (buff 34)
90              (begin
91                      (try! (is-dao-or-extension))
92                      (as-contract (contract-call? asset transfer amount tx-sender recipient memo))
93              )
```

```
94      )
95
96      (define-public (sip010-transfer-many (data (list 200 {amount: uint, recipient: principal, mem
97              (begin
98                      (as-contract (fold sip010-transfer-many-iter data asset))
99                      (ok true)
100             )
101     )
102
103     ;; SIP013
104
105     (define-public (sip013-transfer (token-id uint) (amount uint) (recipient principal) (memo (op
106             (begin
107                     (try! (is-dao-or-extension))
108                     (as-contract (match memo memo-buff
109                             (contract-call? asset transfer-memo token-id amount tx-sender recipie
110                             (contract-call? asset transfer token-id amount tx-sender recipient)
111                     ))
112             )
113     )
114
115     (define-public (sip013-transfer-many (transfers (list 200 {token-id: uint, amount: uint, senc
116             (begin
117                     (try! (is-dao-or-extension))
118                     (as-contract (contract-call? asset transfer-many transfers))
119             )
120     )
121
122     (define-public (sip013-transfer-many-memo (transfers (list 200 {token-id: uint, amount: uint,
123             (begin
124                     (try! (is-dao-or-extension))
125                     (as-contract (contract-call? asset transfer-many-memo transfers))
126             )
127     )
128
129     ;; --- Iterator functions
130
131     (define-private (stx-transfer-many-iter (data {amount: uint, recipient: principal, memo: (opt
132             (begin
133                     (try! previous-result)
134                     (match (get memo data) to-print (print to-print) 0x)
135                     (stx-transfer? (get amount data) tx-sender (get recipient data))
136             )
137     )
138
139     (define-private (sip009-transfer-many-iter (data {token-id: uint, recipient: principal}) (ass
140             (begin
141                     (unwrap-panic (contract-call? asset transfer (get token-id data) tx-sender (g
142                     asset
143             )
144     )
145
146     (define-private (sip010-transfer-many-iter (data {amount: uint, recipient: principal, memo: (
147             (begin
148                     (unwrap-panic (contract-call? asset transfer (get amount data) tx-sender (get
149                     asset
```

```
        )
    )

    ;; --- Extension callback

    (define-public (callback (sender principal) (memo (buff 34)))
        (ok true)
    )
```