DRAFT: A textbook time-locked Clarity smart contract

Part 8

Phillip G. Bradford*

November 29, 2024

Abstract

These notes discuss a time-locked Clarity smart contract. This Clarity contract is directly from the book: *Clarity of Mind* [2].

1 Motivation

The text-book example is from the Clarity of Mind book [2].

Listing 1: Errors directly from the Clarity of Mind book

```
;; Owner
(define-constant contract-owner tx-sender)

;; Errors
(define-constant err-owner-only (err u100))
(define-constant err-already-locked (err u101))
(define-constant err-unlock-in-past (err u102))
(define-constant err-no-value (err u103))
(define-constant err-beneficiary-only (err u104))
(define-constant err-unlock-height-not-reached (err u105))

;; Data
(define-data-var beneficiary (optional principal) none)
(define-data-var unlock-height uint u0)
```

*phillip.bradford@uconn.edu, phillip.g.bradford@gmail.com, University of Connecticut, Department of Computer Science and Engineering, Connecticut, USA

Listing 2 sets the beneficiary, the block-height for the relase of the amount. The block-height is the Stacks block number. In Clarity, the block-height is in the variable `block-height`.

Listing 2: The lock function from the Clarity of Mind book

```
(define-public (lock (new-beneficiary principal)
                     (unlock-at uint)
                     (amount uint))
    (begin
        (asserts! (is-eq tx-sender contract-owner)
           err-owner-only)
        (asserts! (is-none (var-get beneficiary))
           err-already-locked)
        (asserts! (> unlock-at block-height)
           err-unlock-in-past)
        (asserts! (> amount u0) err-no-value)
        (try! (stx-transfer? amount tx-sender
                             (as-contract tx-sender)))
        (var-set beneficiary (some new-beneficiary))
        (var-set unlock-height unlock-at)
        (ok true)
    )
)
```

Listing 2 it is worth noting that,

```
(stx-transfer? amount
                            tx-sender
               (as-contract tx-sender))
```

transfers the amount from `tx-sender` to `as-contract tx-sender`) to (1) test that there is sufficient STX to transfer and (2) the stx-transfer is done as a contract principal rather than the user's principal.

Listing 3 shows how ownership of the to-be transferred can be allocated to another principal.

Listing 3: The bestow function from the Clarity of Mind book

```
(define-public (bestow (new-beneficiary principal))
    (begin
        (asserts! (is-eq (some tx-sender)
                         (var-get beneficiary))
                  err-beneficiary-only)
        (var-set beneficiary (some new-beneficiary))
        (ok true)
    )
)
```

Listing 4 shows how a claim can be executed. Note this contract checks the beneficiary is `tx-sender` and the `block-height` is sufficient.

Listing 4: The claim function from the Clarity of Mind book

```
(define-public (claim)
    (begin
        (asserts! (is-eq (some tx-sender)
                         (var-get beneficiary))
                         err-beneficiary-only)
        (asserts! (>= block-height (var-get unlock-height))
                         err-unlock-height-not-reached)
        (as-contract (stx-transfer? (stx-get-balance tx-sender)
                         tx-sender
                         (unwrap-panic (var-get beneficiary))))
    )
)
```

2 Real time in Stacks blocks

https://docs.hiro.so/stacks/clarity/functions/get-block-info

3 Commands

```
windows>> clarinet console
>> ::get_assets_maps
>> (contract-call? .timelocked-wallet lock
   'ST1SJ3DTE5DN7X54YDH5D64R3BCB6A2AG2ZQ8YPD5 u10 u100)
>> ::set_tx_sender ST1SJ3DTE5DN7X54YDH5D64R3BCB6A2AG2ZQ8YPD5
>>  (contract-call?
   'ST1PQHQKV0RJXZFY1DGX8MNSNYVE3VGZJSRTPGZGM.timelocked-wallet
   claim)
(err u105)
>> ::advance_chain_tip 10
10 blocks simulated, new height: 11
>> (contract-call?
   'ST1PQHQKV0RJXZFY1DGX8MNSNYVE3VGZJSRTPGZGM.timelocked-wallet
   claim)
>> ::get_assets_maps
```

4 Exercise

1. How can we
2.

References

[1] https://docs.stacks.co/docs/cookbook/creating-an-ft

[2] Clarity of Mind Book, https://book.clarity-lang.org/ 2023-04-11.

[3] Harold Abelson, Gerald Jay Sussman, with Julie Sussman: *Structure and Interpretation of Computer Programs*, Second Edition, MIT Press, 1996.

[4] Daniel P. Friedman and Matthias Felleisen: *The Little Schemer*, Fourth edition, MIT Press, 1996.

[5] Kenny Rogers: *Building an NFT with Stacks and Clarity*, https://blog.developerdao.com/building-an-nft-with-stacks-and-clarity, 2022-09-01.

[6] Kenny Roger, Joe Bender: Stacks developer workshop: *Web3 for Bitcoin: The What, Why, and How of Building on Stacks*. Web3 for Bitcoin. Wed, Jun 29, 2022.