Escrow Worker Management

Bitfest 2018 - Amsterdam

Dr.-Ing. Fabian Schuh

Blockchain Projects B.V.

September 22, 2018

BlockchainProjects

The bitUSD worker concept

The concept - today

1. *Escrow service* **sets up** worker
2. *Escrow service* **obtains BTS** from reserves
3. *Escrow service* **obtains bitUSD**
4. *freelancer* **delivers**
5. *freelancer* **receives** bitUSD
6. *Escrow service* **returns** excess BTS to reserves

The concept - today

1. *Escrow service* **sets up** worker
2. *Escrow service* **obtains BTS** from reserves
3. *Escrow service* **obtains bitUSD**
4. *freelancer* **delivers**
5. *freelancer* **receives** bitUSD
6. *Escrow service* **returns** excess BTS to reserves

- First mentioned **December 14, 2016** @ bitsharestalk
- multi-sig group (fox, cass, sigve, blocktrades, chainsquad)
- Burned 6 401 109.278 34 BTS and 2349.6014 USD
- Scripted Proposals (`https://github.com/xeroc/worker-proposals`)

```
print("Burning BTS")
bitshares.reserve(
    Amount(6401109.50898 - fees, "BTS"),
    account=account
)
```

Operation: `1.11.28772311`

```
print("Send USD to committee-account")
bitshares.transfer(
    "committee-account",
    2349.6014, "USD",
    account=account)
```

Operation: `1.11.28772312`

bitUSD workers

The Pros:

- More **volume** in BTS markets
- More **security** for BTS voters
- **Calculable** expenses to reserves (no currency risk)
- Significantly less **currency risk** to freelancer
- High **transparency** (all on-chain)
- **Enables** accounting audits

The Cons:

- Higher **barrier** of entry
- Higher **expense** due to escrow fee
- *(Requires **trust** in escrow service)*

BitShares Blockchain Foundation

Design Decisions

- public accounting journal (github)

[1] except where otherwise stated - for instance to fund a faucet with BTS

Design Decisions

- public accounting journal (github)
- worker-specific accounting

[1] except where otherwise stated - for instance to fund a faucet with BTS

Design Decisions

- public accounting journal (github)
- worker-specific accounting
- *taker*-role only (so far)

[1] except where otherwise stated - for instance to fund a faucet with BTS

Design Decisions

- public accounting journal (github)
- worker-specific accounting
- *taker*-role only (so far)
- no call orders (so far)

[1] except where otherwise stated - for instance to fund a faucet with BTS

Design Decisions

- public accounting journal (github)
- worker-specific accounting
- *taker*-role only (so far)
- no call orders (so far)
- **only bitassets** leave the workers account[1]

[1] except where otherwise stated - for instance to fund a faucet with BTS

Design Decisions

- public accounting journal (github)
- worker-specific accounting
- *taker*-role only (so far)
- no call orders (so far)
- **only bitassets** leave the workers account[1]
- funds leaving workers account **only go to escrow** account

[1] except where otherwise stated - for instance to fund a faucet with BTS

Design Decisions

- public accounting journal (github)
- worker-specific accounting
- *taker*-role only (so far)
- no call orders (so far)
- **only bitassets** leave the workers account[1]
- funds leaving workers account **only go to escrow** account
- two separate accounts

[1] except where otherwise stated - for instance to fund a faucet with BTS

Design Decisions

- public accounting journal (github)
- worker-specific accounting
- *taker*-role only (so far)
- no call orders (so far)
- **only bitassets** leave the workers account[1]
- funds leaving workers account **only go to escrow** account
- two separate accounts
  - `workers.bitshares.foundation`
    - fund consolidation into single account
    - committee owned
    - funds owned by community/reserves
    - controlled by BBF (active key-only)
    - creates and manages the worker objects

[1] except where otherwise stated - for instance to fund a faucet with BTS

Design Decisions

- public accounting journal (github)
- worker-specific accounting
- *taker*-role only (so far)
- no call orders (so far)
- **only bitassets** leave the workers account[1]
- funds leaving workers account **only go to escrow** account
- two separate accounts
  - `workers.bitshares.foundation`
    - fund consolidation into single account
    - committee owned
    - funds owned by community/reserves
    - controlled by BBF (active key-only)
    - creates and manages the worker objects
  - `bitshares.foundation`
    - foundation owned (escrow)
    - separation of concerns
    - mere control of escrow funds

[1] except where otherwise stated - for instance to fund a faucet with BTS

From BBF POV

Tasks of the BBF

1. **setup** worker object on-chain
2. *claim* BTS from reserves
3. **obtain** bitUSD
4. **transfer** bitUSD to escrow account
5. **review** deliveries
6. **release** funds to freelancer
7. return/**burn** excess BTS
8. **accounting & reporting**
9. *maintain* transparency
10. *educate* and *consult*

BitShares Blockchain Foundation

→ Setup

Setup

After coming to agreement with freelancer, the BBF sets up a worker object on chain with public data about:

- ✓ **name** (on-chain)
- ✓ **total payment** (in USD)
- ✓ **price of currency** (BTS per 1 USD)
- ✓ **backoff factor** (usually 2x)
- ✓ **start-/end- date** (on-chain)
- ✓ **url** (on-chain)
- ✓ **payment-account** (freelancer account name)
- ✓ *daily pay* (in BTS, derived)

BitShares Blockchain Foundation

→ Maintenance

Maintenance

After creation of the worker, a cycle begins:

1. claim available BTS from worker
2. try to buy “some” bitUSD
3. transfer bitUSD to escrow
4. go back to 1

Maintenance - Details

What sounds easy in theory becomes more complicated when **requiring transparency and accountability**:

Claiming

1. claim from the worker object on-chain
2. account for additional BTS for worker in in accounting ledger

```
2017/12/15 * Claiming for 201712-infrastructure
    workers.bitshares.foundation:201712-infrastructure:BTS        21937.5 BTS
    reserves:201712-infrastructure        -21937.5 BTS
    workers.bitshares.foundation:201712-infrastructure:BTS        -24.27935 BTS @ 0.3285 USD
    Transactionfee        (24.27935 * 0.3285 USD)
```

Maintenance - Details

What sounds easy in theory becomes more complicated when **requiring transparency and accountability**:

Buying

1. place a **fill-or-kill** order
2. buy from the market
3. wait one block (so order fills)
4. account for each individual trade
5. this way, each bitUSD in the workers account can be associated to a specific worker!

```
2018/01/05 * Create Order for 201712-infrastructure into USD @ 1.278653767 BTS/USD (order: 1.7.45755528)
    ; 201712-infrastructure        -20000.0 BTS @ 1.278653767 BTS/USD
    ; 201712-infrastructure        15641.45081 USD
    workers.bitshares.foundation:201712-infrastructure:BTS        -0.01213 BTS @ 0.8603 USD
    Transactionfee        (0.01213 * 0.8603 USD)


2018/01/05 * Executed Trade 1.7.45755528 for 201712-infrastructure
    workers.bitshares.foundation:201712-infrastructure:BTS      -3,466.32482 BTS
    workers.bitshares.foundation:201712-infrastructure:USD      2,932.9677 USD @@ 3,466.32482 BTS

2018/01/05 * Executed Trade 1.7.45755528 for 201712-infrastructure
    workers.bitshares.foundation:201712-infrastructure:BTS      -2,360.37312 BTS
    workers.bitshares.foundation:201712-infrastructure:USD      1,999.0000 USD @@ 2,360.37312 BTS

...
```

Maintenance - Details

What sounds easy in theory becomes more complicated when **requiring transparency and accountability**:

Transfer

1. Initiate transfer to escrow account
2. account for transferred bitUSD of the specified worker

```
2018/01/09 * Transfering to bitshares.foundation 201712-infrastructure
    workers.bitshares.foundation:201712-infrastructure:USD        -19000.0 USD
    bitshares.foundation:201712-infrastructure:USD        19000.0 USD
    workers.bitshares.foundation:201712-infrastructure:BTS        -0.01759 BTS @ 0.7641 USD
    Transactionfee        (0.01759 * 0.7641 USD)
```

BitShares Blockchain Foundation

→ Payout

Payout

Upon reaching milestones the freelancer requests payment in bitUSD.

1. Request

Request

- receive payment request from freelancer via Email
- extract request file (pdf, xlsx, …)
- upload file for transparency

Payout

Upon reaching milestones the freelancer requests payment in bitUSD.

1. Request
2. Review

Review

- … amounts and total
- … milestone items
- … delivery (code, …)

Payout

Upon reaching milestones the freelancer requests payment in bitUSD.

1. Request
2. Review
3. Payout

Payout

- approving payout results in transfer

```
2018/02/16 * Paying out to blockchainprojects 201712-infrastructure
    bitshares.foundation:201712-infrastructure:USD        -7615.65 USD
    outstanding:201712-infrastructure:USD:blockchainprojects        7615.65 USD
    bitshares.foundation        -0.01759 BTS @ 0.2591 USD
    Transactionfee        (0.01759 * 0.2591 USD)
```

BitShares Blockchain Foundation

→ Implementation

Implemenation Details

Stack:

Structure:

- Object-Relational Mapping (ORM)
  - integrates with MySQL (management, caching)
  - synchronizes with BitShares Blockchain (read)
  - interacts with BitShares Blockchain (transact)
    - integrated worker-specific accounting with `ledger`
    - tracking of individual order-matching
    - accounting for proper trading fee
    - worker creation management
    - burn management
- Flask Interface:
  - Bower, SemanticUI, WtForms, jQuery, Jinja2, Jade, …
  - user Management
  - application programming interface (API, Swagger)
  - detailed accounting reports
  - multiple progress reports (time, budget, funding, obtaining,…)

Limitations

Limitation

- No support for call orders (so far)
- Requires market depth
- Pays a premium to market participants to obtain bitUSD
- No full automation of claim+trade+escrow cycle (yet)

Roadmap

Roadmap

- **bitCNY** support (almost done)
- **Automate** buy cycle
- Improved status **reporting**
- Make use of **proposals** for escrow (accounting)
- Improved security (**BSIP40**) + multi-sig
- Support *maker* role for trades
- Support **call orders**

Live Demo

Live

https://workers.bitshares.foundation

Questions?

Thank you for your interest!