RefToken:
A decentralized affiliate marketplace

Manuel Granados, Alexander Anter
&
Jan Sammut

Version 1.0

Abstract

The architecture and protocol details for developing the infrastructure that supports a decentralized marketplace on the blockchain for merchants and affiliates.

Contents

1. Introduction

As a markets become saturated and conventional forms of marketing cease to be profitable, companies traditionally turn to affiliates to increase their client acquisition performance. Affiliate programs are generally run off third party platforms in a bid to provide transparency and impartiality, these however suffer from a number of trust-based flaws.

Mismatches and disagreements between reporting numbers due to tracking issues and fraudulent behavior frequently occur, these later become a source of conflict between participants in the deal. Attempts at utilizing third party data sources have been made, however these have a high cost associated with them, and remain limited by the origin of the data in question, which remains the merchant's BI platform. Our solution disintermediates the data flow, leveraging the blockchain's immutability as an impartial source of sales and performance data.

RefToken has chosen to build its platform on the Ethereum protocol[1] as this emerging blockchain provides all the characteristics needed for our system: robustness, transparency and automation.

The Ethereum platform is currently thriving due to the paradigm shift its protocol presents. This has led to a vast amount of applications being built on top of the network, referred to as "Dapps". Having RefToken built with an ERC-20 token built upon Ethereum will make it easier to interact with other applications sharing the same blockchain.

2. Overview

This yellow paper contains our implantation methodology. First off we will introduce the actors involved in the whole system. Then, we'll cover our technological base, including blockchain, wallets, tokens, contracts. Related to that, we'll define what data are we going to require and how is it planned to be stored for further usage. Finally, we will give an overview of future plans.

Throughout this document we will discuss a specific use-case of using our platform for an ICO. This is just an example and one of many use-cases our platform can be used for.

3. Actors

There are three main actors in our system: Merchants, Affiliates and Clients.

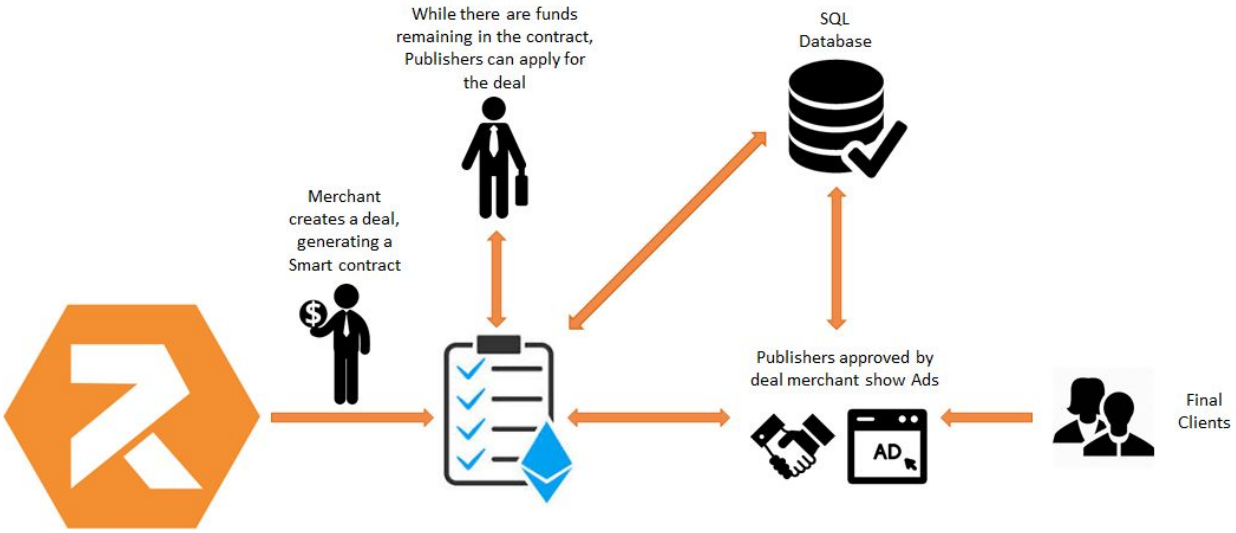

Fig. 3.1: RefToken architecture diagram

3.1. Merchants

We define “Merchants” as those companies who are looking to attract “Clients” to take part of their business.

What can a Merchant do?:

- Register on the RefToken platform.
- Create Brands
- Create new deals associated with a specific brand
- Approve Affiliate applications to a specific deal.
- Leave reviews on an affiliate’s account
- Contact Affiliates
- Upload creatives associated with a specific brand/deal.
- Generate performance reports
- Modify current active deals.

3.2. Affiliates

The term Affiliate includes all the actors, influencers, website-owners etc. that are able to provide marketing value and generate profitable traffic for the merchant.

What can an Affiliate do?:

- Register on the RefToken platform.
- Apply to participate in any deal.
- Ask a deal owner (Merchant) for a custom deal (e.g. negotiating on fees, conditions)
- Leave reviews on Merchants accounts
- Contact Merchants
- On approval, get Ads to be published.
- Get rewarded when fulfilling Deals by delivering Clients to Merchants.
- Generate performance reports

3.3. Clients

The third actor in this system is the final client. Clients visit merchants to take part in their business. In the case of a client referred by an Affiliate, the visit will be tracked, first through cookies and then registered in our separate off-chain DB upon registration.

What can a “final client” do?:

- Register in the Merchant website using the RefToken Module.
- Make use of the Merchant Dapp by depositing to the Merchant through their own wallet.
- Generate a reward for the corresponding affiliate.

4. Wallets and Tokens usage

4.1. Token types and usage

There are four different tokens involved in our system, each one with its own purpose.

4.1.1. Ether (ETH)

We accept ETH in our REF contract to buy/sell REF. ETH transactions to Merchants gets registered in our DB together with associated Affiliate ID.

4.1.2. RefToken (REF)

REF is an ERC-20 token[2] defined by ourselves. It is the primary token involved in all the transactions throughout the system. The ERC-20 standard already includes balance getters and transfer functions, so this can be used to manage transactions between our actors. When creating a Deal a Merchant needs to lock up a certain amount of REF in escrow, the Merchant can then choose to either base the payout on the value of COMM-tokens(EURO) or REF-tokens.

REF-tokens will also be used to reward active Merchant and Affiliates on the platform when they:

- Validate their identity
- Validate website domains through a DNS record
- Create deals
- Get accepted to deals
- Fulfill deals
- Pay out rewards
- Leave reviews
- Refer someone that either fulfills a deal or creates one that gets fulfilled.

4.1.3. Commission Token (COMM)

The COMM-token is defined as the REF token to EUR conversion at the moment of a deal generation. It only exists within the deal smart-contract and will automatically be exchanged back to REF after the deal is over.

The COMM-token will be pegged to the Euro in order to offer stability to the deals, keeping the costs under control for merchant's protection.

4.1.4. ICO-Token

For an ICO use-case we call all new tokens brands generate for their project, ICO-tokens. When a client contributes to the ICO, he sends ETH and receives ICO-tokens back in his ETH wallet.

4.2. Token flows

There are several types of conversions between tokens depending on each actor point of view. In following subsections we analyze each of them. See the whole picture in Fig. 4.1.

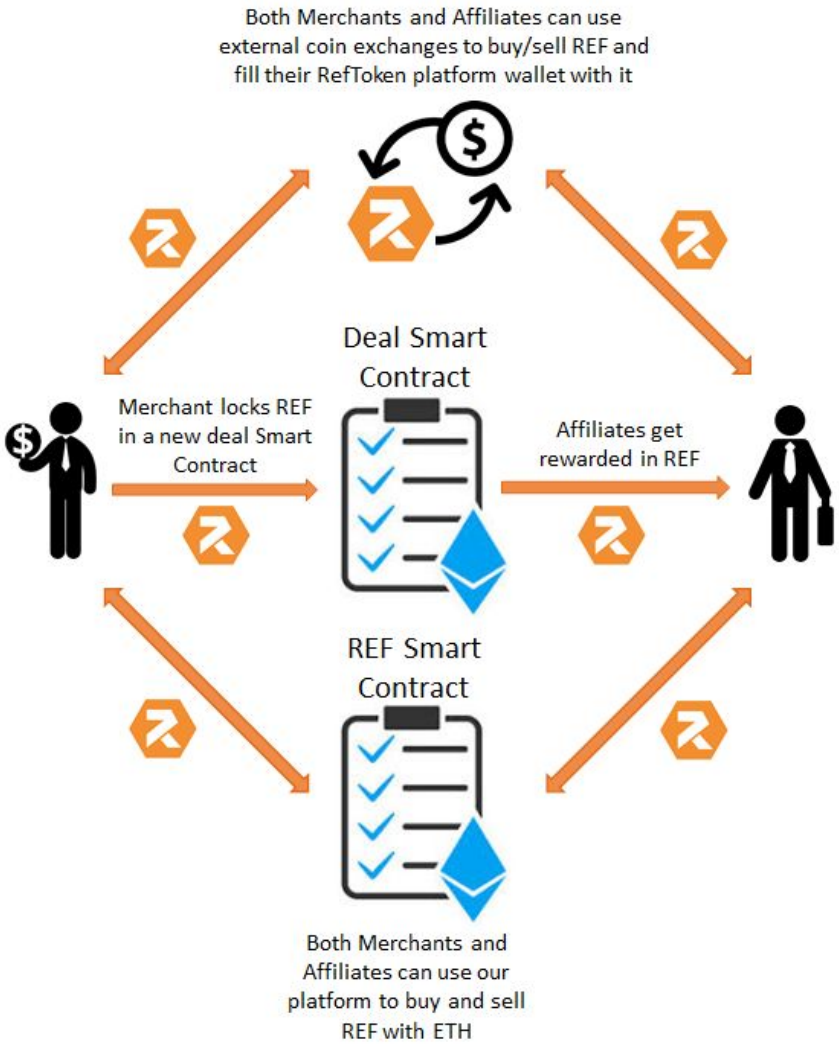

Fig. 4.1: Token flow within RefToken

4.2.1. Token flow for the Client

Client creates an account on the Merchant website. Following which the client gets presented with an ETH-address to contribute to, so he can purchase the Merchant tokens, be tracked for deposits, etc.

In the specific case of an ICO deal, when a transfer is made to that specific ETH-address, the smart contract automatically releases ICO-token to the same ETH-address that was making the transfer, the client's own wallet, so no further addresses are required.

4.2.2. Token flow for the Merchant

Merchant creates a REF-wallet account on the RefToken-platform and either buys REF through our wallet with ETH or transfers REF bought at an exchange to their wallet.

After the Merchant deposits REF in their wallet, it's now possible to create a deal by locking up a specific amount of REF in escrow that will be released under specific terms. There is also a possibility to generate the reward in a Commission-token, "COMM" that is pegged to the Euro to protect the Merchant from volatility in the price of REF.

4.2.3. Token flow for the Affiliate

Affiliates create an account and apply for a deal, when the conversion requirements have been fulfilled the deal's smart contract will automatically release REF funds into the affiliate's wallet. The affiliate can directly exchange these for ETH on the platform or transfer them to another wallet.

5. Contracts

According to Ethereum White Paper[1], Smart Contracts are “cryptographic boxes that contain value and only unlock it if certain conditions are met, can also be built on top of the platform, with vastly more power than that offered by Bitcoin scripting because of the added powers of Turing-completeness, value-awareness, blockchain-awareness and state.”

We take advantage of the Smart Contracts in order to implement the logic behind our automatically triggered transactions, deal implementation on blockchain and the whole system trustworthiness and transparency.

More detail of our contracts in following subsections.

5.1. RefToken definition contract

Our ERC-20 standard compliant token generation contract, gives us everything we need to integrate it within our platform.

5.2. RefToken exchange contract

The RefToken exchange contract allows ETH to REF and vice versa conversion to merchants and affiliates. It will also contain COMM to REF and vice versa conversion for our internal usage.

5.2.1. ETH <=> REF conversion

By implementing an API from an exchange protocol we will be able to convert ETH to REF and vice versa with ease and realtime exchange rates.

5.2.2. REF <=> COMM conversion

When a Merchant choose to offer a reward in COMM instead of REF, a specific amount of REF gets converted into COMM depending on the current EUR value we collect from the exchanges through their APIs. The COMM-tokens are then locked up until the Deal is closed, the COMM-tokens that remain will be converted back to REF, again collecting the current value from exchanges through their APIs on the actual EURO value. It’s our understanding that after the Beta, it will be clear if Merchants prefer to lock up REF or COMM in their Deals and we might possibly remove one option to avoid confusion.

5.3. Deal-affiliate link contract

This is the contract for any generic deal and will be used as contract factory, it will permit the following functions:

- Setup deal options like deadline, ICO maximum values...
- Update the above options by the deal owning merchant.
- Add affiliates with their own agreed specific conditions.
- Call RefToken definition contract for token transactions.
- Call RefToken exchange contract for REF <-> COMM conversion.

Merchants and affiliates can renegotiate the terms of the deal, in the instance that offering more favorable conditions would entice more affiliates to participate in their deal, or in the case of a merchant rejecting an affiliate's application(low reputation cases for example) but agrees to accept them if the Affiliate is ok with a lower reward. These are just examples of cases when it becomes useful to have this renegotiation pattern in the system. Check Fig. 5.1 for the whole flow overview.

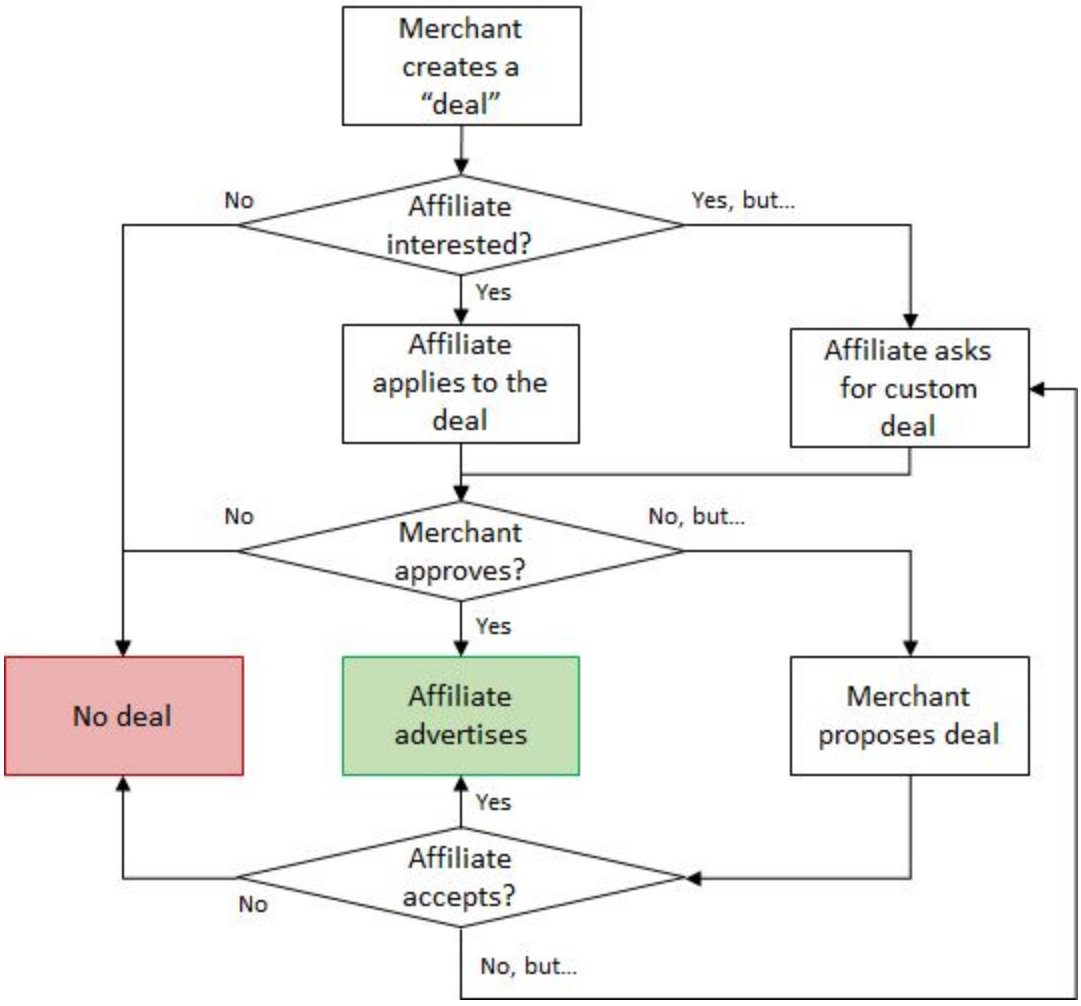

Fig. 5.1: Merchant <> Affiliate agreement process

This contract has a unique owner (*O*) (i.e. the Merchant who created the deal), and an array of Affiliates ($\{A_0,...,A_n\}$) approved by the owner, and some configuration variables (see Fig 5.2).

Element	Type	Description	Symbol	Allowed Ops (by owner)
Owner	Address	Contract's owner (i.e. deal creator)	*O*	Modify
REF/COMM switch	bool	To be set on creation, non modifiable.	*RCS*	None
Contract REF balance	Int	Locked REF to distribute among Affiliates	*RB*	Increase, Auto-Refund
Refund amount	Int	0 until deal expiration, refund amount after that	*RA*	None
Deadline	date	Date for remaining funds automatically return to contract's owner	*D*	Modify
Affiliates	Struct Array	Contains Affiliates data (see Fig 5.3)	$\{A_0,...,A_n\}$	Add, Modify
RefToken fee accumulated	Int	Total RefToken fee paid by this contract	*RFA*	None
RefToken fee value	Int	Reftoken fee, 5 by default	*RFV*	None
RefToken fee type	bool	Modifier for the "Reftoken fee (*RFV*)": Percentage (default false) or amount (true)	*RFT*	None

Fig. 5.2: Deal Factory content

Element	Type	Description	Symbol	Allowed Ops
Affiliate Wallet	Address	Address where the rewards will be sent to	*AW*	None
REF Rewarded	Int	Accumulated reward in REF	*RR*	None
Reward Value	Int	Reward agreed after negotiation	*RV*	None
Reward Type	bool	Modifier for the "Reward Value (*RV*)": Percentage (false) or amount (true)	*RT*	None
Active	bool	Owner can cancel an agreement anytime, so we set this to inactive but keep the struct in the array for the record.	*AF*	Modify by owner

Fig. 5.3: Affiliate struct content

This contract will contain a function to check the fulfillment of the deal conditions and call another function that will calculate the amount of comission to be transferred to the affiliate as payment, calculate the fees for use of the RefToken platform, deduct the amount of the contract's balance and call the corresponding methods in RefToken definition contract to effectively do the transactions.

So for any states S_i and its following S_{i+1} there will be the following changes:

- $RB|S_{i+1} = RB|S_i - \sum_{k=0}^{n} A_k|(RV^*RT + RV^*(\neg RT)/100) - (RFV^*RFT + RFV^*(\neg RFT)/100)$
- $A_k|RR|S_{i+1} = A_k|RR|S_i + A_k|(RV^*RT + RV^*(\neg RT)/100)$

For any state S_i the sum of the remaining balance *RB*, all the affiliates accumulated reward and RefToken fees needs to be the same, and match the originally locked tokens:

$$RB|S_i + \sum_{k=0}^{n} A_k|(RV^*RT + RV^*(\neg RT)/100)|S_i + RFA|S_i = RB|S_{i+1} + \sum_{k=0}^{n} A_k|(RV^*RT + RV^*(\neg RT)/100)|S_{i+1} + RFA|S_{i+1}$$

5.4. Deal conditions contract

This is not a specific contract, but a set of many contracts of different types, covering all the deal conditionals the platform will allow: Deposit counting, deposit amount, ICO... It inherits from the Deal-affiliate link contract, and adds the specific conditions to fulfill in its specific type of deal.

In the first phase of this project, we are going to create a working ICO deal contract, see next subsection for details.

5.4.1. First deal contract type: ICO contribution

This is the main contract for ICO deals, thanks to inheriting from the deal factory it will contain all its parent functions, plus the specific ones for condition checking. See Fig 5.4 to see the workflow.

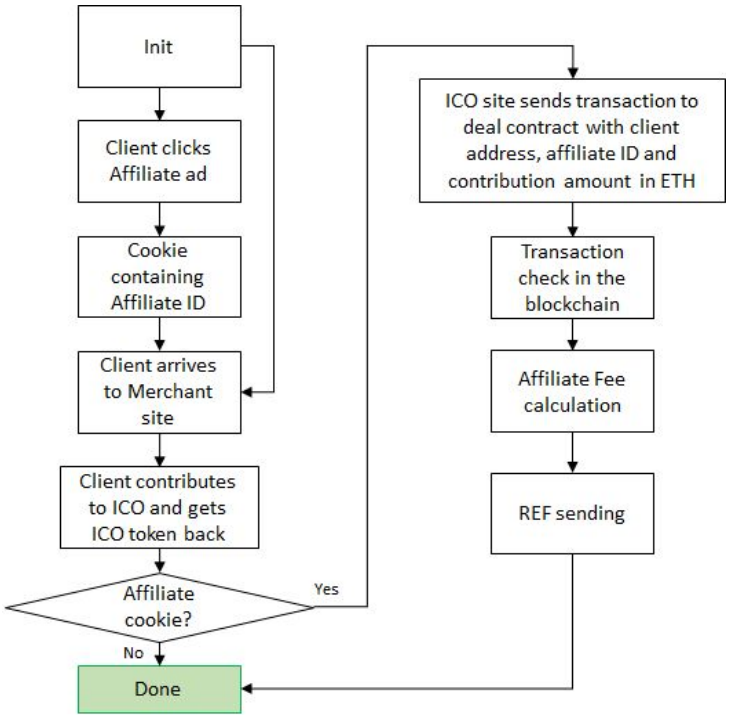

Fig. 5.4: ICO deal flow

6. Storage

6.1. Blockchain storage

In order to minimize gas and processing costs, we are going to reduce the required blockchain storage to the minimal necessary.

With that goal in mind, we have decided to only store in our smart contracts the information pertaining to merchant-affiliate transactions. Everything else will be stored in a regular SQL database.

6.2. SQL DB storage

We are going to use phpMyAdmin client to manage our SQL DB. It will contain registration data for merchants and affiliates, deals, transactions and performance measures. Everything we need to provide with high quality BI dashboards to both merchants and affiliates registered in our platform about their activity, so they can take the best business decisions.

This approach reduces the load on the blockchain side, making it more efficient than using a UI retrieving and sending data directly to the blockchain with the derived gas costs that option comes with.

7. Future plans

After setting up a solid base with this fully working ICO platform, our plans include improvements both on User Experience and System Automation. We want to allow other types of deals based on the same structure, be able to track clients activity on the blockchain, automation and customization in deal contract generation.

During the first phases RefToken won't be 100% Decentralized, we will have a centralised SQL DB and we will act as arbitrator when there is an dispute. The rationale behind this is the avoidance of any critical issues in platform's inception due to a vulnerability in a smart contract Deal. We are however planning to move to a 100% decentralised platform after phase 1 when the platform is more mature and stable with a decentralised DB and incentivised users as arbitrators. This will also put us in a better position to make the best choice when choosing a decentralised DB solution as the ones available are still under heavy development and the area will evolve a lot in the next 1-2 years making more stable options available for Dapps.

8. References

1) Ethereum White paper - https://github.com/ethereum/wiki/wiki/White-Paper
2) ERC-20 token info - https://theethereum.wiki/w/index.php/ERC20_Token_Standard