Redeemeum Protocol White Paper

An open blockchain protocol that represents real world products and services as smart voucher tokens

Authors: Justin Banon, Gregor Boroša

Version: v1.0, April 2019

Abstract

Redeemeum Protocol introduces a smart voucher, as a programmable, cryptographic token which represents the right to claim goods or services in accordance with terms which are programmed within and enforced by smart contracts. We see smart vouchers as a new financial primitive; that is: a modular, composable building block of the decentralized stack. As such, smart vouchers represent a powerful new form of economic coordination with the potential to disrupt centralised platforms. Redeemeum Protocol is a universal smart voucher trading system that almost eliminates transaction costs whilst improving triangulation, transfer and trust, and enables an open, competitive market via a public utility that protects participants from market power abuses. Adoption of Redeemeum is driven by a major paradigm shift in consumer behaviour away from trusted intermediaries and towards trust-minimised institutions for economic coordination that align network and participant incentives via tokens. Redeemeum is a blockchain protocol that enables issuing, administering and trading of programmable voucher tokens using smart contracts, keeper marketplaces and standardized voucher specifications and interfaces. The basis of Redeemeum Protocol lies in blockchain technology. Redeemeum employs a multi-sided tokenised business model comprising a nearly free side to drive adoption, a subsidised side to incentivize target behaviours such as data sharing and a paid side providing businesses with access to data and marketing services and applications.

Index

1 Introduction

1.1 Redeemable promises

Redeemable promises play an important role within the economy as a type of contractual claim that can be purchased, earned or rewarded in advance and then redeemed or collected at a later date.

Redeemable promises are manifested in various form factors including *vouchers, gift certificates* and *cards, discount coupons, membership cards and loyalty points;* or simply a proof of purchase used to collect goods or services.

Such promises can be generalized to a *voucher contract*:

> "A *'voucher'* represents a redeemable promise *or "a digital representation of the right to claim services or goods"*[1] *Fujimura*

1.2 Smart vouchers

We introduce the concept of *smart vouchers*:

> A *smart voucher* is a cryptographic token which represents the right to claim goods or services in accordance with terms which are programmed within and enforced by smart contracts.

We see smart vouchers as a new financial primitive; that is: a modular, composable and programmable building block of the decentralized stack which developers of decentralised applications will "recombine in novel ways to form increasingly complex hierarchies of technologies"[2][3]. As such, smart vouchers represent a powerful new form of economic coordination with the potential to disrupt centralised platforms and "shrink the domain of firms and expand the domain of markets"[4] by:

> *"enabling a spot market exchange to carry forward indefinitely a pure promise"*[5]
> .

---

[1] "Requirements and Design for Voucher Trading System (VTS)." https://www.rfc-editor.org/pdfrfc/rfc3506.txt.pdf.

[2] "The nature of the crypto technological revolution: – Hacker Noon." 16 Jun. 2018, https://hackernoon.com/the-nature-of-the-crypto-technological-revolution-4eb883c54d56. Accessed 4 May. 2019.

[3] "The Nature of Technology: What It Is and How It Evolves - Amazon UK." https://www.amazon.co.uk/Nature-Technology-What-How-Evolves/dp/0141031638. Accessed 4 May. 2019.

[4] "Economics of Blockchain - Archive ouverte HAL." https://hal.archives-ouvertes.fr/hal-01382002/document. Accessed 2 May. 2019.

[5] "Economics of Blockchain by Sinclair Davidson, Primavera De ... - SSRN." 9 Mar. 2016, https://www.ssrn.com/abstract=2744751. Accessed 26 Apr. 2019.

2 Current and future solutions

2.1 Centralised voucher schemes

Despite the potential utility of redeemable promises for coordinating economic activity, their adoption has thus far been limited. This is due primarily to the requirement for centralised entities to operate voucher systems.

Digital voucher systems require a centralised voucher scheme operator to either maintain a centralized server or to issue distributed smart-cards[6]. Such voucher scheme operators are typically firms in competition, for example rival e-commerce platforms, travel ancillary intermediaries, voucher software platforms or group-buying firms.

Rather than adopt a common voucher standard, competing firms are incentivized to build siloed, proprietary platforms and issue incompatible vouchers in order to lock-out competitors and lock-in customers and suppliers. As a consequence, many incompatible, proprietary voucher systems fragment the market. As highlighted within the Voucher Trading System RFC[3] (VTS), fragmentation multiplies implementation costs:

> *"If a different issuing or collecting system to handle such points or coupons must be developed for each individual application, the implementation cost will be excessive, inhibiting the use of such mechanisms."*[3] *Fujimura*

2.1.1 The role of firms

In general, firms exist as an alternative to markets wherever they are a more efficient institution for economic coordination[7][8]. Whilst markets are efficient for pure exchange transactions; where there are significant transaction costs associated with *triangulation, transfer* or *trust*[7], then firms, or more specifically platforms, reduce transaction costs. These costs are described as follows:

[6] "RFC 4154 - Voucher Trading System Application ... - IETF Tools." https://tools.ietf.org/html/4154. Accessed 25 Apr. 2019.
[7] "The Nature of the Firm - Coase - 1937 - Economica - Wiley Online ...." https://onlinelibrary.wiley.com/doi/full/10.1111/j.1468-0335.1937.tb00002.x. Accessed 20 Apr. 2019.
[8] "Transaction-Cost Economics: The Governance of Contractual ...." https://www.business.illinois.edu/josephm/BA549_Fall%202010/Session%203/Williamson%20%281979%29.pdf. Accessed 1 May. 2019.

*"**Triangulation** information about identity and location, and agreeing on terms, including price*
***Transfer** a way of transferring payment and good that is immediate and as invisible as possible*
***Trust** a way of outsourcing assurance of honesty and performance of the terms of the contract."*[9] *Munger*

2.1.2 Limited transaction cost efficiencies

Although voucher scheme operators reduce transaction costs versus markets, they do so in limited ways.

2.1.2.1 Triangulation

- *Choice and discovery* - restricted due to closed, proprietary networks.
- *Product and customer data* - stored within siloed, centralized databases, which enable information asymmetries and tampering.
- *Market and customer access* - is blocked to prevent off-platform transactions.

2.1.2.2 Transfer

- *Pricing* - fixed, aggregate level pricing prevents yield maximisation.
- *Voucher contract terms* - not programmable resulting in suboptimal allocation.
- *Redemption interoperability* - redemption is restricted to a closed, proprietary network through incompatible proprietary vouchers.
- *Payment* - via the trust model involves: reversible transactions, fraud, privacy breaches, high fees and arbitration via a financial intermediary[10].

2.1.2.3 Trust

- *Transaction regulation* - may be skewed in favour of the platform, information is asymmetric and conflicts of interest exist.
- *Rent-seeking* - firm-based platforms suffer from the *"endemic dysfunction of centralised systems"*[11] since they have a fiduciary responsibility to wield market power in the pursuit of maximal shareholder value[12][13].
- *Market structure* - at a macro-level the extraction imperative leads to inefficient and monopolistic market structures.

---

[9] "Tomorrow 3.0: Transaction Costs and the Sharing ... - Amazon.com." https://www.amazon.com/Tomorrow-3-0-Transaction-Cambridge-Economics/dp/1108447341. Accessed 20 Apr. 2019.
[10] "Bitcoin: A Peer-to-Peer Electronic Cash System - Bitcoin.org." https://bitcoin.org/bitcoin.pdf?. Accessed 2 May. 2019.
[11] "Economics of Blockchain by Sinclair Davidson, Primavera De Filippi ...." 9 Mar. 2016, https://www.ssrn.com/abstract=2744751. Accessed 1 May. 2019.
[12] "The Future Of Network Effects: Tokenization and the End of Extraction." 17 Jul. 2018, https://medium.com/public-market/the-future-of-network-effects-tokenization-and-the-end-of-extraction-a0f895639ffb. Accessed 20 Apr. 2019.
[13] "Why Decentralization Matters – Featured Stories – Medium." 18 Feb. 2018, https://medium.com/s/story/why-decentralization-matters-5e3f79f7638e. Accessed 20 Apr. 2019.

Firm-based voucher scheme operators reduce transaction fees versus markets, by reducing opportunism through increased triangulation, transfer and trust. However, they do so in a limited, local way and at the expense of increased market-wide transaction costs and inefficient markets. Blockchains represent an alternative and potentially more efficient institution for economic coordination of redeemable promises.

2.2 Redeemeum Protocol

Redeemeum is an open protocol for issuing, trading and redeeming smart vouchers as programmable, cryptographic tokens which represent the right to claim goods or services in accordance with terms which are programmed within and enforced by voucher smart contracts.

2.2.1 Almost costless transactions

Redeemeum increases voucher transaction cost efficiency to the point of making smart voucher transactions almost costless across the following key dimensions:

2.2.1.1 Triangulation

Finding information and locating opportunities for mutually beneficial exchange.

- *Choice and discovery* - is extended via a transparent, open market.
- *Product and customer data* - enables real-time, secure, self-sovereign customer usage and feedback data for each smart voucher instance.
- *Market and customer access* - open market structure enables access to customers and peers.

2.2.1.2 Transfer

Programming price and terms, facilitating payment and enabling redemption

- *Pricing* - pricing can be set dynamically for each smart voucher instance.
- *Programmable Terms* - smart vouchers enable precise terms to be programmed for each asset instance, enabling optimal allocation and yield.
- *Redemption interoperability* - as an open, generic protocol, redemptions of goods and services are enabled by smart vouchers issued as cryptographic tokens on the Ethereum blockchain under the ERC721 standard.
- *Implementation costs* - as a universal protocol, Redeemeum reduces the costs of implementing and integrating multiple proprietary systems.
- *Payment* - payment and fee disbursal functionality enables secure, low-cost transactions with dispute arbitration via a trusted intermediary who is game-theoretically incentivized to behave fairly.

2.2.1.3 Trust

- *Transaction regulation* - trust is increased by cryptographic enforcement of on-chain terms; and reduction of opportunism for off-chain terms through public transparency, decentralized reputation, and arbitrated escrow.
- *Rent-seeking* - Redeemeum Protocol removes the need for centralised intermediaries, whilst retaining the option of trusted keepers who perform valuable jobs for the network and compete for compensatory fees[14].
- *Market structure* - the protocol's decentralized governance structure provides cryptographically-secured protection against monopolistic rent extraction and promotes an open, competitive market.

Redeemeum Protocol is a universal smart voucher system that almost eliminates transaction costs associated with triangulation, transfer and trust on a market-wide basis. Redeemeum fosters an open, competitive market via a decentralized public utility that protects participants from market power abuses.

2.3 Redeemeum value proposition

Redeemeum is an open blockchain protocol that represents real world products and services as smart voucher tokens that enable:

- efficient issuance, trading and redemption
- secure exchange of rich customer data
- an open, competitive market

[14] "Keepers — Workers that Maintain Blockchain Networks - Medium." 5 Aug. 2017, https://medium.com/@rzurrer/keepers-workers-that-maintain-blockchain-networks-a40182615b66. Accessed 1 May. 2019.

2.4 Solution comparison

The table below compares the current centralised intermediary model with the Redeemeum Protocol.

	Centralised intermediary	**Redeemeum Protocol**
Triangulation		
Choice & Discovery	Restricted choice, local discovery	Extended choice, global discovery
Product and customer data	Scarce, aggregate, insecure data	Rich, granular, secure data
Market and customer access	Restricted market access	Open market access
Transfer		
System transaction fees	Monopoly fees	Nearly free
Implementation costs	Multiplied per system	Once only
Pricing	Static pricing	Dynamic pricing
Yield management	Suboptimal yield	Yield maximised
Voucher contract terms	Fixed aggregate terms	Smart programmable
Redemption interoperability	Restricted compatibility	Global interoperability
Payment	Reversible, fraud, high fees, privacy breaching	Irreversible, secure, near zero fees, privacy preserving
Trust		
Transaction regulation	Incentive to abuse	Can't be evil
Rent-seeking	Extraction imperative	Cryptographically-secured protection
Market structure	Closed, monopolistic	Open, competitive

2.5 Example use cases

Redeemeum Protocol is a highly generic and universal protocol for the issuing of smart vouchers. The table below lists a small subset of potential use cases.

Consumer credit products	
Description	Offchain consumer credit products offer benefits as a means of differentiating products without competing on price / fees and terms. Common examples are travel benefits (airport lounge, spas) and insurance (trip, flight delay) to differentiate products and support fees.
Proposition	Redeemeum enables centralised and DeFi credit products to offer off-chain benefits (e.g. as travel and retail and decentralised insurance), with rich customer and data.

Rewards for loyalty programs	
Description	Loyalty programs offer rewards as redemption items for loyalty points.
Proposition	Redeemeum enables on-chain and off-chain loyalty programs and rewards tokens to convert tokens or points value into rewards, with rich customer and data.

E-commerce purchases	
Description	Ecommerce take payment in return for a promise redeemable on delivery or collection.
Proposition	Redeemeum enables the issuance of smart vouchers for off-chain products and services with funds escrowed on-chain.

Buying charitable acts	
Description	Charities request donations for charitable acts such as adopt a polar bear or plant a rainforest tree.
Proposition	Redeemeum smart vouchers enable low-cost contracting and public transparency.

Buying a membership on credit	
Description	Consumer purchases a membership to botanic gardens using credit
Proposition	Integration with e.g. Dharma or Maker enables consumers to purchase membership with funds borrowed from a decentralised lending pool.

3 Technical Requirements

Voucher Trading System should meet the following requirements[15]:

1. It MUST handle diverse types of vouchers issued by different issuers.
2. It MUST prevent illegal acts, like alteration, forgery, and ensure privacy.
3. It MUST be practical in terms of implementation/operation cost and efficiency.

Handling diversity

Redeemeum supports the issuance of vouchers by multiple actors on a universal protocol and handles multiple voucher types, including: coupons, gift certificates, loyalty points, membership cards, exchangeable tokens for goods or services etc.

Security guarantees

Preventing forgery: it is impossible to forge a Voucher Token as long as standard cryptographic primitives remain secure. While traditional centralized voucher systems are typically limited in ways they can validate the redemption of a voucher, Redeemeum offers significantly more powerful verification capabilities.

Preventing alteration: vouchers cannot be altered after issuance. Voucher orders can only be cancelled by Producers.

Preventing duplicate redemption: a voucher can only be redeemed by the voucher holder. Once consumed, it cannot be redeemed again; if the voucher can be used repeatedly, for example a membership card providing 10% discount on all purchases, the voucher is not consumed until the final expiration period.

Non-repudiation: it is not possible to repudiate issuance, trade or redemption by actors, since their digital signatures bind them to the commitments.

Ensuring privacy

Despite the generally open nature of Redeemeum, it is possible to selectively shield certain parts of data, esp. personally identifiable information, by following a privacy by design principle.

Operational efficiency

Redeemeum enables discovery of globally available vouchers to anyone, which previously was impractical due to scattered and heterogeneous sources. The resilience of its operation rests in the distributed nature of the protocol. The mature, fully-developed stage of the protocol will be horizontally scalable to meet the global demand.

---

[15] "RFC 4154 - Voucher Trading System Application ... - IETF Tools." https://tools.ietf.org/html/4154. Accessed 13 Dec. 2018.

4 Protocol Specification

Redeemeum is a protocol that enables issuing, administering and trading of programmable voucher tokens using smart contracts, keeper marketplaces and standardized voucher specifications and interfaces.

Reedeemum leverages the architecture and methodology of Dharma protocol[16], which itself leverages 0x[17], in combination with the VTS (Voucher Trading System[18]) and Generic Voucher Language[19] specifications. Like Dharma, we use 0x Broadcast Order Messages as a primitive on top of which we create Redeemeum Voucher Orders. With Redeemeum actors, namely Producers, Aggregators, Relayers and Consumers being analogous to Dharma Debtors, Underwriters, Relayers and Creditors.

4.1 Agents

Agents are the platform participants between which Redeemeum Protocol seeks to facilitate interactions.

4.1.1 Producers

The Agent who is selling an asset to a Consumer for an agreed value.
Producer fees:

- Producers may be required to stake fees which may be slashed for non-performance.
- Producers may receive fees for performance.

4.1.2 Consumers

The Agent who is buying an asset from a Producer for an agreed value.

4.1.3 Collectors

Collectors are agents who verify and collect the voucher and implement voucher's promise.

Collectors perform the following functions:

- Commits to a Producer's offer to collect a certain category of vouchers.

---

[16] "Dharma White Paper · GitBook." https://whitepaper.dharma.io/. Accessed 8 Dec. 2018.
[17] "White Paper - 0xProject." 21 Feb. 2017, https://0xproject.com/pdfs/0x_white_paper.pdf. Accessed 8 Dec. 2018.
[18] "RFC 4154 - Voucher Trading System Application Programming ...." https://tools.ietf.org/html/4154. Accessed 8 Dec. 2018.
[19] "RFC 4153 - XML Voucher: Generic Voucher Language - IETF Tools." https://tools.ietf.org/html/4153. Accessed 8 Dec. 2018.

- If the Collector wishes to discontinue collecting vouchers, then they will need to cancel their agreement with the Producer.
- Verifies the voucher validity by triggering on-chain validation and potential off-chain checks.
- Attests to the consumer's conformity with the terms e.g. by *checking and inputting identity, age, eligibility.*
- Collects the voucher.
- Performs the voucher promise, typically by rendering goods or services.

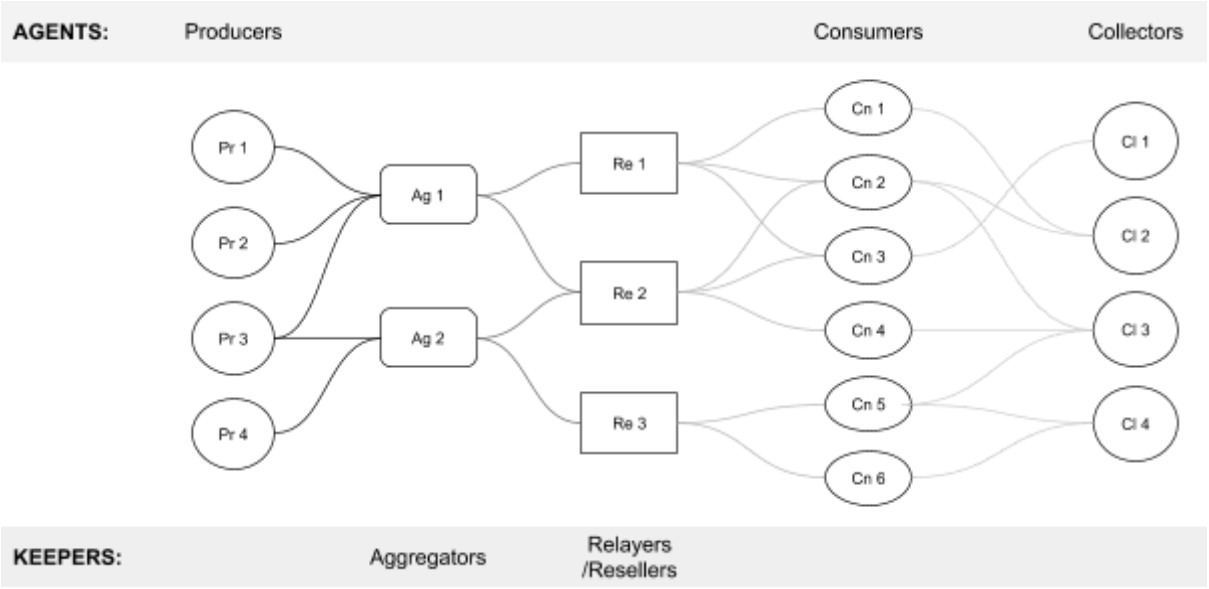
Collector fees:

- Collectors may be required to stake fees which may be slashed for non-performance.
- Collectors may receive fees for performance.

4.2 Keepers

In common with Dharma, we implement a marketplace comprising two Keepers[20], for which we use Ryan Zurrer's definition:

> *"a catchall term for the different utility players in distributed networks that maintain stability and perform crucial jobs in the crypto-economic model"*



Agents and Keepers - platform interaction diagram

[20] "Keepers — Workers that Maintain Blockchain Networks - Medium." 5 Aug. 2017, https://medium.com/@rzurrer/keepers-workers-that-maintain-blockchain-networks-a40182615b66. Accessed 8 Dec. 2018.

4.2.1 Aggregators

An Aggregator is a trusted entity that receives compensatory fees for performing the following functions:

- Originating a Voucher Order from a Producer
- Determining and negotiating the terms of the Voucher (i.e. value/items redeemable, conditions/restrictions) with the Producer
- Committing to the likelihood they ascribe to the voucher being redeemable, quality rating etc.
- Administering the Voucher Order's purchase by forwarding it to any number of relayers.
- Servicing the Voucher -- i.e. ensuring redemption to terms, Consumer satisfaction.
  In the case of defaults or complaints, arbitrating and deciding whether escrowed fees are forwarded to Producer or returned to Consumer.
- Aggregators are rated by:
  - Consumers - Consumer satisfaction, Voucher rating.
  - Producers - fairly handling disputes

  Therefore there is an incentive for Aggregators to curate high-quality Producers, resolve challenges fairly and ensure redemptions are made as agreed.

Aggregator fees:

- Aggregators may be required to stake fees which may be slashed for non-performance.
- Aggregators may receive fees for performance.

4.2.2 Relayers

At a basic level, relayers in Redeemeum Protocol perform an analogous function to relayers in 0x Protocol and Dharma Protocol.

Relayers perform the following functions:

- Relayers aggregate Voucher Orders from any number of Aggregators,
- for an agreed upon fee, host the messages in a centralized order book,
- provide Consumers with the ability to purchase the requested Vouchers.
- Relayers need not hold any agent's tokens, but may do so (see Value-add Relayers below)

Note:

1. Redeemeum Protocol Relayers provide Consumers with signed voucher-specific metadata associated with the accompanying Aggregator so that they can make informed purchase decisions about the quality profile of a given Voucher order.
2. Redeemeum Protocol relayers do not freely allow any anonymous party to publish signed Voucher Orders on to their order book, and use their discretion to only accept Voucher Orders from known, trusted Aggregators.

We define two types of Relayers within Redeemeum Protocol:

- **Relayers** - these will provide simple order book functionality, requiring Consumers to take custody of and manage Voucher Tokens themselves. Radar Relay is an example of a typical Simple Relayer.
- **Resellers** - in addition to hosting an order book, Resellers will abstract away the underlying technology by purchasing as B2B buyers and maintaining custody of vouchers on behalf of Consumers. Resellers will then present a standardized UX to Consumers within wallets and consumer-facing apps. Examples of potential Value-add Relayers include: OTAs, Loyalty Programs and Payment Card Networks.

4.3 Voucher Specification Overview

We use the Generic Voucher Language definition of a voucher:

> *A voucher is a logical entity that represents a right to claim goods or services. A voucher can be used to transfer a wide range of electronic values, including coupons, tickets, loyalty points, and gift certificates, which often have to be processed in the course of payment and/or delivery transactions.*

Asset is claimed at the redemption of a voucher, i.e. the goods or services delivered. It is offered by the Producer to the Consumer and is the basis that a voucher token is referencing.

Promise denotes the promise of the Producer to deliver the Asset to the Consumer under programmable restrictions and funds allocation. Promises are a core construct that enable reusability across a multitude of participants, while still being anchored to the same Producer.

Promise-Collection construct defines the agreements between Producers and Collectors, for example, which Collectors are authorized for a set of vouchers and the fees they expect in return. If there are no restrictions, it can be omitted.

Voucher Offer then wraps a Promise, adding processing details, such as groupings, validity period etc. It is a base building block for the issuing of vouchers. Offers are uniformly addressable across all Aggregators and Relayers.

Voucher Order is a construct in the hands of the keepers. There can be multiple Voucher Orders created from a single Voucher Offer as each derived Order is assigned to a different Aggregator-Relayer pair, committing to their fees, Aggregator's rating of the voucher etc.

Smart Voucher, or synonymously, **Voucher Token** is created when a Voucher Order is filled - by transferring the requested funds from either the Consumer or

from the Value-Added Relayer. Funds remain locked until redemption or potential dispute resolution. The holder can then redeem this token for the promised asset.

Note: the process is sequential, but certain types of vouchers and certain use-cases do not require all actors to be involved. The full set of objects maximizes reusability in complex paths.

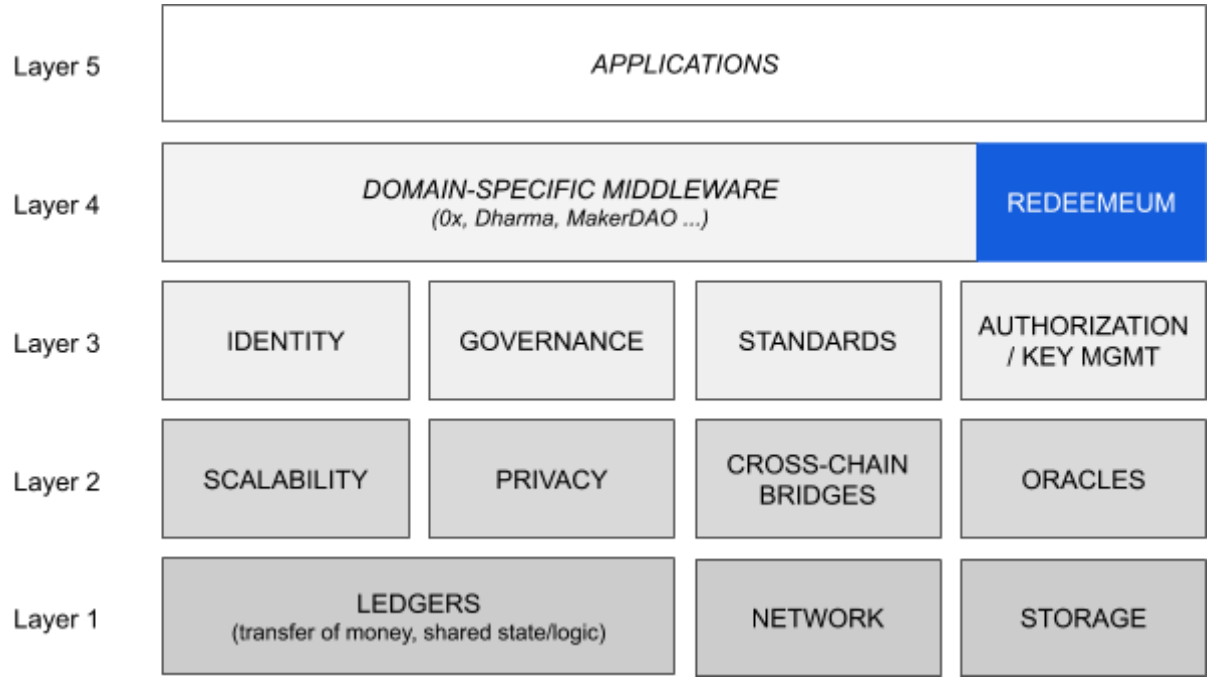
For a more comprehensive specification see: *Appendix - Detailed Voucher Specification*.

5 Technology and Architecture

5.1 Introduction to Technology

The basis of Redeemeum Protocol lies in blockchain technology. Building on the extensive pioneering work done by 0x and Dharma teams, Redeemeum Protocol is adapting their architecture to support a more tangible, oftentimes physical nature of voucher promises. It includes a set of basic constructs (a promise, an offer etc.) and logic for matching the supply and demand side of the market.

In the wider ecosystem, Redeemeum Protocol sits on top of domain-neutral, predominantly technocratic platforms and just below the application layer. It is at the lowest contact with a specific usage domain (e.g. vouchers) and is to be used by applications built on top of it. Thus it can be considered as **middleware**, as shown in the diagram.



The position of Redeemeum in the wider stack

As middleware, Redeemeum builds on sub-systems from the lower layers: decentralized identity platforms to address its users (e.g. 3box), relies on governance protocols to manage its operation (e.g. dxDAO), conforms to the standards for its inputs-outputs (e.g. ERC-721) and uses proven key management services (e.g. Clef). This also marks the scope of Redeemeum.

Likewise, top-layer applications combine middleware components to create new value for their users. For example, a marriage between Dharma protocol and Redeemeum unlocks the option of buying a botanic garden membership card (which is one type of a voucher) by borrowing the funds from a decentralized lending pool, with repayment next month. This could be achieved in a single transaction.

5.2 Emergent Design

Operating in a nascent space, there are several parts impractical to be implemented with the current state-of-the-art permissionless blockchains, facing critical issues with the protection of private data, limited throughput and unpredictable costs of transactions. While these keep on improving, we acknowledge that there are certain fundamental limitations and so Redeemeum is designed to leverage on blockchain's core features that are distinct in unrestricted - permissionless and restricted - permissioned settings. The latter case posing that block generators come from a smaller set, but the blockchain access is otherwise open to public.

As we've learned from the past years, the blockchain part of the layer 1 converges towards global settlement. The layers don't share the same context, so the security and flexibility assumptions can be different. This is manifested in the eventual emergence of application-specific blockchains - highly optimized and customized to a particular domain.

Redeemeum as middleware is designed to be a blockchain-based product on its own, but always coupled with layer 1 blockchain networks, such as Ethereum, for two specific reasons. *Firstly, token standards* on Ethereum have the largest adoption by far and they are only getting stronger, so much so that even corporate blockchain initiatives are taking steps towards interoperability[21]. *Secondly, the network effects* of Ethereum and its *superior development* community make it possible to start building Redeemeum the soonest and contribute to the adoption of the disruptive blockchain technology.

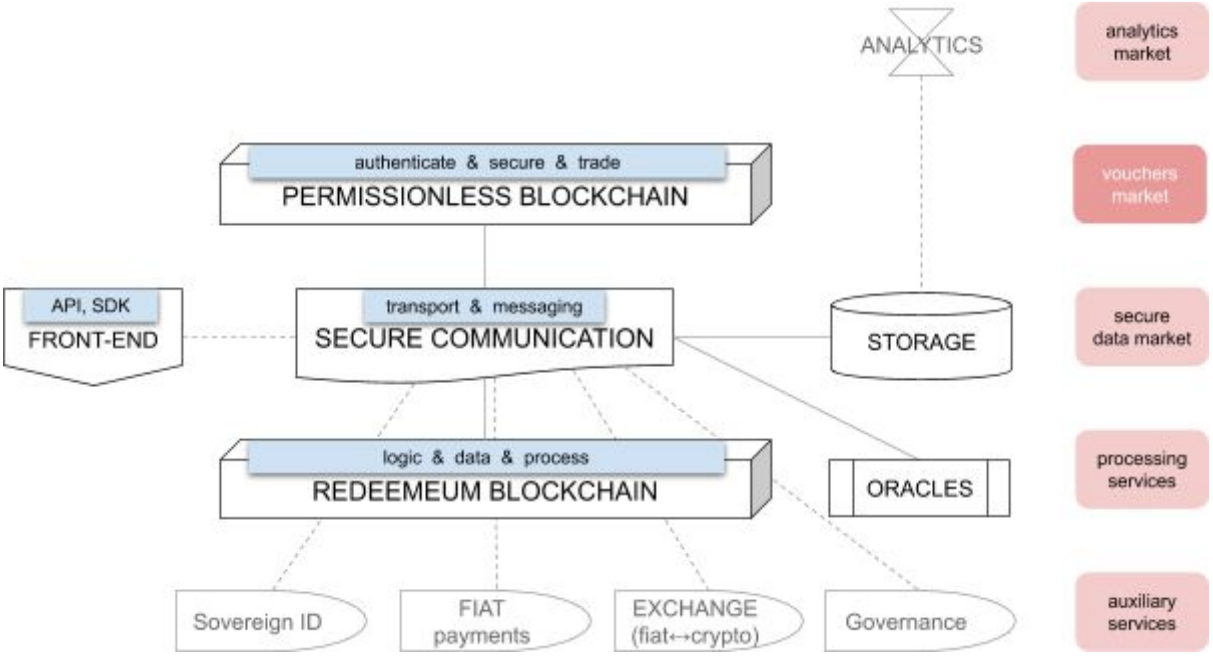
[21] "Enterprise Ethereum Alliance Launches Blockchain-Neutral Token Taxonomy Initiative to Accelerate a Token-Powered Blockchain Future – Enterprise Ethereum Alliance." 17 Apr. 2019, https://entethalliance.org/enterprise-ethereum-alliance-launches-blockchain-neutral-token-taxonomy-initiative-to-accelerate-a-token-powered-blockchain-future/. Accessed 26 Apr. 2019.

There is a downside of having a secondary shared ledger: the cost of securing it. The operational resources for this are expected to be low and mostly communication-bound, but node operators are expected to offer *secure and reliable service*, which bears non-negligible costs. It would be risky to expect they will provide a service to others simply because an operational protocol is beneficial for all. A safer approach is to actively reward them for their service, for example via a tenable token distribution model.

Therefore, Redeemeum is **starting with a dual-chain architecture** and is later expected to morph according to the maturity of its wider ecosystem. More details are described in *Appendix - Development Roadmap*.

5.3 Architecture Overview

The following diagram represents an overview of components in Redeemeum and interoperability with (semi-)external service providers.



Redeemeum technological overview

5.3.1 Vouchers market on Permissionless Blockchain

Issuance of voucher tokens, trading and redemption takes place on public, permissionless blockchain (e.g. Ethereum).

- **Token issuance** is triggered by accepting the funds from the Consumer, which in turn mints a voucher token, with the terms of redemption, costs and the ownership of the voucher explicitly and publicly defined.
- **Trading** is made possible with the voucher token being a standard non-fungible token (i.e. ERC-721 on Ethereum), therefore supported by the majority of token wallets. The protocol is not limiting secondary trading, unless vouchers are so configured.

- **Redemption** of the voucher is recorded in a secure and immutable manner, leaving no room for ambiguity. Additionally, it is uncovering the operational insights of the protocol.

The goal is not to place all functionalities on the permissionless blockchain - on the contrary, it is to be used as little as possible, but to leverage its main advantages: authentication of participants, secure creation of vouchers, secure transfers of funds and trading of tokens.

5.3.2 Processing on the Redeemeum Blockchain

Before a voucher token can be issued, its nature must be defined and made available by the keepers. This being a more intensive process in terms of computation and communication resources, Redeemeum implements it on a separate blockchain. It is closely tied to a permissionless blockchain on tier 1, but has lower operating costs and controlled write access.

Data: the data on this shared ledger is generally open, with exceptions that protect personal data and potential voucher-specific business secrets. Off-chain data can be made private using standard cryptographic methods, while on-chain privacy benefits from lower transaction costs on Redeemeum blockchain and could eventually support a wider range of pre-compiled cryptographic operations.

Funds: vouchers are programmed to support simple distribution of funds (fees) as per the Terms Contract. The unavoidable technological fees, such as blockchain gas, are paid by the transaction initiator. Redeemeum also supports covering such fees by a predefined actor.

Processing consists of several actions, that are executed and recorded on-chain:

- the initial definition of the underlying asset and the promise for its redemption,
- the fee and delivery negotiating between the keepers,
- the minting of voucher tokens and
- final steps at redemption, i.e.validation, execution, rating and potential challenge.

Once the Consumer sees an available voucher and decides to purchase it, three actions happen atomically: on the Redeemeum chain the voucher order gets filled, on the permissionless chain the Consumer's funds are locked and a unique, non-fungible token is minted, representing Consumer's right to the redemption of the voucher.

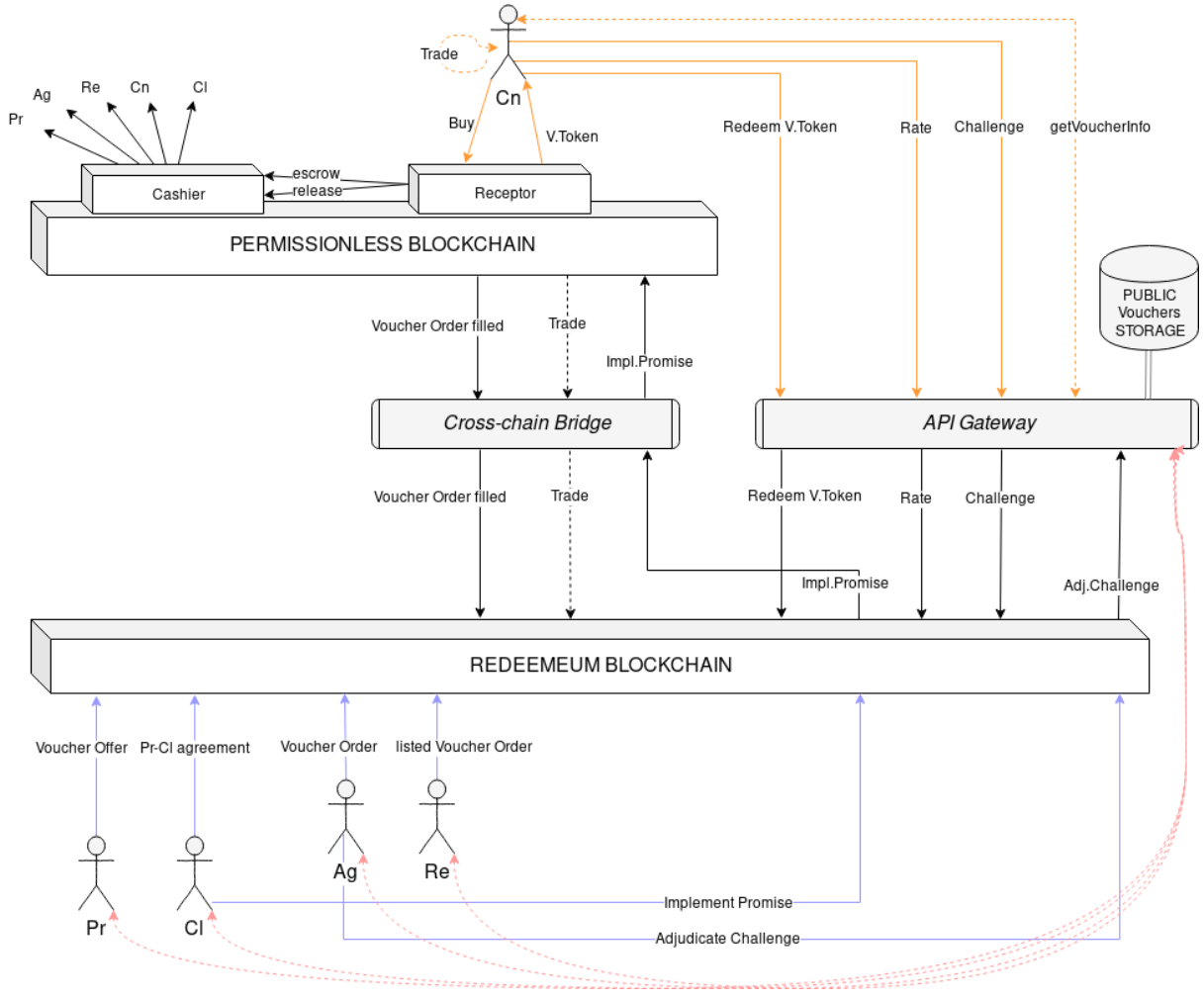
Filling of the order triggers broadcasting a message to decrease the remaining quantity of that particular voucher offer. It is up to the Relayer to display one order per group or as many orders as are vouchers in the group. The whole voucher's lifecycle is recorded on the blockchain and this makes it possible to assert the quantity of currently available vouchers.

The holder of the voucher can trade it further like any other non-fungible token, unless restrictions of the voucher prohibit it. In such case, the redemption under the new owner will not succeed as the Collector will not be able to match it against the previously agreed terms.

When the Consumer requests the redemption of the voucher, it is checked against its restrictions. Firstly, the Collector's account is checked against agreements with the Producer. Then, other terms are checked. The Collector might need to perform some of these checks off-chain. If the checks pass, the Collector collects the voucher and delivers the promised goods or service.

Note: in case the voucher is issued as a promise for a natively on-chain asset, such as a token or similar crypto-asset, then the collection can be completely automated and the Collector replaced with a zero-fee smart contract by following a hash commitment scheme.

The diagram below describes high-level interactions.



Overview of components and interactions in Redeemeum

All participants can use a classic API gateway as a convenience, but notably the Consumer only interacts with the permissionless blockchain, while all others mainly interact with the Redeemeum chain and access permissionless chain mainly for payments. From the Consumer's perspective, the protocol is comparatively flat.

5.3.3 Communications

Communications protocol is serving as transport rails for data as well as messaging between participants. It uses end-to-end encryption - note that parties are aware of each other's addresses, so asymmetric encryption can be used and form private *per-voucher channels*. While the finalized agreements are visible to others, the prior negotiation is private. Communications are also based on a decentralized model, e.g. Postal Service over Swarm and Swarm Feeds.

5.3.4 Storage & Data

To minimize on-chain data for reasons of costs and ease of access, space-consuming data is stored on a specialized decentralized storage system like Swarm, including additional descriptions of vouchers, rich content, voucher ratings, keepers reputation scores etc.

Attention is given to the data and metadata that unlocks valuable insights into the operation and usage of the protocol, such as product data, routing metadata, quantitative & qualitative ratings, value for money etc. All processing and data exposure can be automated through external platforms such as Ocean Protocol, prepared for further use in data marketplaces, analytics etc. Where appropriate, it is consent-driven and processed to protect the privacy and anonymity[22].

The data is a valuable part of the platform, but not all of it can be shared openly in the same vein as the open-source code of the platform, because the platform doesn't own it. Instead, it is the users who control their data and should they agree to share it selectively, that might be against reimbursement and privacy-preserving guarantees.

5.3.5 Front-end & Developers support

Enforcing compliance with the protocol is a given using blockchain technology, but that is covering mainly the back-end. The front-end is often more chatty, ephemeral and as such it implies a less strict approach, covered by providing a standardized API (Application Programming Interface) and an SDK (Software Development Kit) for keepers and consumer-facing applications.

The negotiation part of the process, where the details of vouchers get defined, is performed at the communication layer with the support of APIs. It does not require to be managed through the use of blockchain technology, as standard

[22] "Elastic Sensitivity - Noah Johnson, Joseph P. Near, Dawn Song." 4 Sep. 2018 https://github.com/uber/sql-differential-privacy . Accessed 10 Mar. 2019.

cryptographic methods suffice. Redeemeum Protocol does not impose restrictions about how participants communicate in any way, but it encourages the use of standard APIs to streamline the process and minimize the development time for users. Nonetheless, once the commitments are submitted to the blockchain, the smart contracts will only allow the process flow as it was agreed upon.

5.3.6 Interoperability with the wider ecosystem

Rather than reinventing the wheel for each auxiliary component, the protocol evolves best when it is in sync with the wider ecosystem, by plugging-in the best currently available solutions. "*The most successful networks, projects, and organisations within the Convergence Ecosystem will be those that have inclusive and aligned communities. The Convergence Ecosystem drives collaboration rather than competition.*"[23]

For these reasons, Redeemeum Protocol is working hand-in-hand with platforms that provide services such as decentralized identity - a long awaited tech that is finally within reach due to blockchain technology. Self-sovereign identity, as its subset, is giving users a true control over their digital identity, because it is consent-based and following the principle of least disclosure. The DID provider will be chosen based on several requirements, such as the ability to use multiple addresses linked to a single identifier, key recovery mechanism to ensure an uninterrupted user experience and compatibility with Redeemeum stack.

Converting between currencies, loans and refactoring are auxiliary services that are naturally left to specialized providers. Likewise, accessing off-chain data that is out of reach of smart contracts and triggering of on-chain transactions are some of the tasks of oracles.

Interoperability with existing, legacy systems is important for the adoption and even though smart contracts can't access the off-chain world, Redeemeum has placeholders to store passive data which the legacy software can interpret and trigger some actions. For example, it is possible to attach arbitrary data to the voucher, that is needed to verify the redemption conditions, such as a special code to redeem the voucher for a rent-a-car service.

Redeemeum, a web3 component for vouchers. The capability of blockchain to empower the edges by unlocking a decentralized internet with read/write/execute operations, also re-defines how vouchers can be used. As a web3 component, Redeemeum provides a uniform way to issue, trade and redeem a diverse set of voucher types. It can be embedded into other DApps, for example to issue exchangeable tickets or to reward their users with redeemable tokens for goods/services, a.k.a. smart vouchers.

---

[23] "The ConvergenceEcosystem, Building the Decentralised Future - Outlier Ventures." https://outlierventures.io/wp-content/uploads/2018/03/The_Convergence_Ecosystem_Report_Outlier_Ventures_2018.pdf . Accessed 15 Feb. 2019.

6 Voucher Issuance and Redemption Process

A voucher token is issued when a voucher order is filled, passing all checks and payment forwarded to escrow. Any party can submit a signed order to Redeemeum smart contracts.

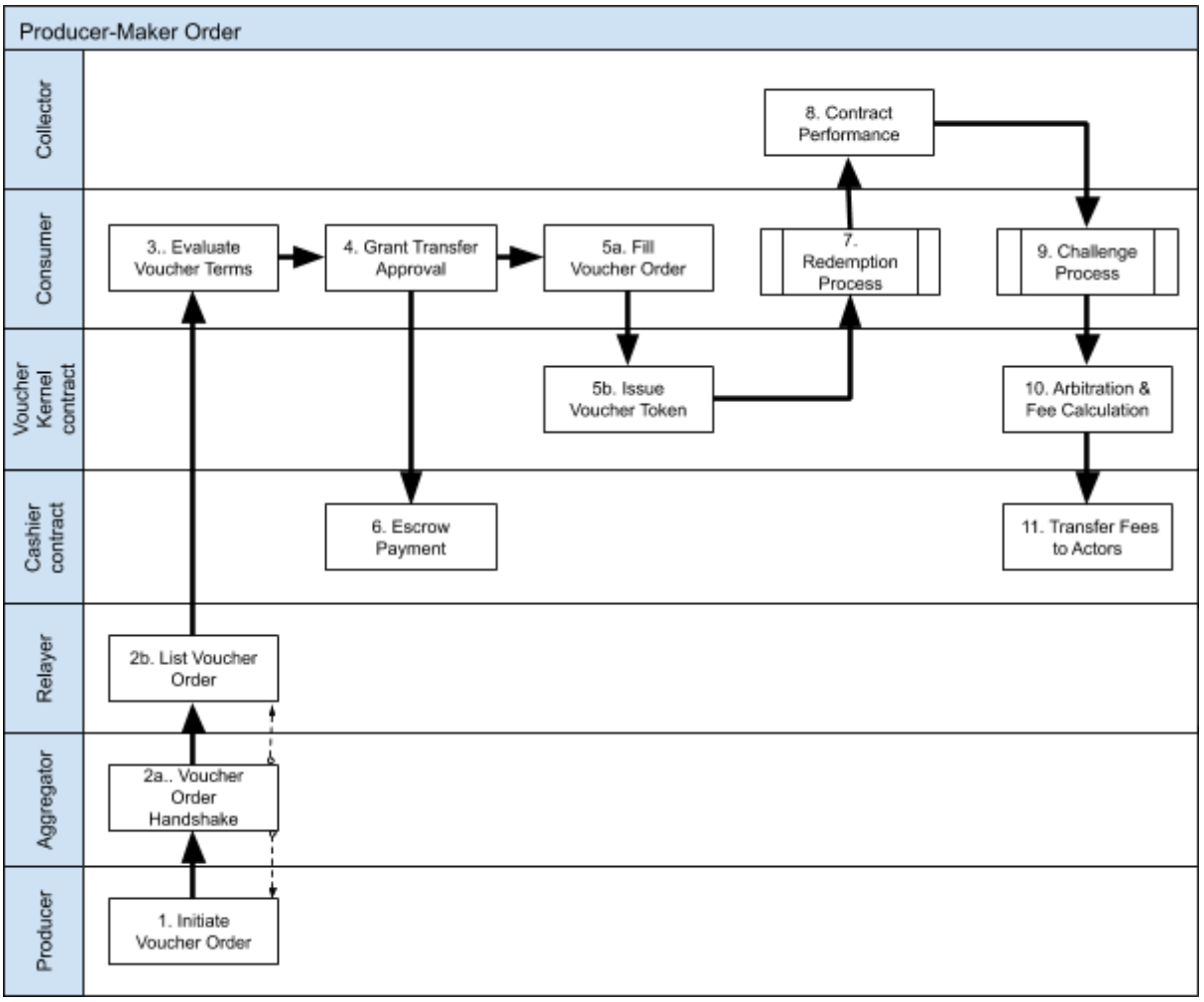
Note: zero-pay vouchers are also possible when all parties relinquish their fees.

Note also: henceforth, complex paths are described. Redeemeum Protocol does, however, not prevent simpler flows, e.g. where the Producer is also the Collector and/or Aggregator, or Relayers are omitted, or there are multiple Aggregators and/or Relayers for a single voucher offer.

- **Producer-maker Orders** - where the Producer is offering assets for sale, typical for voucher type of exchangeable tokens; *the focus of this chapter*.
- **Consumer-maker Orders** - where Consumer is requesting to obtain a voucher, typical for voucher types such as membership cards, gift certificates etc.

6.1 Producer-Maker Orders

The following diagram describes voucher lifecycle from Producer's offer till redemption.



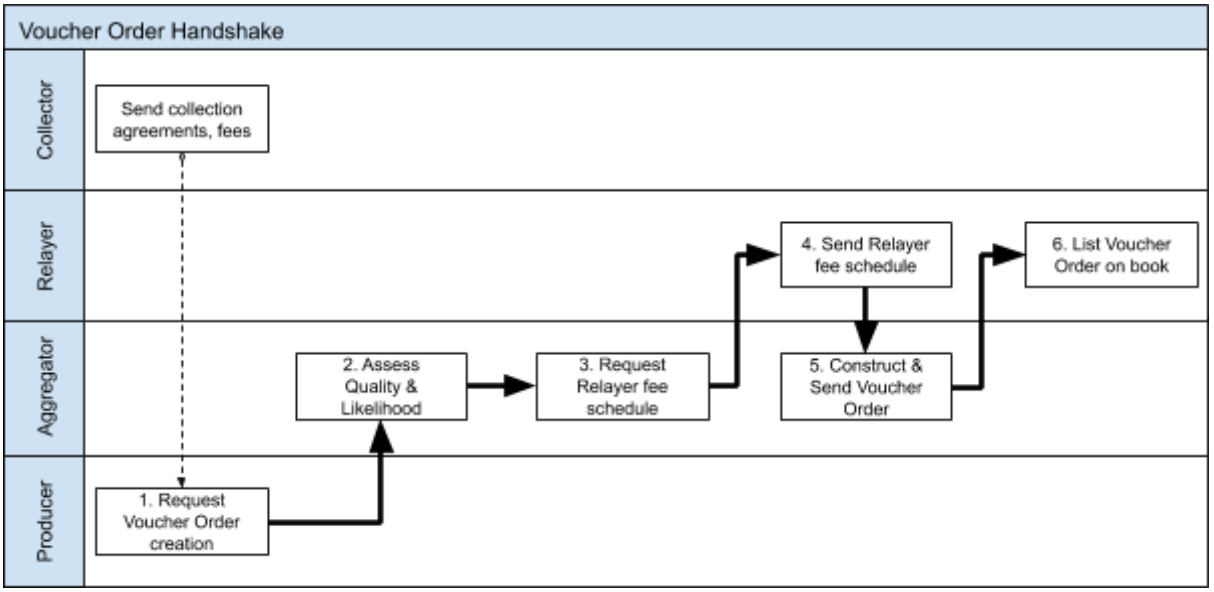
1. Producer commits to a promise of delivering a good or service. This commitment is an offer to an Aggregator, who then assembles it into a voucher order. Prior to that, Producer can negotiate the delivery with a set of Collectors.
2. (a) Voucher Order handshake (see details below) is performed between Producer, Aggregator and Relayers, (b) resulting in the Relayer listing a complete Voucher Order.
3. Consumer evaluates terms of Voucher Order on Relayer's public order book.
4. If Consumer wants to buy a voucher, i.e. fill an order, Consumer grants approval for transferring sufficient funds. *Note that a single transfer approval may be sufficient for multiple vouchers, potentially bought later.*
5. The Consumer then (a) fills the Voucher Order. Voucher Kernel contract then (b) issues to the Consumer a non-fungible, non-divisible token representing the Consumer's agreement to the terms contract and the right to redemption.
6. The Cashier contract transfers the payment amount into escrow.

7. Later on, the Consumer initiates the Redemption Process (detailed below).
8. The Collector performs the contract (this could be providing a service, delivering a product or applying a discount or promotion).
9. The Consumer may challenge the redemption during the challenge period, in which case the Aggregator will perform arbitration (or invoke an external service) and may recalculate fees and/or slash deposits.
10. The Cashier contract transfers calculated fees to actors.

6.2 Voucher Order Handshake

The following diagram and text describes the Voucher Order Handshake process, where Producers, Aggregators and Relayers determine and agree on voucher's qualitative rating and distribution of fees.

Note: the agreements between Producer and Collectors, who are authorized to collect a set of vouchers, are arranged beforehand and need not be repeated for every Voucher Order.

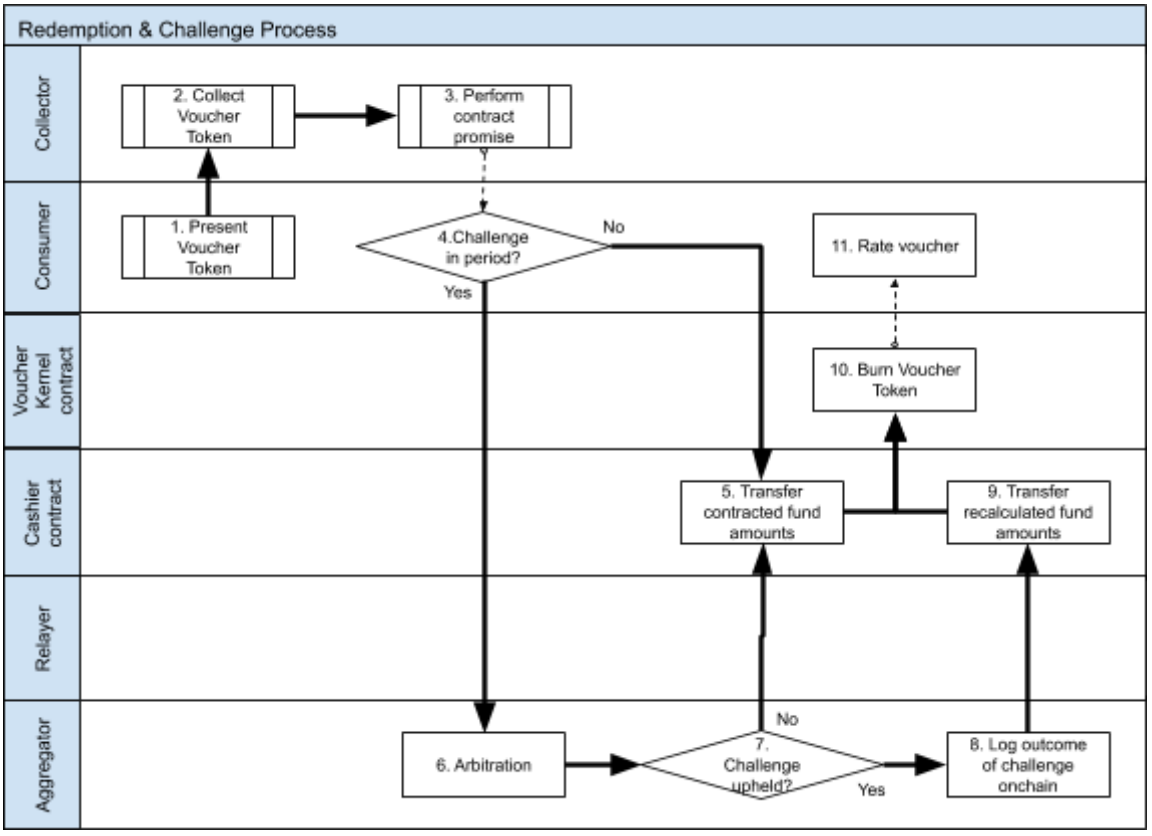


1. Producer requests creation of a Voucher Order from an Aggregator by sending a Voucher Offer, with defined promise of goods/services to be delivered under defined conditions by contracted Collectors.
2. Aggregator assesses the Likelihood of Redemption and Quality Rating of the offer.
3. If Aggregator wants to use Relayer(s), then Aggregator requests their fee schedule.
4. Relayer sends fee schedule and receiving address to Aggregator.
5. If fees comply with Aggregator parameters, then Aggregator constructs a complete Voucher Order using inputs from Aggregator and Relayer, and assigns Voucher Order to Relayer.
6. Relayer lists completed Voucher Order on their order book.

6.3 Redemption & Challenge Process

The following diagram and text describes the redemption and challenge process, which assumes initial conditions where Voucher Kernel has transferred newly minted Voucher to Consumer and escrowed payment and any deposits.

Note: challenging is possible immediately after the minting of a Voucher Token, up until token expiration delayed by the full challenge period. E.g. a voucher token is minted at time T, expires at $T + 10$ with a challenge period of 3, then a challenge is possible in the interval $[T,\ T + 10 + 3)$.



1. Consumer presents voucher token to Collector.
2. Collector validates and collects voucher token.
3. Collector performs the promise (typically renders goods, services or promotional offer).
4. The challenge period starts ticking upon redemption. If the challenge period is undefined (i.e. equals 0), the voucher does not support challenges.
5. If the Consumer does not challenge before the end of the challenge period then Voucher Kernel contract sets *r-value* to 1 (redemption promise delivered), then it calculates originally contracted fund amounts and transfers them via Cashier contract.
6. If Consumer challenges before the end of the challenge period then the Aggregator performs off-chain arbitration.

7. If the challenge is not upheld, the process continues as in step 5. If the challenge is upheld, it is followed by step 8.
8. Aggregator registers outcome of challenge onchain by setting the *r-value* between 0 (redemption promise not delivered) and 1 (promise delivered).
9. Cashier contract calculates and transfers fund amounts.
10. Voucher Token is finally burned when the challenge period expires or the challenge is resolved.
11. Consumer can optionally rate the voucher after the redemption or when potential challenge is resolved.

6.4 Funds

In order to give the system maximum security, the management of funds is defined in detail by the Terms Contract and controlled by the Voucher Kernel:

- input funds are in accordance with input code, e.g. Consumer buys the voucher and funds are transferred into escrow until redemption;
- participants commit to smart contracted terms, e.g. by listing their fees;
- commitment to the authority of the Aggregator to arbitrate a challenge;
- receive output funds in accordance with output code.

Redeemeum has a defined schema for the escrow and distribution of input funds. Alternative schemes can be defined and linked to the Voucher Kernel.

See detailed description in *Appendix - Details of Funds Distribution*.

7 Business & Token Model

When designing complex adaptive systems such as economies[24] and token models, it is beneficial for designers to be able to draw upon the results of previous experiments. Results from early token experiments, such as 0x' failed governance token experiment and subsequent proposal to move to a stake-based token model[25] , are only just beginning to augment cryptoeconomic theory with empirical evidence. Therefore, we intend to wait for more evidence before designing the details of Redeemeum's token model.

Further, as start-ups are *"organisations in search of a repeatable and scalable business model"*[26], they benefit from being simple and agile. Implementing a token model now would add complexity and rigidity to Redeemeum Protocol. Therefore we intend to design Redeemeum's token model after we have confirmed our business model.

What follows then, is our initial hypothesis and assumptions for our business model and token model.

7.1 Business model

Redeemeum employs a multi-sided tokenised business model comprising a nearly free side to drive adoption, a subsidised side to incentivize target behaviours such as data sharing and a paid side providing businesses with access to data and marketing services and applications.

7.1.1 Nearly free side - public utility protocol

The core Redeemeum protocol is a public utility which is almost free to use. Transactions incur gas fees in order to secure transactions and prevent spam. In addition, a small network fee is charged in Redeemeum's native (RDM) token and deposited into the token pool. The level of the network fee will be set by the governance mechanism and will target the point of maximum value which is projected to be above the 0% level, thus enabling protocol funding to outcompete 0% forks, but below a rent-seeking level, thus driving mass adoption.

---

[24] "Amazon.com: The Origin of Wealth: The Radical Remaking of ...." https://www.amazon.com/Origin-Wealth-Remaking-Economics-Business/dp/1422121038. Accessed 5 May. 2019.

[25] "Stake-based Liquidity Incentives · Issue #31 · 0xProject/ZEIPs · GitHub." 11 Apr. 2019, https://github.com/0xProject/ZEIPs/issues/31. Accessed 5 May. 2019.

[26] "Steve Blank What's A Startup? First Principles.." 25 Jan. 2010, https://steveblank.com/2010/01/25/whats-a-startup-first-principles/. Accessed 5 May. 2019.

7.1.3 Subsidized side - Reward data sharing

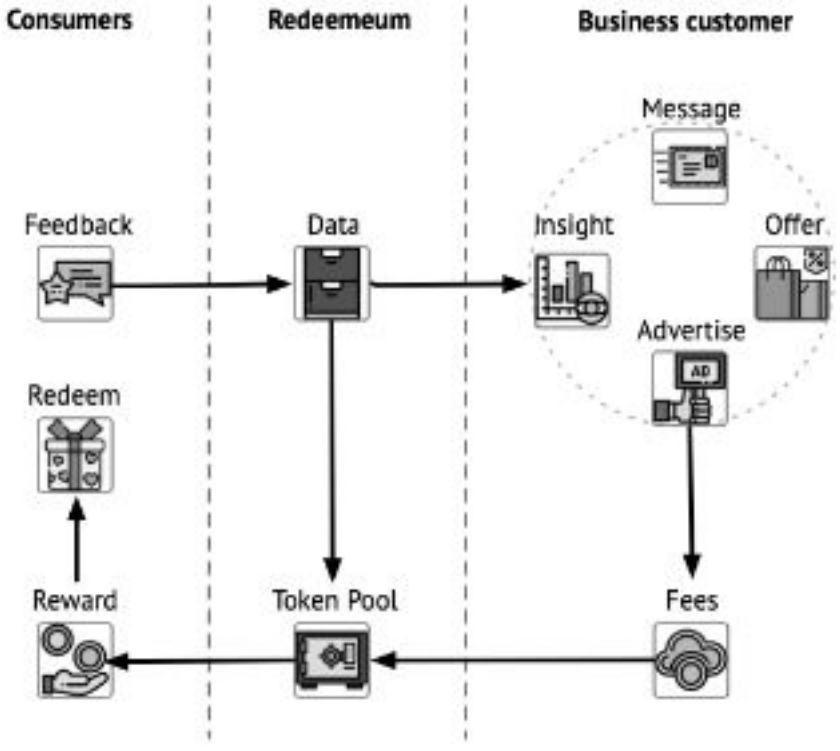
A tokenised rewards program aligns the incentives of the platform and its participants, by mapping the earning of Redeemeum tokens (RDM) to target behaviours. Target behaviours include sharing data and consenting to receive messages and paid advertising. Rewards tokens earned can then be redeemed for smart vouchers within the system. At all times participants retain sovereignty of their data and participation is entirely voluntary.

7.1.4 Paid side - dapp store business model

The paid side of Redeemeum's business model provides businesses with access to data and insight and also access to customers via advertising, offers and messaging. Fees are paid into the token pool using Redeemeum's reward token RDM, and are set via the governance mechanism. A dapp store business model enables third party businesses to consume native services and charge additional fees for their own services via the RDM token.

7.1.5 Token Pool

Fees are transferred to the token pool and are then used to fund development of the platform, reward token holders and incentivise participant target behaviours via the tokenised rewards program.



Redeemeum Protocol multi-sided business model

8 APPENDIX

8.1 Detailed Voucher Specification

Let $Asset$ be claimed at the redemption of a voucher, i.e. the goods or services delivered. It is offered by the Producer Pr to the Consumer Cn. Versioning enables continuous operation of older assets to coexist with new ones. Categorization is used for narrowing the discovery of vouchers and can include prefixed namespaces for more detailed categorization. Asset can be further described with a pointer to additional information, URI (Uniform Resource Identifier).

$$Asset \equiv \{ID_{asset}, Pr, version, title, description, category, URI\}$$

where ID_{asset} is asset's identification, calculated as:

$$ID_{asset} \equiv hash(Pr, version, title)$$

Let $Promise$ be the promise of the Producer to deliver the Asset to the Consumer under programmable restrictions on redemption, collection, monetary allocations and human-readable conditions. Promises are a core construct that enable reusability across multitude of participants, while still being anchored to the same Producer.

$$Promise \equiv \{ID_{promise}, ID_{asset}, value, merchandise, conditions_{TXT}, challenge_{period}, terms\}$$

where $ID_{promise}$ is calculated as a hash of its components;

$value$ is an encoded value of the voucher, defined with the type of the voucher, the type of the representation of value which can be either amount or percentage, currency denomination, and the number of vouchers needed for redemption (zero, if the voucher can be used repeatedly and is therefore not consumed).

$$value \equiv \{type_{voucher}, type_{value}, quantity, currency, spend\}$$

where:

$$type_{voucher} \in \{exchange\ if\ 0,\ discount\ if\ 1,\ monetary\ if\ 2\}$$
$$type_{value} \in \{fixed\ if\ 0,\ ratio\ if\ 1 \wedge type_{voucher} = discount\}$$
$$quantity \in \{amount\ if\ type_{value} = 0,\ percentage\ if\ type_{value} = 1\}$$
$$spend \in \{n_{consume}\ if\ > 0,\ 0_{present}\ if\ 0\}$$

$merchandise$ is an optional, collector-specific meaning of the voucher, important for the Collector's interpretation, such as a voucher identification by the Collector or a pointer to an external restrictions object. Note that if $merchandise$ is specified, then

$conditions_{TXT}$ must also be specified in order to enable understanding of restrictions in natural language.

$$merchandise \equiv ID^{Cl}_{voucher} \vee ext.restrictions^{Cl}_{voucher}$$

$challenge_{period}$ is optional. The redemption process can support a challenge by the Consumer when $challenge_{period} > 0$, in which case the Aggregator is engaged for dispute resolution.

$terms$ define the collection and allocation of funds, defined within an immutable contract, that is instantiated with parameters corresponding to a particular voucher set; terms are a separate construct, enabling common calculation templates:

$$terms \equiv \{terms_{contract}, terms_{parameters}\}$$

where the $terms_{contract}$ is deployed at a particular address, defining the calculation of input/output funds (such as the fees per participant); and $terms_{parameters}$ instantiating the terms contract with specific parameters, encoded in a predefined format.

Let $Promise_{collection}$ define the agreements between Producers and Collectors, for example, which Collectors are authorized for that voucher and their fees. If there are no restrictions, it can be omitted. The redemption process can support a challenge by the Consumer (when $challenge_{period} > 0$), in which case the Aggregator is engaged for dispute resolution. It is also used to register redemption attempts at a particular Collector.

$$Promise_{collection} \equiv \{agreement^{Cl}_{voucher}, redemption^{Cl}_{i}\}$$

Let voucher $Offer$ then wrap a Promise, adding processing details such as groupings and validity period. It is a base building block for the issuing of vouchers, uniformly addressable across all Relayers, which is required to manage the number of available voucher offers.

$$Offer \equiv \{ID_{offer}, ID_{promise}, quantity_{group}, quantity_{remaining}, Pr, validity_{period}\}$$

where:

ID_{offer} is voucher offer's identification, calculated at the time of generation as:

$$ID_{offer} \equiv hash(ID_{promise}, timestamp, validity_{period})$$

$quantity_{group}$ denotes the number of similar vouchers in the group. Default is 1, meaning that a default group of equal ("fungible") vouchers represents only one "non-fungible" voucher.

$quantity_{remaining}$ denotes the remaining number of available vouchers in the group. Default value is equal to $quantity_{group}$. It is decreased when voucher orders get filled.

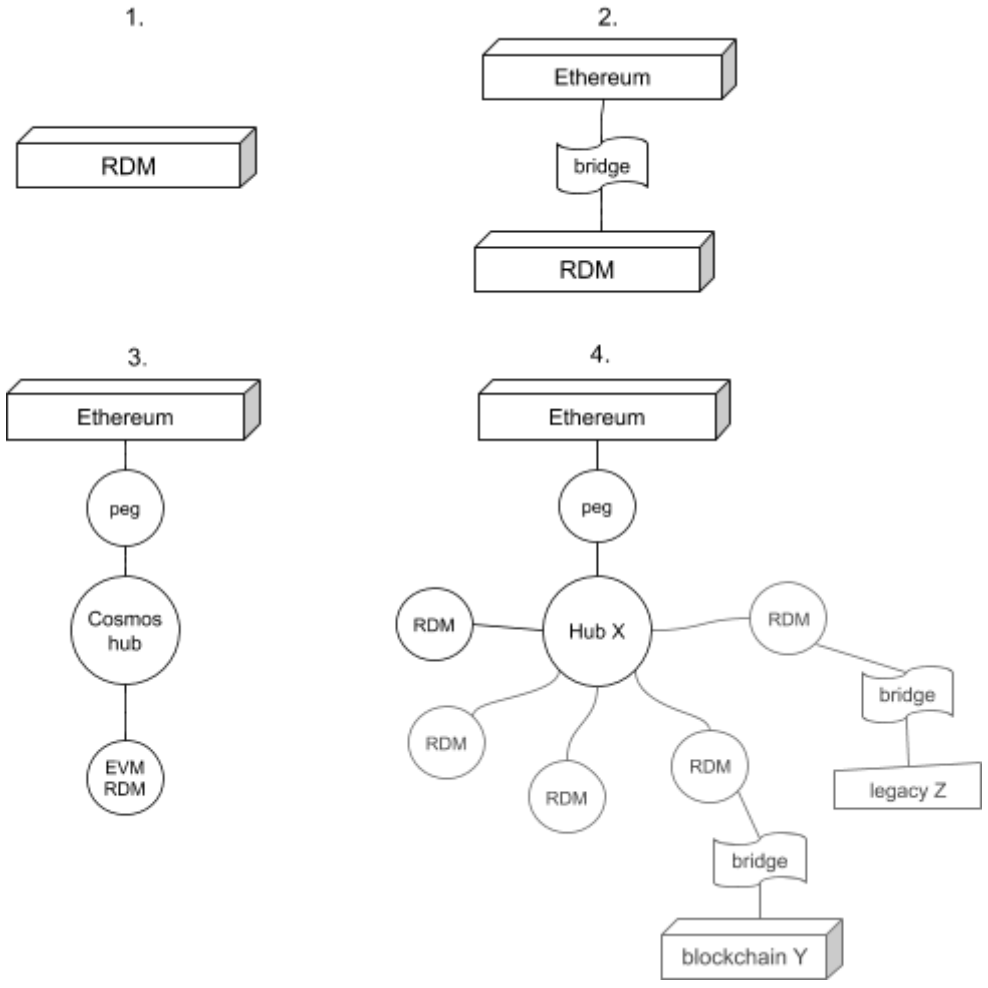
Finally, a $Voucher$ is instantiated from the $Offer$ when the offer is accepted by the Consumer, usually against payment, but not mandatory.

$Voucher \equiv \{ID_{voucher}, Cn, ID_{offer}\}$

8.2 Development roadmap

The development roadmap can be expressed in evolutionary stages, honoring the nascent nature of the ecosystem. There are several active projects exploring multi-chain operation and are closely monitored, such as Ethermint-Peggy-Ethereum, general-computation Plasmas and Polkadot. Until they mature, a trust-minimized mode is an unavoidable compromise at decentralization.

The technologies on the diagram are based on current research and publicly available information and could be replaced with better alternatives, not excluding the ultra secure Bitcoin-based sidechain later on.



Architecture roadmap of Redeemeum

Initially, a proof-of-concept is created on a private chain, developing the core constructs of the protocol. It is immediately followed by stage 2, where the previously isolated chain is connected to Ethereum mainnet via a trust-minimized cross-chain bridge. This stage takes longer to develop as it showcases the full

potential of the protocol. It is a contemporary design, found in many adjacent projects, that could support Redeemeum and grow with it for a duration.

Stage 3 is characterized as a short, intermediate step towards the long-term design, where smart contracts are taken as-is and migrated into a Cosmos realm. This is an infrastructural change: the core logic stays the same, even the Solidity code is preserved and moved into the EVM environment of Ethermint as a Redeemeum zone, connected to Cosmos hub. The link to Ethereum is migrated from a trusted bridge to a general-purpose Ethereum peg zone, Peggy.

Next stage 4 represents an optimized and scalable Redeemeum. The logic is rewritten in Golang thus making it more powerful, the hub is chosen such that supports the necessary security and performance requirements, integrations with legacy systems and other blockchains are established, e.g. using Interledger. Last, but not least, Redeemeum can scale horizontally by deploying multiple zones. Industry- or use-case- specific applications of the protocol can likewise be deployed in separate zones, all the while maintaining seamless communication between the zones and other blockchains by leveraging on Cosmos Inter-Blockchain Communication (IBC).

The majority of protocol logic will be constructed in stage 2 and so it is the focus of this paper.

8.3 Smart Contracts

The core logic is driven by smart contracts that enable a high degree of autonomy, needed for an uninterrupted and uniformly accessible public protocol. Measures are taken to evolve contracts in a graceful way and to allow intervening in the automatic operation via a chosen governance mechanism.

Asset Registry serves to provide the transactional base. Vouchers can be issued multiple times for the same asset. Versioning enables updating the asset details as its properties change. Example: rent-a-car package, a bar of chocolate ...

Voucher Kernel controls the use of the voucher, such as the issuance of the token, possible transfers between Consumers and final redemption. Voucher Kernel covers core logic:

- managing records of available vouchers (voucher orders, groups etc.)
- minting of voucher tokens
- map between Term contracts and voucher tokens
- routing payments and fees between actors
- routing metadata
- arbitration in case of disputes
- ratings by Consumers

Terms Contracts are recording the financial terms and conditions of vouchers. Being in a separate contract, they can be reused for multiple vouchers and also making it easier to shield potentially private data. Financial terms for vouchers generally conform to a common format, with flexible price/costs variables, starting from the simplest zero-fee tokens awarded to consumers directly, to more complex value-chain covering reselling and disputes management. They can also include collateralization agreements.

Collection Contract is used for managing the collection and redemption of vouchers. It defines the agreements between Producers and Collectors (authorizations and fees), which are verified during the redemption. It is used for registering redemption events.

Cashier Contract is managing the funds and is minimalistic for security reasons. It can accept funds as inputs from participants as per the Terms Contract, hold them in escrow and release them to corresponding actors after the redemption.

Transfer Proxy is an intermediary contract managing approvals of fund transfers in any currency. It is a separate contract in order to support upgrades of the protocol logic without requiring additional re-negotiating.

Oracle Contract enables interactions between blockchain and the external world. Some examples include: intercepting fiat payments, fetching currency rates, etc.

8.4 Operational Considerations

The nature of any decentralized project bears the risks that must be considered in advance. Just as the use of blockchain technology provides a more resilient security through Object-Capability model, rather than the traditional Access Control Lists, problematic for being concentrated targets for hackers, its immutable nature limits the options of responding to exploits. Identifying and addressing attacks and other risks is therefore a crucial part of blockchain-based system design and will need to be constantly updated.

Block generators

The operation of the Redeemeum blockchain depends on block generators, that will initially be known in advance, thus a good consensus candidate is Proof-of-Authority (POA), such as Clique for go-ethereum. Other, more advanced options are considered, such as Honey Badger BFT and Istanbul-BFT. A high-performance algorithm like Clique that favours availability over consistency is a fitting starting point, because malicious and unreliable block-sealing nodes are exposed via identity couplings[27].

[27] "PBFT vs Proof-of-Authority: Applying the CAP Theorem to Permissioned Blockchain - Italian Conference on Cyber Security." 6 Feb. 2018, https://eprints.soton.ac.uk/415083/2/itasec18_main.pdf. Accessed 15 Feb. 2019.

Securing nodes
As the network will be public, there needs to be appropriate security model in place for the node operators, such as sentry nodes that are protecting full nodes and unikernel-based design, that is both lightweight to maintain and has minimal security footprint. Unikernels, such as MirageOS and UniK provide an immutable infrastructure for node operators, which is often under-appreciated in the blockchain space. As the protocol and its surrounding ecosystem matures, an industry consortium could potentially play an important role in securing the Redeemeum blockchain.

Authorization
Redeemeum Protocol can only be enforced in the on-chain part through smart contracts. The suggested standard communication through APIs and Redeemeum SDK is not compulsory, as smart contracts automatically check the origin of transactions by verifying digital signatures and transaction data conforming to the rules.

Authentication
Authentication of participants is generally still a pain point. Blockchain-based systems rely on public-key cryptography, where the safety of private keys plays a crucial role and is inevitably a target for attackers. Redeemeum being a protocol-layer endeavor, its keepers will mostly rely on automated processing without manual interactions with the network. Best efforts will be made to grow awareness and modernize access credentials, such as strong first factor authentication or hardware-based multi-factor authentication.

Contingency procedures
Critical smart contracts can be paused and later resumed to respond to emergency situations. Details depend on the governance model in place. Redeemeum will facilitate a predefined way for emergency communications between affected participants, minimizing the uncertainty of such events.

Code audits
Redeemeum is encouraging a wider community to participate in its development and maintenance. An attack where malicious code is submitted to the project's source-code repository needs careful consideration[28]. All contracts must undergo security audits by reputable specialists, no exceptions. Contracts that are involved in the management of funds are maximally restricted in their scope.
Copycats

[28] "Enterprise Ethereum Alliance Client Specification v3" https://entethalliance.github.io/client-spec/spec.html . Accessed 15 Feb. 2019.

The defense against parasitic contracts rests in the minimal operating fees of the protocol and its valuable state[29] (funds in escrow, network effects, and off-chain integrations).

8.5 Details of Funds Distribution

We define the following variables:

- I_j is the input funds for participant j
- O_j is the output funds for participant j
- D_j is the decimal fraction of net voucher price that transfers to output funds for participant j, where $D_j = O_j / V_{NP}$
- F_j is the flat fee that transfers to output funds for participant j
- r is the redemption coefficient r, which represents the degree to which the promise has been delivered
- G_F is the total gas fees*
- V_P is the voucher price

Several methods are provided, such as:

- sending the funds into escrow, registering the input accordingly;
- optional routing of Consumer's actions to the blockchain-connected proxy that pays for technological fees, e.g. gas paid by the Producer on behalf of the Consumer
- calculating the outputs per participant when releasing the funds.

The following table describes a simple example where:

Inputs:

- Input code
  - Consumer Inputs voucher price, so $I_{Cn} = V_P = 100$
  - For all other participants Input is zero, $I_j = 0$

Redemption

- Redemption is completed but a challenge is upheld, with r = 0.5

Outputs

- Output code:
  - Any surplus funds are returned to Consumer, so
    $$O_{Cn} = V_P - O_{Pr} - O_{Ag} - O_{Re}$$
  - All other participants take a fraction of funds: $O_j = r * D_j * V_P$

- Output values:
  - Producer receives 92% of funds
  - Aggregator receives 5% funds
  - Relayer receives 3% funds

[29] "On Value Capture at Layers 1 and 2 - Multicoin Capital." 14 Mar. 2019. https://multicoin.capital/2019/03/14/on-value-capture-at-layers-1-and-2/ . Accessed 26 Apr. 2019.

- As r = 0.5 above funds are proportionately reduced
- All other participants receive 0% funds

So:

D_j	**Value**
D_{Pr}	0.92
D_{Ag}	0.05
D_{Re}	0.03
D_{Cn}	0
D_{Cl}	0
D_{3P}	0

**Note*: technological fees, such as gas on Ethereum main network, are excluded from the example for simplicity. In real scenario, they could be covered by the keepers, in order to make the user experience smoother.

Participant	**Input Code**	**Input values**	**Output Code**	**Output values**
Producer (Pr)	$I_{Pr} = 0$	0	$O_{Pr} = r * D_{Pr} * V_P$ $O_{Pr} = 1 * 0.92 * 100$	46
Aggregator (Ag)	$I_{Ag} = 0$	0	$O_{Ag} = r * D_{Ag} * V_P$ $O_{Ag} = 1 * 0.05 * 100$	2.5
Relayer (Re)	$I_{Re} = 0$	0	$O_{Re} = r * D_{Re} * V_P$ $O_{Re} = 1 * 0.03 * 100$	1.5
Consumer (Cn)	$I_{Cn} = V_P$ $I_{Cn} = 100$	100	$O_{Cn} = V_P - O_{Pr} - O_{Ag} - O_{Re}$ $O_{Cn} = 0$	50
Collector (Cl)	$I_{Cl} = 0$	0	$O_{Cl} = 0$	-
Third Party (3P)	$I_{3P} = 0$	0	$O_{3P} = 0$	-
Gas fees (G_F)*				
	Total Input (I_{Tot})	**100**	**Total Output (O_{Tot})**	**100**