TokenScript design paper

Author's Note: Tokens and Tokenisation in Focus

The blockchain speculation frenzy of 2017 - 2018 shone a glaring spotlight on crypto tokens. Amidst the tumult of buying and selling, we seemed to lose sight of the fact that these crypto technologies were intended for practical use by its founding designers, much like forgetting that houses, amidst a housing bubble, are fundamentally places to live, not just speculative assets.

Blockchain, at its core, serves as a trusted third party. However, to fully appreciate its role, we must delve deeper into its relevance to the global economy and the next-generation web. As technical experts who have dedicated years to researching and experimenting with its applications across financial institutions and startups, we've discerned that blockchain, in its capacity as a trusted third party, fulfills two primary functions:

1. It facilitates a frictionless market.
2. It integrates the web.

Despite the misadventures of 2017-2018, the initial emphasis on tokens wasn't misplaced. Tokens are the enablers of these two primary functions. We term the technique of harnessing this potential as "Tokenisation".

Tokenised rights, when traded on the market and integrated across systems, foster a seamless market and enable boundless integration. Now, imagine if these tokens were not just passive carriers of value, but could actively interact with their environment, make decisions, and adapt to changing circumstances. This is the concept of Smart Tokens, a new layer of sophistication added to the basic token model.

While previous efforts in this field have largely concentrated on enhancing technical aspects such as transaction throughput, our project shifts the focus to tokenisation and the potential of Smart Tokens, which operate in the functional dimension. We introduce a standardisation effort known as TokenScript, which complements the blockchain technical stack and provides utility for the economy and the next-generation web.

The work on TokenScript is ongoing at tokenscript.org and we are looking to standardise it through a standardisation organisation. We invite you to follow our progress closely and join us on this enlightening journey!

Contents

Author's Note: Tokens and Tokenisation in Focus....................iii

Chapter 1: Introduction: The Potential of Blockchain.................... 6
- Creating a Frictionless Market.................... 7
- What can be tokenised?.................... 7
- Payment Token and Deliverable Token.................... 7
- Token properties.................... 8
- Smart Tokens: A New Approach to Tokenisation.................... 8

Chapter 2: The problem of the Internet as we know it.................... 10
- The client side can't integrate a web that is not designed to integrate.................... 11
- Web integration goes beyond account authentication.................... 11
  - "Account authentication" is not a substitute for web integration.................... 11
  - The web lacks a built-in ownership mechanism.................... 12
- The Car Ownership Token: A Case Study in Smart Token Integration.................... 13

Chapter 3: Smart Tokens: A New Paradigm for Next-Generation Web.................... 16
- The Complexities of Tokenisation.................... 17
- The Limitations of Traditional Tokens.................... 17

Chapter 4: Addressing Frictionless Market needs.................... 20
- Deliverable Example: 1% Property Token with Smart Tokens.................... 22
  - Product Description.................... 22
  - Attested Information.................... 23
  - Reference Information.................... 23
  - Action Information.................... 24
- Payment side example: Intelligent Currency.................... 24
  - Challenges on the payment side.................... 25
- Benefits of Smart Tokens.................... 25
  - Delivery Side.................... 26
  - Payment Side.................... 27

Chapter 5: Address the "Integrate the web" need.................... 30
- Acquiring an iPhone.................... 31
  - The Shipping Smart Token.................... 31
  - Warranty Smart Token.................... 31
  - Receipt Smart Token.................... 31
  - The Ownership Smart Token.................... 31
- A Future Iteration of AirBNB.................... 34
  - Requirements.................... 35

Chapter 6: The Architecture of TokenScript with Smart Tokens.................... 38
- Evolution in Business Processes and Market Dynamics.................... 39
  - Implications for the Design of TokenScript: Messaging in the Context of Smart Tokens.................... 40
- Types of Tokens in the Context of Smart Tokens.................... 41

Attestations........41
The Components of TokenScript........42
Actions........42
Magic Links........43
Attestations........43
Assets........43
Conclusion........44

Chapter 1

Introduction: The Potential of Blockchain

Topics:

- Creating a Frictionless Market
- What can be tokenised?
- Payment Token and Deliverable Token
- Token properties
- Smart Tokens: A New Approach to Tokenisation

Blockchain technology serves two primary functions that are essential for the future economy and the next-generation web:

- Facilitating a frictionless market; and
- Integrating the next-generation web.

This paper addresses the vision of where we can be and follows up with the design and reasoning behind the architecture needed on top of the blockchain. We explain TokenScript, a critical missing layer, and go over its design principles and how we are building it.

Creating a Frictionless Market

The 80s' "Back to the Future" featured a world of powerful machines filled with hovering boards and flying cars. It didn't happen. As Peter Thiel once famously lamented, "we were promised flying cars; instead, we got 140 characters". The technological advancement of our time is beyond the imagination of the 80s science fiction movies, albeit not through more powerful machinery, but efficient use of the Internet.

Ride-sharing revolutionised the way we organise our daily lives, and AirBNB changed the way we travel. These are the new, less frictional markets. They incur less cost to operate, are more accessible and have finer operational units.

However, despite this web 2.0 revolution, the majority of markets still operate with high costs. The stock market, for example, has so much overhead that it is only justifiable for multi-million dollar businesses to afford to operate under the rules and regulations.

With blockchain, any tokenised asset can be transacted swiftly at any time, as long as it follows the rules, without an intermediary or the process to "enter the market", eliminating frictions and enabling maximum market efficiency.

With the traditional intermediary-operated market model, a trade is made in two stages: entering the market, making a deal. Blockchain can simplify that into a protocol; therefore the blockchain token assets can be considered always on the market.

What can be tokenised?

Could we tokenise a house, enabling its automation systems to recognise friends and guests by connecting to social networks, or allowing the owner to apply for insurance, buy services, and more?

Could we tokenise electricity, allowing power users to benefit from finer scheduling of the use of resources, and households to benefit from collecting surplus sun energy?

Could we tokenise AirBNB bookings, so that hosts can purchase a guaranteed cash flow from the market, while speculators profit from predicting the travel needs?

Could we tokenise books, creating a new paradigm where authors could issue tokens representing ownership or access rights to their works, enabling them to directly monetise their intellectual property and readers to trade or sell their purchased books in a secondary market?

Could we create an insurance token that depends on cryptographic proofs, so that the insurer can remove the costs incurred by fraudulent documents from the pricing? Could we go even further, so a car insurance token connects the car's operating system and allow the insurance company to connect to a road side assistance company with latest car data such as location and self-diagnosis? Could we auto-compensate the user through that token?

Blockchain can provide the foundational layer to achieve these. It enables a working, frictionless market with tokenised assets always on the market. However, this can only become true when there is a reliable and precise method to define how tokens should be used and transacted. This is the focus of our work on TokenScript.

To carve out the difference, we look at how Tokens are used today.

Payment Token and Deliverable Token

In 2017-2018 we did end up having hundreds of tokens. However, they uniformly fall into one category of token: created with the ERC20 standard they are currency-like, filling up the payment side of the market. There is nearly zero effort devoted to making tokens goods and services - which is the deliverable side of the market and a fundamental need for a market to work.

We categorise tokens as payment tokens and deliverable tokens. ERC20 tokens bearing the hallmarks of payment tokens only fills one side of the market with tokens. They can't lift the market, as they merely compete with other payment-token - like Bitcoin or Ether - on the payment side. They represent a good, but they do not actually deliver. They are rather like gift cards.

During the speculative bubble of 2017, an energy token ICO did not need to provide any explanation of how the tokens can be used. All speculators needed to know is that they represent for example a "stake in the future world of tokenised electricity". As long as the token can inspire investors with imagination, it's good enough for an ICO. There is no more functionality needed other than an ERC20 interface.

Such a speculative token didn't depend on attestations - the proof of actual power production - nor did it need properties like where the energy is provided or for how long it is available. Instead of enabling a frictionless market, those tokens have just been a promise on enabling it, which has been sold as a gift card. By far most will never deliver.

Now that the madness is over, it's time to present the technical framework to make the market actually work.

Token properties

Tokens can be products. Therefore, they need to have different properties: Do tokens expire? AirBNB booking tokens certainly do, but Graduation Diploma tokens probably don't. Should the token owner receive a notification on a specific event? An energy token needs that, for the change in the power supply is dynamic. Is a token stream-able?

How does it look on the user's mobile, and how is it called in a users language? If a buyer wants to purchasea tokenised country estate from a seller, how do they establish a trusted method of communication? If a token entitles the user to do specific actions online, how can the user login to the web services with that token?

It's easy to see the need for an open framework defining tokens and making them interoperable with different methods of trading, listing and rating. TokenScript provides such a framework. It overcomes the limitation of the approach to put everything in a smart contract or a set of smart contracts.

However, to diligently design it we need to consider the second part of what Blockchains can do: Integrate the next-generation web.

Smart Tokens: A New Approach to Tokenisation

Smart tokens introduce a new perspective to tokenisation. Unlike traditional tokens, smart tokens are programmable, with built-in rules and logic that govern their behaviour. This programmability allows for a more dynamic and responsive market, where tokens can adapt to changing conditions, enforce compliance, and automate complex interactions.

Smart tokens can represent a wide range of assets and rights, from tangible assets like real estate to intangible ones like intellectual property. They can also represent a set of conditions or an agreement, such as a contract or a promise. This flexibility and versatility make smart tokens a powerful tool for creating a frictionless market.

However, to fully realise the potential of smart tokens, we need a robust and flexible framework for defining, issuing, and managing them. TokenScript serves this purpose. It is a language and a set of standards for smart token design. It provides a way to define the behaviour and interactions of smart tokens, making them interoperable across different systems and platforms.

By providing a standardised way to define and interact with smart tokens, TokenScript can help facilitate the full potential of the next-generation web, where every asset, every right, every product and service can be tokenised, traded, and integrated into the web in a seamless and frictionless way.

Chapter 2

The problem of the Internet as we know it

Topics:

- The client side can't integrate a web that is not designed to integrate
- Web integration goes beyond account authentication
- The Car Ownership Token: A Case Study in Smart Token Integration

The internet, as we understand it today, was primarily shaped by two models: a) the public library model, and b) the computer-human interaction model, both of which were championed by pioneers like Tim Berners-Lee.

a. The library model, which is the foundation of the internet, provides free access to information, indexed and cross-referenced by a URI. Its digital equivalent, the URL, signifies the location of data, with no restrictions on accessibility.

b. The computer-human interaction model is a dialogue between two participants - the human user and the machine. While a computer's knowledge is limited, it can guide the user to the appropriate machine.

As a result, the internet was designed as a vast library, where each book is a computer capable of engaging in a conversation.

However, this design has led to numerous contemporary challenges. Let's consider two examples:

Flight Delays and Hotel Rebooking: A user has a flight and a hotel booking. The flight gets delayed, and now the user has to manually contact the hotel to adjust their reservation. This process is not only time-consuming but also adds unnecessary stress to the user's travel experience.

Health Data Management: A user wants to access her health data stored across various healthcare platforms. She needs to remember multiple login credentials, navigate different user interfaces, and manually compile the data. This fragmented and disjointed experience makes it difficult for her to manage and make the most of her health data.

The question arises: why are we, as users, performing tasks that machines excel at? The current design of the internet, akin to a giant library, forces us to keep track of index numbers, much like readers jotting down notes. The vision for the future is to transform the internet from a library-like structure into a more personal assistant-like entity.

The client side can't integrate a web that is not designed to integrate

The pioneers of the web, such as Tim Berners-Lee, envisioned a digital landscape where information is freely accessible and interactions are seamless. However, the reality of the web as we know it today is far from this vision. The current web design, despite its advancements, is fundamentally not designed for integration.

This lack of integration is evident even in technologies that were intended to serve as personal assistants, such as smartphones. Despite their capabilities, smartphones have not been able to fully integrate the web. The reason is simple: the web, in its current form, is not designed for integration. This issue is further exacerbated on mobile devices where tasks such as copy-pasting become even more cumbersome.

The truth is, integration requires more than just client-side efforts. The infrastructure itself needs to support integration. A smartphone, in its current form, is essentially a miniature version of the dial-up Internet, with each app representing a website. Users still need to figure out which app to use for a specific task, and they still need to manually transfer information between apps. For instance, it's currently not possible to ask your smartphone to sum up all the money you can access through your various online banking apps.

The root cause of these inconveniences is the lack of integration in the web. Here are a few examples that illustrate this issue:

Consider the scenario of a traveler whose flight has been delayed. The traveler now has to manually adjust their hotel booking, car rental, and other related services. Each of these services is provided by a different entity, and there is no integration between them. The traveler has to manually navigate between different apps or websites, adjust their bookings, and ensure that everything aligns with the new flight schedule.

Another example can be seen in the healthcare sector. A patient receives a health token that contains their medical history, prescriptions, and other health-related data. However, this token can't be used directly to order medications from a pharmacy or to schedule appointments with a doctor. The patient has to manually transfer the relevant information from their health token to the pharmacy's or doctor's system.

These examples highlight the need for backend integration between different service providers. However, such integration is not often done due to the complexities involved. It requires collaboration and negotiation between the parties involved, and it also poses challenges related to:

Security: Each integration point increases the attack surface, making the system more vulnerable to security breaches. For instance, if one system is compromised, the attacker could potentially gain access to all connected systems.

Availability: The availability of an integrated system is not just about whether it's online or offline. It also involves how efficiently and accurately it can provide the necessary functions and data for integration.

Scalability: The complexity of integration increases exponentially with the number of parties involved. For n parties, there would be n^2-n integrations needed, which is a significant challenge. This complexity not only hinders the process of integration but also stifles competition and limits the freedom of the market.

The challenge of integrating the web requires several building blocks that were not part of the original blueprint of the web. These include authentication, ownership, transfer of value, and trading. We will see in the later chapters the introduction of smart tokens provides a promising solution to these challenges.

Web integration goes beyond account authentication

"Account authentication" is not a substitute for web integration.

The web, as it stands, lacks an inherent authentication mechanism. Many users resort to third-party solutions like "Sign in with Facebook" to circumvent this issue. However, these solutions only offer authentication through a trusted

[tls] Efforts have been made to establish client/server certificates in Transport Layer Security (TLS). However, these methods are not designed for processes, but for sites. They follow a delegation model, which is akin to a buyer verifying the seller's identity against the name on a title deed, without checking the authenticity of the deed itself. In this model, TLS can only confirm the authenticity of the website, not the content it hosts. For instance, Facebook uses

third party, raising concerns about privacy and availability. Moreover, they only serve for account authentication and do not facilitate integration.

Furthermore, the account-based authentication model introduces additional complications. For instance, consider a simple business scenario where a car owner wants to check the service history of their car. This process doesn't require an account. Accounts are merely a workaround for the limitations imposed by the current structure of the Internet.

Enforcing account-based authentication can lead to undesirable outcomes:

- When a car is sold, the new owner must create a new account on the service website and verify their ownership of the car. This process is cumbersome and unreliable.
- If a third party, such as a vehicle modification workshop or an insurer, needs to access the repair history, there's no straightforward way to authorize them without sharing the account details. This is inflexible.

These integration needs, which are inadequately addressed by account creation, are common in sectors like healthcare, retail, and virtually every web-based business. Currently, the solution seems to be creating more accounts to meet the growing integration needs. This approach is akin to treating every problem as a nail that needs to be hammered down. Most people are uncomfortable with this model but can't envision an alternative for web integration. The idea of an account-less Internet is beyond the imagination of most people.

We propose that token-based integration is a superior solution for integrating the Internet. A crucial component of this process is the concept of ownership.

The web lacks a built-in ownership mechanism

The web does not inherently support ownership, value transfer, and trading.

To illustrate, let's extend the car scenario: If you want to sell your car, you need to post the car information on a website, which requires creating an account. When someone wants to buy your car, you and the buyer must navigate a series of processes: Insurance, unused service quota, vehicle registrations, documentation handover, payment, and so on. All these actions must be performed separately, relying on easily tampered paper proofs, forms, and accounts. The process starts on the web but ends elsewhere, instead of being automatically completed when the buyer clicks "buy".

Is it possible to automate the entire chain of bureaucratic procedures securely in the backend, with just a click of the "buy" button? With the current web of accounts, you'd need to tie together numerous accounts and trusted third parties, which would obscure the process from the user while still following the same paper trail.

In contrast, if the same process were based on a blockchain and tokens, it would be automatic, fraud-proof, and atomic. You could securely complete a car sale with a single click, without the need for accounts and paper trails.

These missing features of the web are the well-known functions of the blockchain. A blockchain is an immutable, decentralized record of ownership, sometimes referred to as a "triple-entry bookkeeping" system. The virtual union of this perfect fit couple requires a virtual exchange of tokens, or what this paper refers to as "tokenisation".

To achieve this, tokens must seamlessly traverse systems, carrying their trading rules, user interfaces, and business context. This approach, enabled by smart tokens, allows for the integration of the next-generation web, offering a more efficient, secure, and user-friendly experience. The concept of ownership, a crucial element in this process, is redefined and enhanced through the use of smart tokens.

TLS, but it doesn't prevent the spread of fake news. The level of trust in this model is not granular enough to deliver an integrated web experience.

attestations The method of providing cryptographically signed attestations as a condition for a transaction is discussed later in the "Attestation" chapter.

atomic In blockchain terms, an atomic transaction either happens or not. If well defined, it's not impossible for a buyer to have successfully paid for a car yet not getting the ownership token, or only have transferred the car's ownership but not the compulsory insurance on it.

The Car Ownership Token: A Case Study in Smart Token Integration

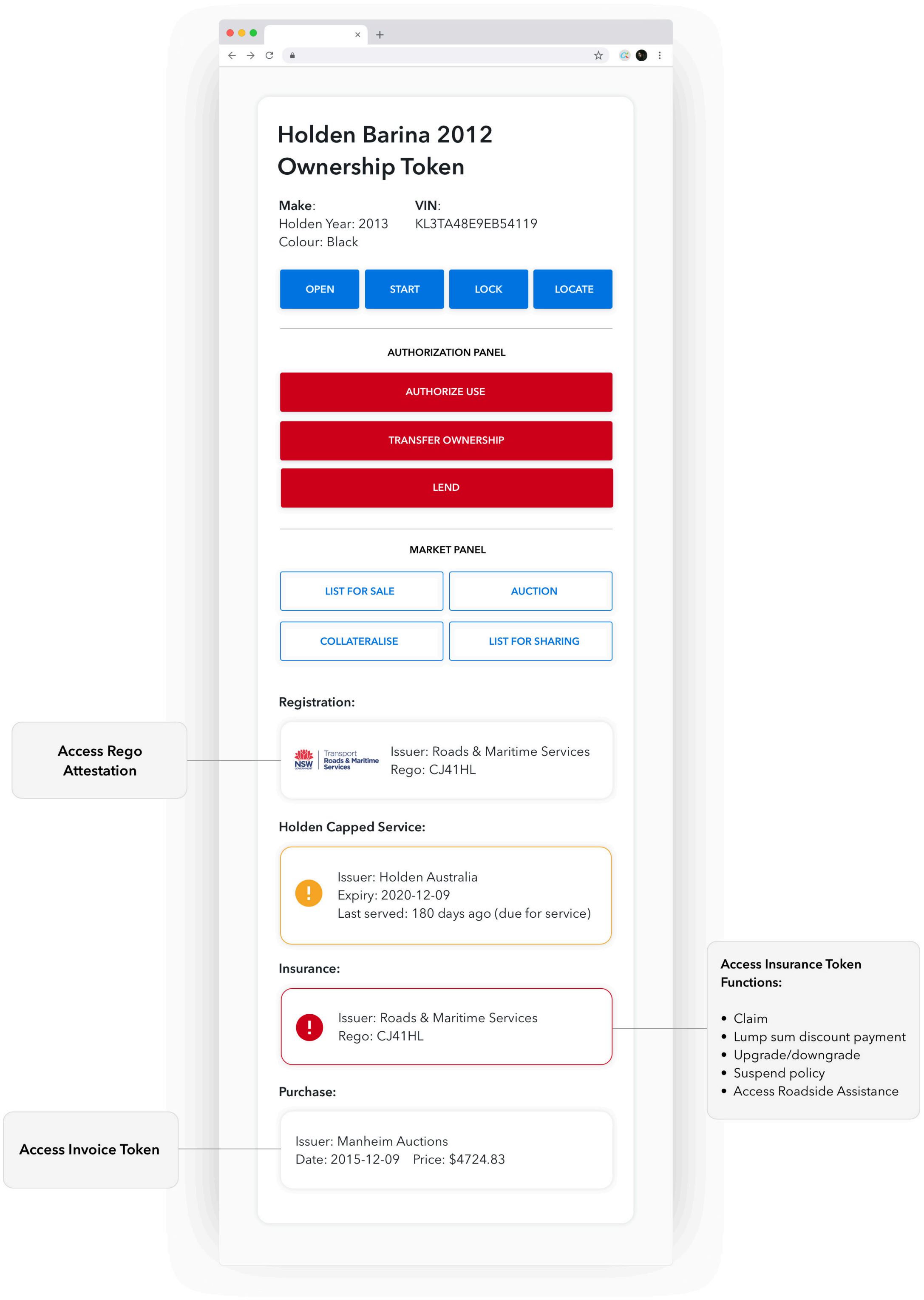
The fusion of frictionless markets through asset tokenization and web integration via smart tokens as integration points can be exemplified through the concept of a car token. This is merely one instance among a plethora of potential applications, extending to realms such as real estate markets, business-to-business transactions, and resource transfers, essentially any transaction involving digital goods or a digital representation of a physical good.

The car token serves as a practical illustration to comprehend these concepts. A car, in this context, is a tokenized asset that can be purchased, sold, transferred, auctioned, collaborated upon, and insured, all facilitated by the blockchain.

Simultaneously, a car possesses utility. A car's ownership token can transform a blockchain wallet into a car key, with additional functionalities such as visually representing the car's current location. Authorizing someone to access your car or renting it out for profit could be seamlessly executed by signing blockchain transactions or attestations, eliminating the need to physically exchange car keys.

In both scenarios, the token embodies the delivery aspect: they *are* the product. These tokens can interact with non-token forms of payment, but tokenizing the payments would render the entire process more streamlined.

The following image of a car token symbolizes the ultimate stage of tokenization.



Holden Barina 2012 Ownership Token

Make: Holden Year: 2013 Colour: Black

VIN: KL3TA48E9EB54119

Registration:

Issuer: Roads & Maritime Services
Rego: CJ41HL

Holden Capped Service:

Issuer: Holden Australia
Expiry: 2020-12-09
Last served: 180 days ago (due for service)

Insurance:

Issuer: Roads & Maritime Services
Rego: CJ41HL

Purchase:

Issuer: Manheim Auctions
Date: 2015-12-09 Price: $4724.83

Access Rego Attestation

Access Insurance Token Functions:

- Claim
- Lump sum discount payment
- Upgrade/downgrade
- Suspend policy
- Access Roadside Assistance

Access Invoice Token

At first glance, it appears to be a convenient portal to manage everything about the car, including market functions and utility. However, this is not feasible with the traditional web model. In the web 2.0 model, you are constrained to manage each element individually:

- Registering the car involves a separate process that requires creating an account with the Road and Maritime Services and manually proving ownership without the aid of cryptography.
- When you want to insure the car, you have to create another account and manually provide proof of its registration to that new service.

- Similarly, if you want to make the car available for sharing economy through Uber or hour-based car rental, the task of proving and settling payments and insurance cost adds friction to the market.

The intended portal does not enable these functions by itself but merely serves as a gateway to merge a lot of different accounts as we know it from the current web. It's just another temporary solution, which conceals

paper trail processes from the user, without addressing the underlying issue.

Now, let's envision this in the context of the next-generation web where such elements can be tokenized, step by step:

Purchasing and Registration: The Vendor (in this case, Holden) provides an ownership token to the new owner, which can be used to operate the car. The token, transferred to the owner at the time of purchase, is subsequently used to acquire the registration token. An inbuilt IoT device allows the car to be operated with proof of ownership via a token.

Insurance: The owner, desiring to purchase insurance, only needs to provide the proof of ownership and registration token to meet the requirements with the insurance company. The insurance company's standards are automatically met by matching the tokens to their requirements. Once validated, the insurance company can send the owner an insurance token in exchange for payment. The insurance token carries its own functions and services.

Smart tokens bridge the gap between different providers and services, which used to be built by accounts, trust, and paperwork.

Uber: If the owner wishes to become an Uber driver, she can easily prove her vehicle's eligibility by providing proof of ownership, insurance, and registration with her tokens. Uber then automatically provides her with an Uber token which, depending on the owner's need, can be used to start as an Uber driver or allow a third-party driver to do so. None of these processes require manual verification or account creation.

Smart tokens enable a more flexible, even programmable, use of ownership rights and their interaction, as centralized, account-based services can provide.

Self-Uber: Taking this even further, the owner can bypass Uber altogether and rent her car directly to strangers. To ensure her car isn't mishandled by some random stranger, she can restrict her renters to those who have an attestation-based token issued by the 'better drivers bureau'. The renter proves they have this token, pays a sum to the owner, and is atomically issued with a temporary token that allows them to unlock and use the car for a certain period. This is done without the creation of an account or the need to submit tons of documents to be validated manually by the owner.

Selling: If the owner wishes to sell the car, she only has to list it on any website with a price. The ownership token and payment can be swapped atomically (ensuring neither the buyer nor seller is cheated), and the new owner can drive away with the car without even meeting the original owner face-to-face. The new buyer knows in advance whether the car has been registered and is legally owned by merely validating the original owner's ownership token in their wallet. The original owner's token is invalidated once the swap occurs, and she can no longer operate the car. It is also possible to automatically void the insurance policy once the exchange has occurred and provide the original owner with a rebate for premature cancellation.

This chapter serves to present the vision. Smart tokens enable the entire ownership and utility processes around car trading and sharing to happen automatically, fraud-proof, and atomic. This eliminates a lot of friction and allows much more flexibility to individualize the economic transactions.

We will have the opportunity to inspect the technical aspect of this well-integrated, well-tokenized car token in later chapters again.

Chapter 3

Smart Tokens: A New Paradigm for Next-Generation Web

Topics:

- The Complexities of Tokenisation
- The Limitations of Traditional Tokens

Tokenization, the process of converting rights to an asset into a digital token on a blockchain, holds immense potential for the next-generation web. However, the complexity of tokenization, with its requirements for richness, embeddedness, flexibility, and trust, poses significant challenges.

Legacy approaches to tokenization, prevalent during the ICO boom, were primarily payment replacements, failing to fully exploit the potential of tokens. These models were rigid, complex, and lacked interoperability, limiting their ability to create a frictionless market and integrate digital services.

The advent of Smart Tokens heralds a new paradigm. These tokens are not just sophisticated; they are adaptable, capable of interacting with various systems, and can carry trust relationships to third parties. They offer a solution to the shortcomings of the legacy model, paving the way for a more integrated and decentralized web. This chapter delves into the transformative power of Smart Tokens and their potential to reshape the next-generation web.

The Complexities of Tokenisation

The illustration of car ownership underscores the potential of tokenisation to establish seamless markets and integrate digital services. This is a novel paradigm for many markets. However, the current usage of tokens, as evidenced by the ICO frenzy of 2017/18, is far from delivering what we define as tokenisation. Most tokens merely serve as a substitute for the payment side, without attempting to be more.

To realise their potential, tokens need to become significantly more sophisticated and cater to the delivery side. A token protocol must meet several requirements:

Diversity

Tokenisation necessitates a vast array of tokens, each customised for its specific use case. This demands bundling the tokens with their *transaction rules* and *behaviour patterns*. New tokens should be able to join the ecosystem on an abstracted layer, enabling them to be traded and used in *various contexts*. With the expected proliferation of new plasma subnets, tokens should also be capable of operating seamlessly on them.

Integration

Tokenisation must enable users to interact with different systems through the tokens. In the car example, the car token contains code to interact with a *smart lock* (the *Open*, *Start*, *Lock* actions) and the maker's own *web service* (the *Locate* action). The *List for sharing* is provided by *another third-party service* that tokenises the usage of the car by hours or days and sells them piecemeal. The token must be embedded in different environments and used by different services - while the owner must be able to access all these markets solely through this Token.

A token must also be renderable and associated with the actions it can perform in the user's wallet. In the car example, if the registration expired, the web component at work would paint the Registration Token red or display a warning. Actions like *List for sharing* will not be available with an expired car rego, and the integrated token interface should clearly pass that message to the user. Tokens must be rendered differently according to what happened to them in the user's wallet. Tokenisation requires the wallets to correctly react to a wide variety of token events.

Adaptability

It must allow new protocols to be developed on tokens. A token never has a finished state. There are always options to attach new protocols to it. In the property example, collateralisation might be something desirable to add later, as well as identity information or the ability to transfer the token through plasma state channels. This has to reflect in the user interface, thus there must be a way to deploy trusted code to the user-agent's wallet or preferred dapp.

Trust

A token must carry trust relationship and business context to 3rd parties. In the car example, the insurance token provides Roadside Assistance service through NRMA. The driver might be able to access this through the token of his insurance provider and immediately be identified as qualified for help. In both examples, the token must carry trust relationships, which shouldn't depend on the availability of a certain service, but passed directly by the token. Both business context and the relationship must be part of the token, while being highly available, private and integrative.

The Limitations of Traditional Tokens

Early public blockchain projects attempted to implement both token logic and business process into smart contracts. Using an online retail project as an example, such a smart contract would not only process an order but also manage the inventory. The token transaction logic, like under what condition the transaction is valid, is tied with business process, like checking inventory. This method is, naturally, inherited from the way people build websites.

We argue that this method is not suitable for creating a frictionless market and integrating

the next-generation web. Fulfilling the challenges with the conventional token model is difficult, often nearly impossible, while it adds complexity and causing scalability, interoperability and security issues:

Diversity

In the world of Ethereum - the de facto standard for token - diversity of Token is usually created with DApps: The business logic of a token - all kind of applications - are coded in a smart contract, and centralized websites enable users to access the contract. For example, you have an ERC721 crypto kitty token. To use it you must access the cryptokitties website.
This method requires the designers to fetch all possible business scenarios before creating the smart contract. As the future is rarely predictable, this method is problematic. It makes it also hard to develop a business model by trial and error. At the same time, it adds a lot of complexity to the code, which often causes security issues. After the DAO was hacked, the Ethereum community restricted itself to only implement relatively limited behavior patterns in smart contracts.
So the result is that Token developer need to restrict the scope of diversity, but still add security problems.

Integration

Ethereum token have a very limited way of interacting with other systems like wallets or dapps. If the logic of interaction is part of the smart contract, we have the problems explained above. *and* must deal with the fact that you hardly can represent all systems language in one contract. It is impossible to do this without the help of external frameworks.
Currently this is mostly solved by hosted DApps on websites, which structure the interaction between users and smart contracts. Such DApps have made it possible for users to access relatively advanced smart contracts like CryptoKitties and other token based games. However, this reintroduces the centralization problems Blockchain was made to solve. We will address the problems of hosted DApps soon.
A similar problem is the integration in a wallet. Events in the token history must trigger actions in the user interface. It is hard to do this, when the smart contract doesn't define the behaviour of all those systems in a way they understand. It's also hard to do so when a contract is upgraded. The usual solution is, again, to use a hosted DApp instead of a wallet under the user's control.

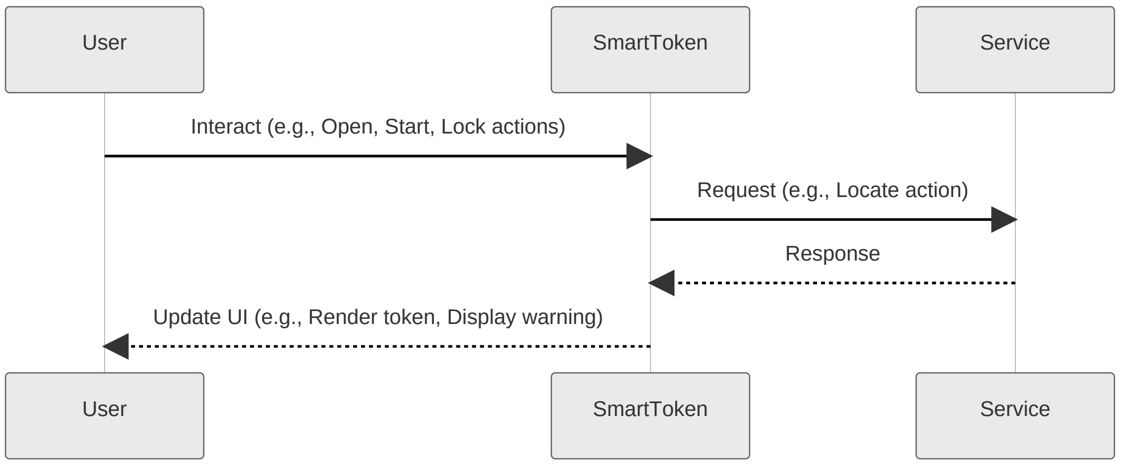
Adaptability

The inflexibility and immutability of a smart contract tokens makes it hard to develop new protocols for it, especially when those protocols are not known when the contract is written. You will also need your smart contract to interact with other smart contracts in a way you can't predict when designing it.
This could end with a locked-in-state of the token on one certain protocol. It could also introduce trusted third parties to migrate to other protocols or to interact with other smart contracts or tokens.
On Ethereum, there are methods to upgrade a smart contract. They usually add a lot of complexity to the original smart contract and introduce another smart contracts which performs the upgrade onchain. This approach increases the complexity of the smart contract system, which causes security issues. An example is the multisig contract of Parity, one of the most advanced implementations of Ethereum. Even their developers failed to make the contract secure, resulting in one of the largest losses in the history of Ethereum.

Trust

To be used in a wide scope of business activities, Tokens must carry trust relationships. With the legacy token model this casts two problems: First, you will have to input private data on a blockchain, which has, even when encrypted, several risks. Second, you need to carry the relationship over a hosted DApp, which means you are dependent on a website being online. If one part of a chain of trust relationships is offline, your token will

Here's a simplified sequence diagram illustrating the interaction between a user, a smart token, and a service:


User
SmartToken
Service
Interact (e.g., Open, Start, Lock actions)
Request (e.g., Locate action)
Response
Update UI (e.g., Render token, Display warning)
User
SmartToken
Service

Chapter 4

Addressing Frictionless Market needs

Topics:

- Deliverable Example: 1% Property Token with Smart Tokens
- Payment side example: Intelligent Currency
- Benefits of Smart Tokens

To understand the requirements for Smart Tokens to create a frictionless market, we need to define markets and the concepts involved. In this context, *a market is a place where delivery versus payment* occurs. The role of Blockchain, enhanced by Smart Tokens, is to reduce reliance on intermediaries and eliminate frictions in markets.

To comprehend the design considerations for Smart Tokens, we first need to establish a clear understanding of fundamental market concepts, including markets, marketplaces, payments, and deliverables. We'll then explore a real estate example to illustrate the capabilities required of a token markup language.

For clarity, we define the three concepts involved.

Deliverables	These are all sorts of things that money can buy: assets, goods, and services. In this context, deliverables are not physical goods, but entitlements. A proto-tokenization of deliverables happened long ago, like land or security titles. However, only with blockchain and the advent of Smart Tokens, they can be transferred in real-time without needing a third party to prevent double spends. Deliverables can greatly benefit from tokenization - if the token design properly reflects their needs.
Payments	All value transfers done with anything currency-like. In traditional markets, payments are done with Fiat currencies, either as physical banknotes or as electronic transfers through trusted third parties. Blockchain, enhanced by Smart Tokens, eliminates the intermediaries from payments and enables the use of programmable currencies like Ether or Dai.
Market	This is a concept of individuals exchanging payments versus deliverables. It is neither meant as a

single marketplace nor the entirety of all marketplaces. To access a market, users can, but do not have to be on a marketplace like Amazon.

In traditional markets, both *the deliverable* and *the payment* side tokens must "plug-in" to the *market*. The market participant must transfer both parts of a deal to the market, which adds friction and introduces intermediaries. The promise of Smart Tokens is that both deliverable and payments are *always on the market*.

Enabling this is a main requirement for Smart Tokens. To do so, Smart Tokens enable tokens to be presented, indexed, transacted, traded, auctioned, combined and so on. We will demonstrate this requirement by both an example for delivery and payment and explain how Smart Tokens can address these needs.

Deliverable Example: 1% Property Token with Smart Tokens

Let's envision a market for "1% property". A property owner can issue multiple Smart Tokens, each representing 1% ownership of the property. These tokens can then be sold for cash. A potential buyer would need to understand a significant amount of information. It's straightforward to comprehend that such a token would yield 1% of the sales revenue if the underlying property is sold, but more details are necessary:

1. Where is the property located and what is its current status?
2. Does a 1% property token owner have voting rights? For instance, on the decision to purchase insurance against a bush fire?
3. Is the 1% automatically converted into currency at the time of property sales, or can the token holder choose to continue holding it?
4. Is the token properly underwritten to prevent double-collateralization?
5. If the property was collateralized for a mortgage, what is the condition for a liquidation event?
6. Is providing a buyer's identity attestation a condition of a purchase?
7. Is the seller the actual owner of the property?
8. What was the performance of similar properties in the region in the past years?
9. What was the historical sales price of this property?

Specific to blockchain, we also have:

1. How to correctly and securely construct a transaction for the asset (purchase, voting etc)? You can't expect any investor to be a blockchain enthusiast who knows how to load the smart contract files in his wallet.

We categorize these trade-sensitive information into four categories:

1. Product description: Item 2, 3, 5, 6
2. Attested information (attestations): Item 1, 4, 6, 7.
3. Reference information: Item 8, 9.
4. Action information (how to perform an asset action): Item 10.

Understandably, the buyers need to access all these for an informed decision. To allow tokenization of deliverables to happen, these informations must show up on his wallet when he starts interacting with the token. In the following chapters, we will describe how these informations categorizes manifest in Smart Tokens.

Product Description

Product description information is typically part of the smart contract. They can be obtained by making a few Smart Contract function calls, therefore, the only needed work is to convert them into a presentation. Usually, this means translating them to the language user speaks and converting "True" value into a nicely ticked checkbox.

This introduces the first functionality of Smart Tokens: acting as a presentation layer for smart-contracts. In XML it looks like this:

```
<attribute-type id="voting-right">
   <name xml:lang="en">Voting right</name>
   <name xml:lang="zh">###</name>
   <origin contract="holding-contract" as="mapping">
      <function name="getVotingRight">
     <inputs>
        <uint256 ref="TokenId"/>
     </inputs>
  </function>
      <mapping>
     <option key="0">
         <value xml:lang="en">No Voting Right</value>
```

[pd] The word is borrowed from the financial sector, usually used to describe packaged investment products. It means the formula which profit is calculated and the current values of the variables in the formula.

```
  <value xml:lang="zh">####</value>
         </option>
     <option key="1">
         <value xml:lang="en">Voting rights on sale</value>
  <value xml:lang="zh">#######</value>
     </option>
     <option key="2">
         <value xml:lang="en">Voting rights on expense (e.g. insurance)</
value>
  <value xml:lang="zh">##############</value>
     </option>
  </mapping>
</origin>
</attribute-type>
```

This simplified `attribute-type` code snippet allows the value for Voting Right to be fetched from `holding-contract`, which is a smart contract defined somewhere else in the Smart Token, and present it in one of a few languages.

As another functionality, Smart Tokens can extend the product description and integrate variable parts, which can be upgraded without messing with upgradability of smart contracts.

Attested Information

Attestation is simply a signed message stating a fact. Attestations are often used to satisfy the conditions of the transactions — more on that in chapter Attestations. In the 1% property token example, the involved attestations are:

- The Identity authority and title deeds office attest the issuer's ownership of the property.
- A collateralization authority prevents double collateralization
- The buyers provide their identity or capacity to invest in this type of asset

The first two attestations are not stored in a smart contract for privacy and cost (size and number of transactions) reasons. It's possible to utilise zero-knowledge proof to provide anonymous proof that the attestation is for the said property and said owner, and it has not expired. What proofs are expected and can be validated is also described in Smart Tokens.

Furthermore, the transaction requires an identity attestation or investment capacity attestation from the buyers. These are described in Smart Tokens as well so the context (e.g. user-agent) can prevent the user from submitting a transaction without qualifying proof or help the user to select suitable attestations for a purchase transaction.

Smart Tokens do not just help relay the attestation data, but it also enables wallets to conveniently allow buyers and sellers to submit and read the attestation data without the need for a hosted DApp service.

Reference Information

Reference information is relevant to the token, but not part of the smart contract. In our example reference information includes data like previous property sales price or regional property performance. This is data which is useful for the owner or buyer of a token. It is provided by web services, typically through a RESTful API call.

It could be possible that eventually such information will end up being onchained. But even than they will be created and uploaded by an entity which has to be trusted. Reference data always leads to the problem of trust and web integration. With Smart Tokens this data is signed by the token issuer (not by the token owner - the token issuer is often an entity that deployed the smart contract). In the case of the property this could be a real estate specialist. The

set-operation Eventually, this could be a cryptographic set operation, but even if that happens, the metadata directing the context (user-agent) to perform the computation still needs to be described in Smart Tokens.

trusted-information Originally we call it "Trusted information", meaning the data is just "provided", without blockchain proofs or attestations, hence, it has to be explicitly trusted by the user. As it turned out, this term misfired as some developers think it means "proven information" and provided as trusted already. So we used a less precise term "Reference information", which, unfortunately, feels like a catch-all phrase.

reference information sourced from web APIs specified in Smart Tokens is assumed to be trusted, but can be changed later. The security chapter will detail different levels of trust

.

Action Information

Action information dictates the correct method to construct a blockchain transaction, like:

- What attestations are needed to prove the buyer's capacity to purchase?
- What parameters are needed for a purchase (e.g. number of 1% shares)?
- How to render the purchase form and translate to the user's local language?
- Are the conditions all met (e.g. a purchase isn't possible after the underlying property is liquidated)?
- How to cast a vote if one is aligned with token ownership?

In Ethereum this information is a super-set of the smart contract programmable interface called ABI. It also contains business logic (e.g. property must be still valid and owned by the seller) and presentation logic (e.g. the message "The property is liquidated. Purchase no longer possible"). With more sophisticated token, this parts becomes more complex and will make it hard to load it in a smart contract *and* visualize it appropriately in the wallet.

Smart Tokens make both the content as the visualization of the action information more flexible and accessible.

In conclusion, Smart Tokens allow the context (user-agent or trading engine) to:

- Fetch token related information from its holding smart contract, attestations and references.
- Produce a visual or audio rendering of the token
- Produce a list of actions that can be performed and explain how to construct the transactions.

Any party is able to render and apply functions to the token using Smart Tokens, including entities like generic marketplaces, user-agents and 3rd party apps. We call these parties "context" in general. This approach allows for a more seamless and intuitive interaction with the next-generation web, enhancing the user experience and expanding the possibilities for token utilization.

Payment side example: Intelligent Currency

In a token economy, it's essential to incorporate advanced business logic not only on the delivery side but also on the payment side. Here, payment refers to a token symbolizing a monetary unit used to compensate for deliverables, such as programmable tokens like DAI or ETH.

Consider our property example. There's a need for payment functions that aren't readily available in current wallets. For instance, a buyer may need to provide identity proof when purchasing a property share. Conversely, the property token issuer might want to establish recurring payouts from the income of the 1% token. If a platform is developed to tokenize property shares, the owners will require a straightforward way to integrate these types of payments. Additionally, purchasing a property share might involve multisig-payments, introducing trusted third parties like notaries. A property trading platform might want to associate payments with a list of certified notaries.

When we consider advanced payment tokens like DAI Dollar or collateralized loans with Dharma, wallets need to understand the specific mechanisms of the underlying smart contracts. For example, if you spend the last remaining DAI in your CDP, you're at a high liquidation risk, and the wallet should alert the user of the potential consequences. Given the growing popularity of these lending and stablecoin contracts, it's likely we'll see more of them in the future. Wallets must be capable of understanding the mechanisms of these payment tokens and informing the user accordingly.

One of the key features of blockchains in payment is their ability to facilitate programmable, intelligent currency. With Ethereum smart contracts, you can construct a vast universe of intriguing payment schemes. This could involve multisig contracts with complex logic, such as having a floating amount threshold that requires more cosigners as the amount increases, or contracts that only allow single-signed payments to certain accounts. Payments can also automatically include cashback or affiliate schemes, which trigger when a payment is made to a given address.

In the future, the payment side of smart contract blockchains like Ethereum is expected to undergo significant advancements and changes. There are proposals to introduce new IBAN-like address schemes or to associate payments with domain names of Ethereum Name Service (ENS). We're already witnessing smart contracted payment request providers and offchain payment railways, like Raiden or Plasma.

This list is far from exhaustive. There are numerous payment side innovations in China, such as points earned for encouraged payment behaviors, advanced cashback logics, lotteries on being the 100th, 200th, or 600th payment, free shipping insurance under certain conditions, red-packets that can only be used in paying consumption, and so forth.

It's anticipated that we'll see a broad range of innovations on the payment side in the future, which will coincide with the tokenization of deliverables. To handle these, wallets must be prepared.

Challenges on the payment side

The innovations on the payment side will pose challenges for wallets, merchants, and payment providers, similar to those on the delivery side:

How will the wallet understand the payment logic and visualize it? For instance, if you buy a pizza and you're registered for several cashback services. Some pizzas might give you cashback points under certain conditions, or you have cashback points which only count for certain merchants. Your wallet needs to be aware of this and inform you. With some cashback schemes, you might be required to interact with a smart contract during payment through a specially crafted transaction. How does your wallet know this?

What if the payment logic is updated? Sometimes the payment logic might be updated. For example, a cashback provider offers special discounts, or there's a new law for property shares. In these cases, the wallets making the payouts need to update the

logic to craft transactions.

How to incorporate complex payment logic into a smart contract? With the current model of smart contracts and hosted DApps, all the payment logic needs to be encapsulated within a smart contract or a set of smart contracts. This increases the contracts' complexity while limiting what can be achieved. The smart contract might need to interact with other smart contracts or trusted third parties providing changing lists of discount offers.

When done with traditional models, more sophisticated payment logics introduce complexity, security, and privacy issues, while creating a significant burden on wallets to integrate the logic and establish a robust updating infrastructure. At the same time, it limits what is achievable and restricts interoperability.

With the knowledge of smart tokens, TokenScript serves as an elegant and simple solution to circumvent these problems. Similar to the delivery side, the TokenScript XML allows the creator of a smart contract - or the owner of a receiving address - to introduce and update a set of information to the wallet:

1. Product description: Information about products, merchants, cashbacks, discount offers, handling fees, CDP, and other collateral information.
2. Attested information: Identity information, tax information, and so on.
3. Reference information: Third-party databases aligned to the payment or the product, like a list of special offers or notaries to select.
4. Action information (how to perform an asset action): Multisig-schemes, interaction with other smart contracts, recurring payments, receiver handle or ENS name, translation into other address schemes, crafting of transactions triggering smart contracts, etc.

In the context of the next-generation web, the integration of smart tokens into the payment side of transactions will significantly enhance the capabilities of wallets, enabling them to handle a wide range of innovative payment schemes while maintaining security and privacy.

Benefits of Smart Tokens

The advantages of Smart Tokens become evident when we examine the challenges of tokenization and the solutions that Smart Tokens provide. This section will delve into the benefits of Smart Tokens, focusing on how they address the issues inherent in the traditional token model and enhance the token ecosystem.

Delivery Side

Interoperability:

Consider a scenario where a property expert, Peter, creates a website called "Peter's Property Picks". On this platform, he curates the best properties available on the market, each represented by a token. Peter can list these properties with detailed information about price, location, etc., and allow users to purchase them with a single click.

In the traditional model, Peter would need to understand how to render the token on his website and update his site whenever the underlying smart contract or transaction rules change. If he fails to do so promptly, his users may submit transactions that don't comply with the updated rules, leading to transaction rejections.

Smart Tokens simplify this process. They allow Peter to keep his platform updated and responsive to events, and they enable the property tokens to be easily operated across various platforms.

Scalability

Horizontally, the same type of asset might have its token instances across multiple networks like Plasma Chains. Without such architectures, a token economy will hardly be able to scale. But having an all-knowing node to provide rendered token information for all existing tokens will be hard - and detrimental to the goal of scaling the blockchain economy while keeping the burden on nodes small. Therefore, the knowledge about the token (Smart Token) must be detached from the access to the token.

Vertically, transactions can be structured to create tokens on top of a token. For example, you might have a token that is made up of 1% property tokens, but distributes the risk over a sample of 100 global cities. The transaction and token access the component tokens. It cannot depend on the availability, security, and openness of the original DApp tied to that asset. Smart Tokens work in the middle for the making of such tokens, allowing the flexibility needed to vertically scale token assets.

Security

How do you know that the transaction you sign is what you want to sign? With standard payment transactions, this is a minor problem, but when it comes to the complex logic of token transfers, the traditional model of blockchain tokens will face significant challenges.

Take the 1% property token as an example. A confirmation might look like this: You are going to purchase 1% of property #802820 with 45 Ethers, are you sure? The user will be unsure if the glass ceiling designer 2-bedroom house he is watching is #802820. A dictionary-based translation visualizer cannot go further because correctly rendering the property token requires more than word processing.

Smart Tokens are designed to separate token rendering code and transaction generating code, and package them into its container, signed by a party that the user is likely to trust (often, signed by the same key used for deploying a smart contract). There are a few trust levels left, which we will detail in later chapters.

A user who is purchasing a 1% property token from Peter's Property Picks can be supplied with a rendering and transaction package, signed by the same group of people who created the holding contract of such tokens. Therefore, the user can purchase assets from any website with a similar level of trust, or purchase it from a WeChat or Facebook private message and know it is the real token being rendered and transacted.

Privacy

Almost all business operations involve some kind of identity. When you purchase a 1% property token, in most jurisdictions, you will be required to provide some kind of identity proof. In the traditional model, when you use a third-party website like Peter's Property Picks, this

site will require the identity proof and forward it to the seller, the notary, or the authority. This approach has well-known problems. You simply don't want your identity documents being stored on many website databases, if you don't want to fall victim to identity theft. The website taking your credentials can misuse it - for example, sell or analyze it - or the website can be hacked.

With Smart Tokens, the issuer of the token can formulate a destination where to send the identity file, add a public key of the receiver, a reference to the cryptographic library used to encrypt the file, and let the buyer forward it automatically. For example, you pick your property on Peter's site, and while the amount to pay will go to the property seller - minus the provision for Peter - the encrypted identity proof could go directly to the land registry authority, with a note about the purchased property attached.

User-Interface

Imagine you have purchased a few 1% property shares on Peter's site. In a traditional wallet, you only see it as little symbols - at best - with no further information. This is not what property investors want to see. They want to have pictures of the estate, prices, charts about the regional estate properties, expected date of payout, and so on.

Smart Tokens allow wallets to easily display all the needed data, as shown above.

Availability

As an owner of a 1% property share, you have some interesting options: You can vote on decisions about renovation or restructuring of the property, you can request a payout of income, you can sell the property, or you can take a mortgage on it. If those options are part of the smart contract, you need to access the blockchain to process them.

When Peter's Property Picks serves as a DApp site, you will perform your actions using his site. When his site is unavailable or down - or maybe when he forgot to pay his SSL certificate - your token will be effectively unable to be used to pursue the rights you bought with the token. This could essentially be abused to manipulate property prices or ballots about renovation.

The problems become worse when the 1% property token is embedded in further contracts or tokens. For example, there could be a site called Brad's Properties, which bundles 1% property token from other sides to create a property index. In the classical DApp model, you would be dependent on Peter's site being available to use a property token you bought there to become part of an index token. Or imagine you take a mortgage on your token. Which would be effectively a transaction between you and a bank would become a transaction involving Peter's site.

With Smart Tokens, the wallet will be able to support any required visualization and action by the user. This will make it unnecessary to rely on trusted third parties like DApps to use the functions associated with a token.

Payment Side

In this section, we delve into the advantages of TokenScript from the payment perspective. We will primarily use DAI Dollar as a reference point for payment, but we will also incorporate other examples to enhance understanding.

Security

When a user makes a payment with a DAI Dollar, the transaction is typically generated through a combination of local JavaScript (for instance, the JavaScript from a pizza shop's website) and the secure JavaScript provided by DAI. The transaction payload that the user sees contains parameters to access both the DAI contract (such as the payment amount) and the pizza shop's contract (such as the number and type of pizzas to be purchased). This transaction is then sent to the DAI contract and channeled (or proxied) to the pizza shop's contract.

There are two immediate security concerns here. First, there's the risk that the website may not have correctly implemented the MakerDAO JavaScript library, which is responsible for building the final transaction. Second, there's the possibility that the JavaScript at work isn't actually MakerDAO's, but a version that has been tampered with by a malicious actor.

TokenScript addresses these issues through its encapsulation method. Firstly, the transaction forming code is signed by MakerDAO separately and updated independently from the user-agent side. The website's code doesn't need to be signed because it only provides the business logic, not the payment logic. If a bug is discovered, DAI can suspend the payment by updating these signed instructions, and the pizza shop's website would behave as if it has been updated to address the issue. If the bug is found to be in the DAI holding contract, and a replacement contract is deployed, MakerDAO would update TokenScript and sign it again, without the pizza shop having to do anything.

Secondly, the wallet can explicitly ask the user to trust the TokenScript signed by MakerDAO, so the user would not need to trust the pizza shop's rendering of the transaction content, since it would be rendered by the trusted MakerDAO TokenScript.

Thirdly, if secure protocols need to be added, such as an attestation from the website (which can re-use the SSL certificate) to certify the transaction receiving smart contract, or the smart contract returning explicit trust of the website by domain name, the additional logic can be a combined effort of upgrading the dapp browser's support of a new TokenScript feature and the token issuer's new TokenScript code, without touching the website.

These security considerations apply to many other methods of payment and deliveries.

Interoperability

Adding support for DAI itself is challenging enough, let alone adding other payment side tokens. During the 2017-2018 frenzy, a lot of payment side tokens were invented and heavily invested in. Almost anything advertised not as a security token outlines some way their token can be used to pay or co-pay some goods and services. For example, electricity tokens were proposed as the future currency of tokenized electricity. Even if only 10% of these tokens are created by sincere ICO teams, all of them would foresee similar challenges as the integration of DAI token into the market.

Each payment side currency brings its own payment side logic. For instance, DAI has the following payment side logic:

1. The creation of DAI tokens requires a set-up phase, called CDP.
2. The risk level of a CDP changes. Users should receive a notification if their CDP is at liquidation risk.
3. If the balance runs low, but the user has quite a bit of Ethers on his/her account, they may pause the checkout to top up before returning to the checkout.

At the same time, you can think of many payment side innovations like Point+Pay

, where the points are selected at the same screen as payment. We presented some examples of these innovations above. Integrating all of them in wallets seems impossible.

TokenScript aims to facilitate payment side innovation as well as the deliverable side. Traditionally, partner support used to curb payment side innovation. For instance, American Express implemented points to pay API, but after years only less than 5% of partner e-commerce websites provided this as a checkout option. With TokenScript, such innovations can be integrated seamlessly directly into the wallet.

Scalability

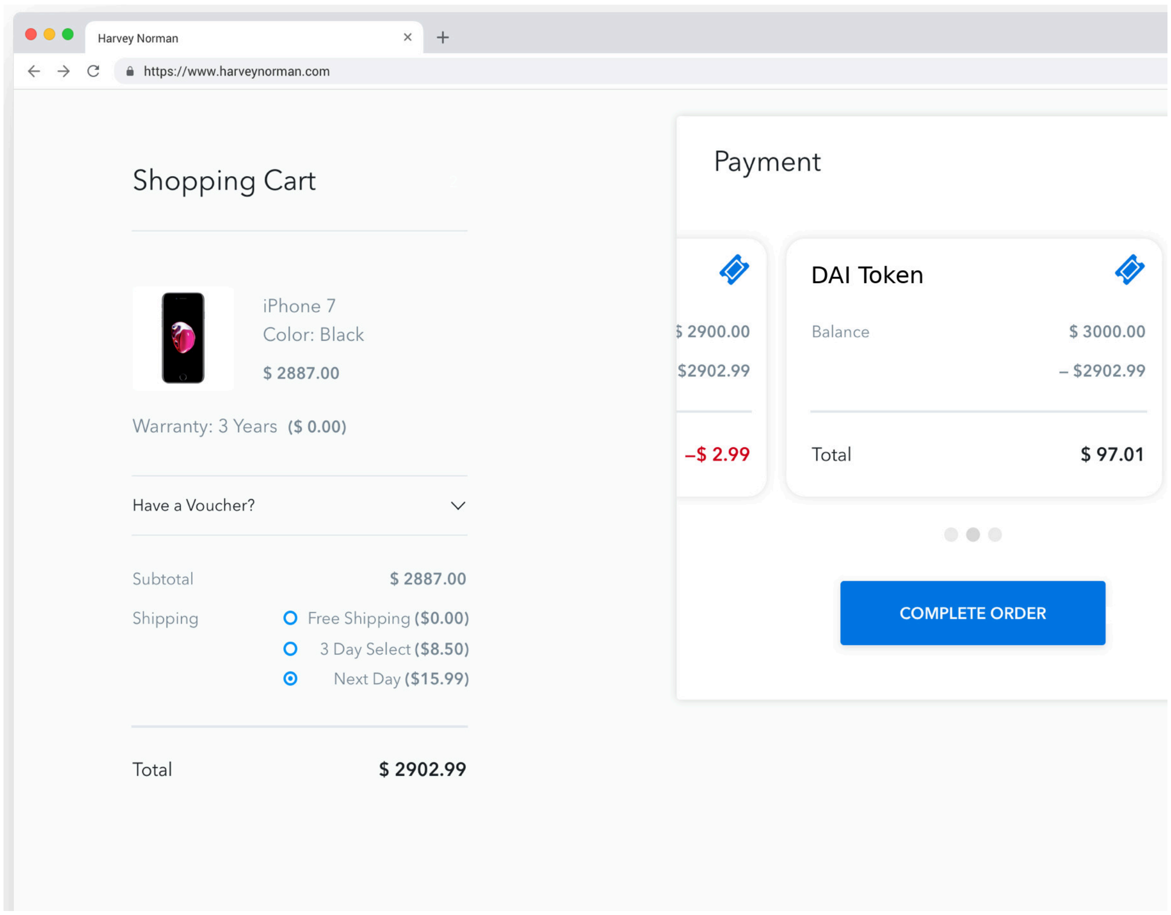
The payment and delivery may not be on the same blockchain. In fact, from a long-term perspective, they can't be, as blockchains are meant to scale on various layers.

Imagine the popular model of an ICO creating a token with the advent of a payment to a certain contract. What if the server does not have equal access to the client on some data like balance?

It's unlikely any scalability plan will not involve the participation of dapp browsers and wallets. They result in a situation that dapps could not take care of the payment side with whatever advanced JavaScript they can supply. TokenScript can help to create the needed communication channels to tell the wallet which chains are accepted for payments - and the server on which chain the payment happened.

User Interface

Any sophisticated payment logic, as presented above, needs to be represented in the wallet and allow the user to execute certain payment actions. TokenScript allows wallets to easily integrate a vast scope of payment options and innovations. Also, it allows the issuer of the payment TokenScript to update the payment logic, without needing the wallet to update it too.



Shopping Cart

iPhone 7
Color: Black
$ 2887.00

Warranty: 3 Years ($ 0.00)

Subtotal	$ 2887.00
Shipping	Free Shipping ($0.00)
	3 Day Select ($8.50)
	[✓ Next Day ($15.99)]
Total	$ 2902.99

Payment

$ 2900.00	DAI Token	
$2902.99	Balance	$ 3000.00
		– $2902.99
–$ 2.99	Total	$ 97.01

In general, the advantages of TokenScript for the payment side reflect the advantages on the delivery side. In both areas, TokenScript serves as a tool for innovators to create sophisticated logics, implement it into wallets, and update it.

Chapter 5

Address the "Integrate the web" need

Topics:

- Acquiring an iPhone
- A Future Iteration of AirBNB

We trace the reason that the web is poorly integrated to the only link between the units of the web, i.e. URL links. A link carries no business process, authentication or trust relationship. There are no anchoring points for integration on links.

We believe the token is the anchor points for integration. Again, this is best illustrated by examples.

Acquiring an iPhone

Let's consider a scenario where a user procures an iPhone from Harvey Norman, an online retailer, using blockchain technology. The transaction input would be a form of currency, and the output could potentially be five smart tokens:

- A shipping smart token, which can be redeemed at a local pick-up station to receive the product.
- A warranty smart token, issued by Apple, which enables the iPhone to be serviced at locations other than Harvey Norman (e.g., Apple Centre).
- A receipt smart token, issued by Harvey Norman, which permits the product to be returned within 90 days. It's also useful for obtaining a Tourism Tax Refund if the phone is taken out of Australia.
- A login smart token, which provides access to Apple's App Store, iTunes, and so on.
- An ownership smart token, representing the ownership of the iPhone.

Without smart tokens as the integration anchor, these different services might be carried out through various means, leading to a clutter of papers, order id numbers, links, and so on. With smart tokens, all the user needs is the seed of their wallet and the TokenScript file for the iPhone purchase. Even if they lose their wallet, they can easily restore all necessary information.

This model also simplifies interactions with tax authorities. For instance, if the buyer wants to claim an input tax deduction, they only need to provide their address to the tax authority. If the input tax deduction covers several purchases, they could provide their xpub key, and the tax authority can verify all claimed purchases in seconds without further proof.

The Shipping Smart Token

In the absence of a smart token, a user might receive a tracking number, which carries no authentication information and cannot be used to pick up the product unless a pickup code is provided, perhaps via SMS - a process that is poorly integrated.

With a shipping smart token, the token status can be remotely updated by the shipping company, even sending notifications to users about an upcoming delivery (if the token is held in a mobile wallet). With some cryptography, it's easy to authorize someone else to pick up a product.

Warranty Smart Token

Without this smart token, a user might need the serial number and an online registration process to activate the warranty. They might even need to create an account for that, whose password they will probably soon forget.

With a warranty smart token, the terms and expiration are easy to find, as they are token properties. The user can log in to the warranty service website with the token, bypassing the need for an account. The token can be programmed to receive messages like product callbacks or emergency security updates.

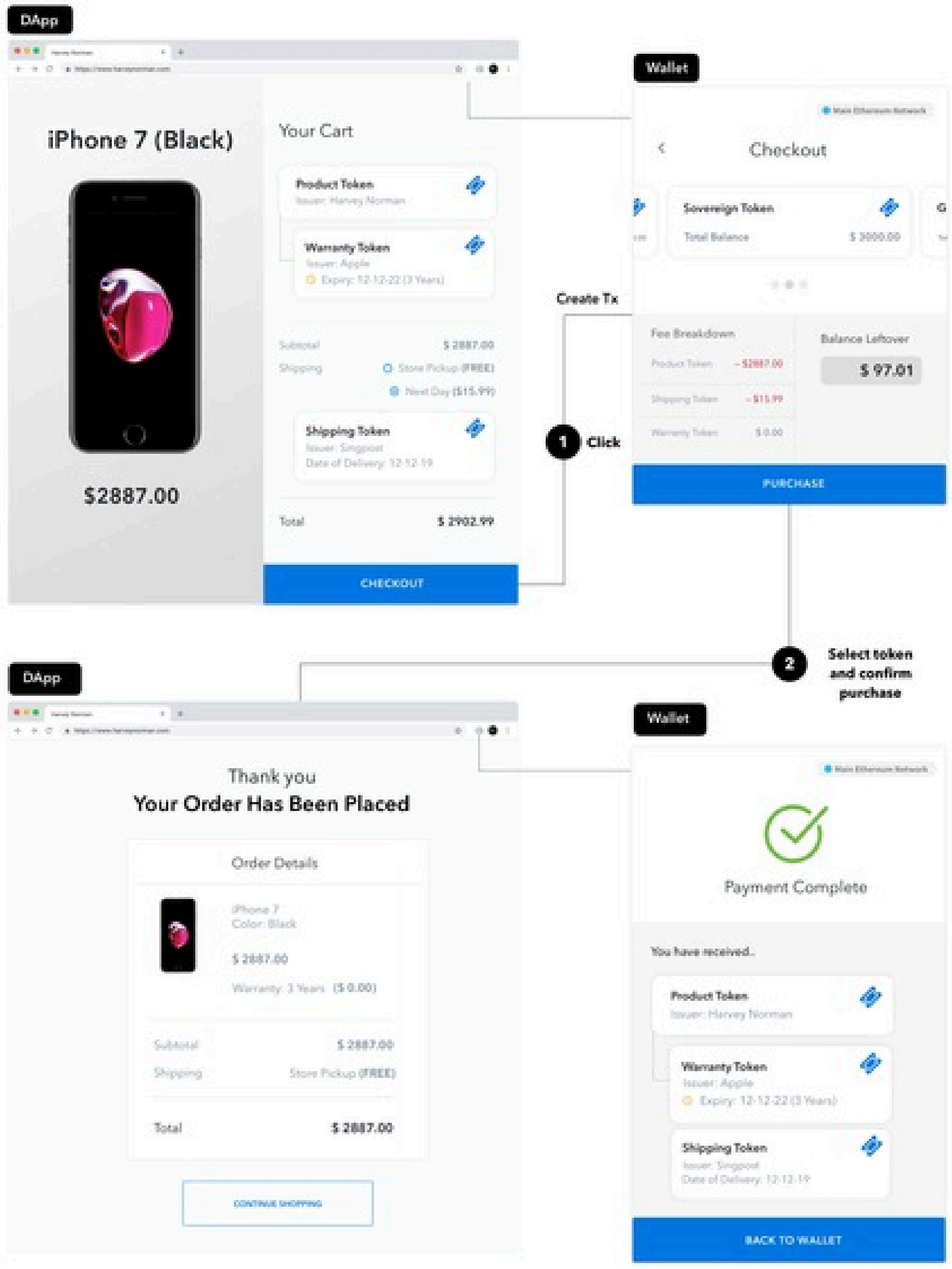
Receipt Smart Token

Without a reliable way to authenticate the purchase, an online purchased product usually cannot be returned to the store but might be returned via online means such as a postback. A smart token carries sufficient authentication methods for the process to be done in store.

Despite such a token not being transferable or authorized, it is still useful for 3rd party integrations. The Tax office will be satisfied that the receipt can't be faked without collaboration from the seller, and allows a swift and easy tax-refund process. If the phone is purchased for work, the employee can easily reclaim the expense from an employer with the trust implied.

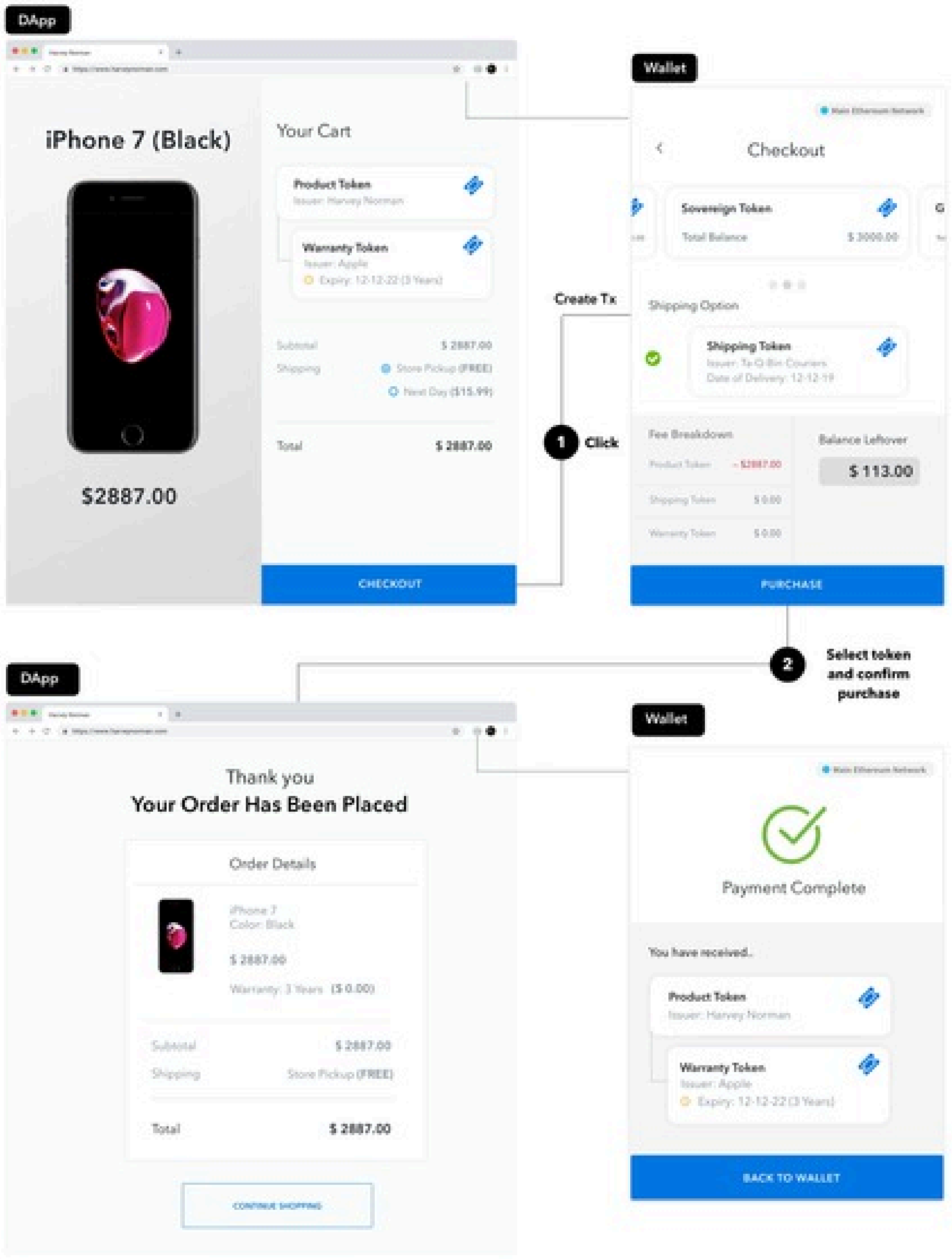
The Ownership Smart Token

This smart token could be needed to use any other token. If the iPhone is sold, the ownership smart token is transferred too, and the old owner will not be able to log in to the account of the new owner. There can also be a mechanism to delete the account history when the ownership smart token is transferred.



As we can see, the use of smart tokens allows for the integration of typically disparate business processes and web experiences. This aligns closely with another benefit of blockchain: a frictionless market. For instance:

- When the phone being traded is second-hand, it would be straightforward to pass the warranty to the next user through a smart token transfer, further opening the market.
- Since shipping can be tokenized, it would be easy for the buyer to choose their preferred shipping company without having to manually supply it with business context (address, product, weight, dates), further opening the market for competition. This is illustrated below.



This example can be further extended to solve complicated and innovative business cases. Suppose the purchase is not made with fiat currency but with American Express points. The iPhone purchase will be insured for screen damage, and as a result, the transaction will also output an insurance smart token.

When the mobile phone is repaired for screen damage, an invoice is issued concerning the purchase record of the phone to prove it is the same phone purchased with the points, enabling the insurance to be paid on the spot.

Without such smart tokens, the user will have to submit a billing statement, invoice, and evidence of repair to submit a claim. Many users will surely miss one of those documents; the claim may take a few days, and still be prone to fraud.

In this insurance case, the blockchain allows for business process innovation that otherwise would require the user to sacrifice convenience, simply because too many parties are involved and there lacks an integration anchor. This is the power of smart tokens in the next-generation web.

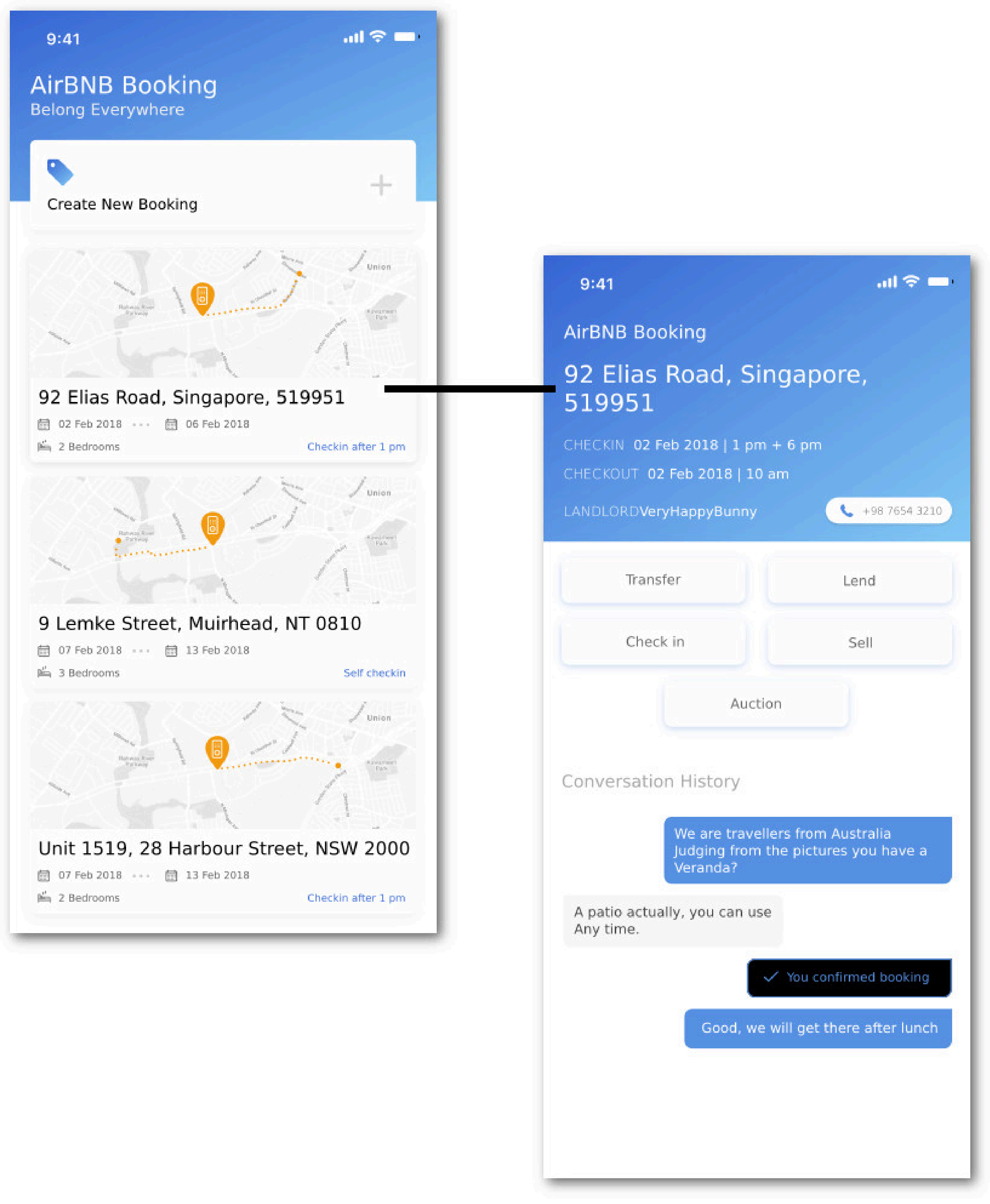
A Future Iteration of AirBNB

The potential of integration is significantly amplified by the next-generation web and the Internet of Things. Envision a future iteration of AirBNB, where bookings are represented by smart tokens. When a traveller rents an AirBNB property, they receive a smart token, which grants access to the property for a specific duration. Once the duration expires, the smart token essentially becomes void.

The property is secured by a smart-lock, which the traveller can unlock using their smart token. The smart-lock identifies the current holder of the booking smart token. If the property owner prefers additional security, they can request the traveller to provide an identity smart token, which replaces the need for an account on platforms like AirBNB.

If Alice holds a smart token that signifies her right to use a room during a specific timeframe, or "a booking" in layman's terms, she can check-in by either generating a QR code to validate the booking to the landlord or using an NFC-enabled phone to open the smart-lock.

This process doesn't introduce something fundamentally novel, but it streamlines an existing process. Presently, booking a property on AirBNB or similar platforms requires the traveller to create an account on the platform, provide credit card details (or some other form of identity verification), receive an order confirmation and an unlock key via email, among other steps. All these steps can be consolidated into a single smart token.



9:41

AirBNB Booking
Belong Everywhere

Create New Booking

92 Elias Road, Singapore, 519951
02 Feb 2018 ··· 06 Feb 2018
2 Bedrooms — Checkin after 1 pm

9 Lemke Street, Muirhead, NT 0810
07 Feb 2018 ··· 13 Feb 2018
3 Bedrooms — Self checkin

Unit 1519, 28 Harbour Street, NSW 2000
07 Feb 2018 ··· 13 Feb 2018
2 Bedrooms — Checkin after 1 pm

9:41

AirBNB Booking

92 Elias Road, Singapore, 519951

CHECKIN 02 Feb 2018 | 1 pm + 6 pm

CHECKOUT 02 Feb 2018 | 10 am

LANDLORD VeryHappyBunny — +98 7654 3210

Transfer | Lend | Check in | Sell | Auction

Conversation History

We are travellers from Australia Judging from the pictures you have a Veranda?

A patio actually, you can use Any time.

✓ You confirmed booking

Good, we will get there after lunch

Requirements

From the perspective of this desirable integration, we can deduce that TokenScript needs to fulfill the following requirements:

- Enable the definition of token actions. For instance, a shipping token would have a "redeem" action (via a QR code or NFC) and an "authorize" action which would permit someone else to pick up a delivery. In the case of AirBNB, there would be an unlock or verify action.
- Facilitate access to blockchain functions within an action.
- Facilitate access to web functions within an action. Essentially, this implies that both blockchain and web functions must be incorporated into TokenScript, enabling its integration. TokenScript should not only define how a token behaves but also how other environments respond to a token action.
- Enable the update of the token status, via a web API or signed message. The token status could be its validity during a specific timeframe, the extension of this timeframe, and much more (more on that later).

Chapter

6

The Architecture of TokenScript with Smart Tokens

Topics:

- Evolution in Business Processes and Market Dynamics
- Types of Tokens in the Context of Smart Tokens
- The Components of TokenScript

TokenScript, an XML markup language, serves as a pivotal link between smart tokens or smart contracts, and wallets, websites, and third-party platforms. Each TokenScript is authenticated by the token issuer and can be updated by the same entity. This ensures that TokenScript maintains the same security attributes as a smart contract, circumventing the drawbacks associated with storing all information in a smart contract.

Incorporating the concept of smart tokens, we have identified several key advantages of TokenScript, including Security, Interoperability, Privacy, Scalability, Availability, and User-Interface. The architecture of TokenScript should facilitate these benefits while supporting a broad range of business scenarios and diverse token and transaction compositions.

The architecture of TokenScript, particularly with the integration of smart tokens, is an ongoing development. While a comprehensive design guide is not yet available, we can outline some of the most significant design principles.

Evolution in Business Processes and Market Dynamics

In the early days of public blockchain projects, the prevalent approach was to embed both token logic and business processes into smart contracts. For instance, in an online retail project, a smart contract would not only process an order but also manage the inventory. The token transaction logic, such as the conditions under which the transaction is valid, was intertwined with business processes like inventory checks. This method was a natural extension of traditional website building practices.

However, this approach is akin to an IKEA manager deciding to include details like the aisle number for fetching the furniture package in the sales contract. Such a contract would require constant modifications to reflect changes in warehouse management, making it impractical. The aisle number has no bearing on the validity of the trade.

When these attempts proved unfeasible, developers started voicing concerns about the performance and privacy issues in the current generation of blockchains. While it's true that these blockchains have significant performance and privacy limitations, extending them is not the correct approach to address business process issues. The belief that new technology must be a faster and stronger version of the previous ones led to misconceptions about blockchain being a superset of AWS' functionalities. However, due to additional security assumptions, Byzantine Fault tolerance blockchains will never outperform AWS's business engines.

The authors of this paper propose a distinction between a smart contract and a business process:

1. A smart contract should govern the transaction rules of tokens, not their utility.
2. A smart contract and business process should be integrated through the tokens.

This distinction is fundamental for the design of TokenScript. Let's illustrate this with an example: The iPhone purchase.

At the time of the purchase, a shipment token is created, representing the user's right to receive the delivery. The blockchain doesn't act as a cloud platform for the inventory management system. Instead, the shipment token serves as a point of integration. It allows the warehouse to locate the product, label it, inform the user that the product is ready to be shipped, and send it on its way.

As the business matures and markets become less frictional, two changes might occur:

Change in the business process

The first change could be the online retailer finding a better shipment company. In this case, the new shipment company will integrate the same shipment token, sending shipping progress information in place of the old one. The customers can still prove ownership of the shipping with his token, e.g., by using an NFC mobile phone to touch the deliverer's hand-held device.

There is no need to change the smart contract transaction rules. The online retailer can even change the shipping company when the product is first under-delivered without the user changing his token.

This change illustrates that the business process should be decoupled from the token and integrated through it instead.

Change in the market

It might happen that some users bulk-purchase a year's shipping from a delivery company, to enjoy AmazonPrime-like free shipping privileges without using Amazon. A credit card company might even start to provide such a privilege to the subscribers of their card, which is also represented by a token.

Now the online retailer decides to join the game to stay competitive. This time, he would need to modify his smart contract, changing the transaction rule so that a shipping token can be accepted at the time of purchase. In such a case, the output of the transaction will not have a shipping token, since one is already provided.

The online retailer will necessarily modify his business process to expect pick-ups from any delivery company the user purchased shipping tokens from.

This change illustrates that a new transaction rule would result in a change of smart contract.

Implications for the Design of TokenScript: Messaging in the Context of Smart Tokens

In a dynamic business environment, changes in business processes should not necessitate alterations in the smart contract. Instead, modifications in transaction rules, a natural outcome of a free market's evolution, should instigate changes in the smart contract. The primary function of the blockchain is to facilitate a seamless market, not to streamline business processes.

This perspective is actualized through the application of TokenScript in the context of smart tokens. In the first scenario, TokenScript defines a shipping token capable of receiving messages. At its most basic, the message is entrusted and presented to the user interface.

Here is an illustration of the TokenScript segment pertinent to messaging:

```
<token>
  <name xml:lang="en">Shipment</name>
  <name xml:lang="zh">##</name>
  <name xml:lang="es">Despacho</name>
  [...]
  <states>
     <state name="initialised"/>
     <state name="dispathced"/>
 <state name="collectable"/>
 <state name="used"/>
 <state name="expired"/>
 <state name="returned"/>
  </states>
  <messages-acl>
     <trust signed="issuer">
     <permission>
         <display type="history"/>
  <display type="notification"/>
         </permission>
         <condition state="initialised"/>
 </trust>
 <trust certified="issuer">
     <permission>
         <display type="history"/>
  <display type="notification"/>
         </permission>
         <condition state="dispatched"/>
 </trust>
 [...]
```

The `<states>...</states>` segment enumerates the states, forming the foundation for defining the messages that the token holder is permitted to receive.

The first `<trust>...</trust>` structure enables the user agent to accept and display any signed messages from the token issuer, in this case, the online retailer, as a notification and an entry in the message history when the token's state is initialised.

The second `<trust>...</trust>` structure allows the user agent to accept and display any signed messages, whose signing verification key is certified by the issuer of the token, as a notification and an entry in the message history when the token's state is "dispatched". This effectively permits any entity explicitly trusted by the token issuer to issue a message at the "dispatched" state.

When the online retailer changes their delivery company, they could issue a certificate on the public key of the new delivery company, thereby authorizing them to send messages to the token holders (buyers) to update the delivery status, yet restricting the messages to only certain stages of the business process.

This code snippet illustrates that TokenScript, by providing such flexibility, connects to a new business process without necessitating a change in the smart contract or impacting the user experience. It also facilitates communication to the token holder without sending messages through smart contracts.

The actual method of communication is left to be implemented by other layers of blockchain technology like a message queue or even a distributed message queue.

It's also feasible to write TokenScript in such a way that only messages from the online retailer are trusted and displayed. Therefore, any new delivery company must send their delivery status message to the online retailer's systems to be forwarded to the buyer. However, there are availability and privacy reasons why this may not be ideal. For instance, a delivery company should be able to operate when the online retailer is offline; the user might send the door entrance passcode to the delivery company which the online retailer should not learn.

Types of Tokens in the Context of Smart Tokens

TokenScript, in its quest to define types of tokens, has to consider the evolution of token categorization within the Ethereum community. Since 2018, tokens have been broadly classified as fungible and non-fungible tokens.

Fungible tokens are akin to currency, with a balance, often implemented via ERC20. However, practical applications such as pre-authorisation and state channel setup necessitate functionalities beyond the typical ERC20. Non-fungible tokens, on the other hand, are unique entities like crypto-kittens, typically having one unit per token.

However, this categorization does not fully encapsulate the potential spectrum of tokens and may overlap in certain instances. For instance, consider a property token representing 1% ownership. Each such token is fungible with another issued by the same issuer for the same property. However, a token representing 0.88% ownership, created to appeal to Chinese investors due to the cultural significance of the number 8, would not be fungible with the 1% token. Clearly, a percentage of ownership of different properties are not fungible with each other. A mixed token with 1% of multiple properties would be fungible, but simultaneously, individualized, non-fungible tokens could represent an investor's unique selection of properties. TokenScript's design must accommodate the fact that the distinction between fungible and non-fungible tokens is not absolute and that hybrid forms may exist.

This paper introduces the concept of attestations, a long-standing but underutilized concept, and categorizes tokens as "blockchain tokens" and "attestations". The former encompasses both fungible and non-fungible tokens, while the latter, "attestation", will be elaborated upon here.

Attestations

An attestation is a cryptographically signed message affirming something about an entity - a person, a token, or another attestation. If the attestation is a defining attribute of the entity, neither the entity nor the attestation can be transferred independently on the blockchain.

Consider a car ownership token: it would be a blockchain token, subject to typical buy, sell, and transfer rules. However, an insurance token associated with the car should not behave like a blockchain token. If insurance is compulsory, it is an attestation on the car and therefore cannot be transferred independently. If the insurance is comprehensive, it is an attestation on the car and the driver, and cannot be seamlessly transferred even if the car is transferred. In both cases, the insurance token should not be transferable.

This principle applies to all attestations: They are non-transferable as they are tied to a specific entity. An attestation does not necessarily need to be on the blockchain.

For instance, a personal identity attestation doesn't need to leave any trace on public blockchains like Ethereum unless it is used for a blockchain transaction or revoked. However, it remains an item in the user's wallet, as it might need to be extended, re-attested due to changes in a person's identity, or used to log in to services.

Moreover, an attestation can influence transactions. For example, a VIP member can enjoy a 10% discount on services by providing a VIP member attestation with a cryptocurrency transaction. An attestation of capped car services, valid for 5 years, allows the car to be serviced with the bill capped to a certain amount before its expiry. Adult services could demand an "over 18" attestation for their transactions. Linking attestations to transactions opens up a plethora of intriguing business cases.

As attestations involve identity, privacy is a primary concern. To combat linkability (the subject of an attestation being identified by the public use of such an attestation), the attestation used in transactions must differ from the one in a user's wallet. This issue has been addressed in another publication.

In most cases, attestations only leave traces when a transaction requires it. However, there are instances when attestations leave traces on the blockchain when they are created or revoked.

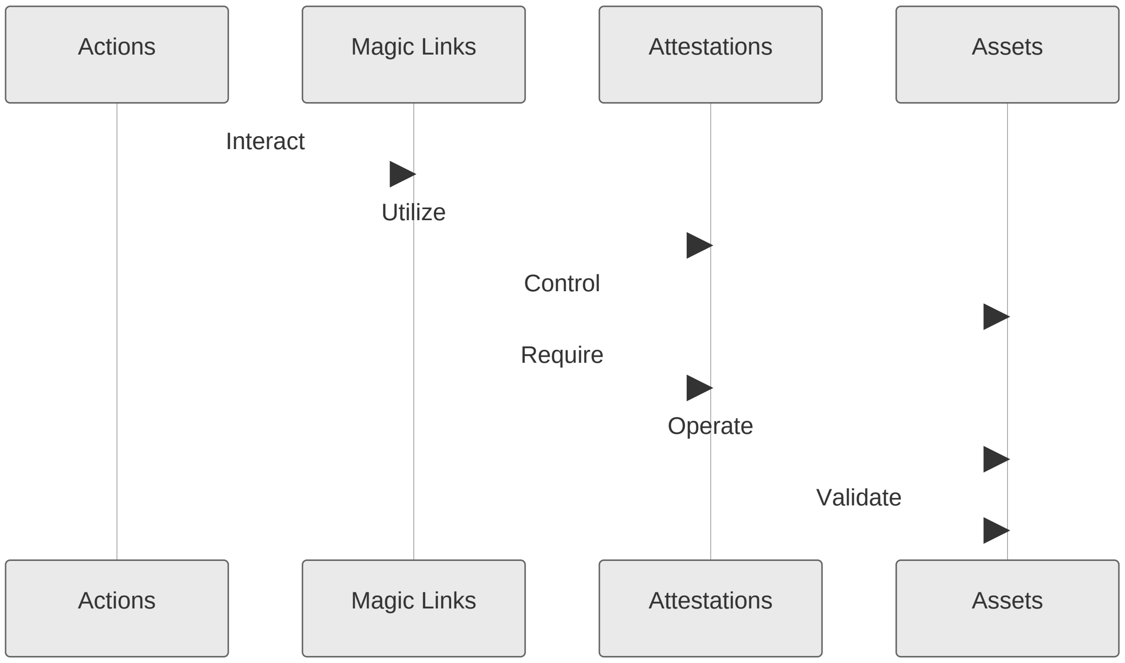
To illustrate a case where the *issuing* of an attestation must occur on the blockchain or leave a trace on the blockchain, consider an aircraft engine. Represented by a token, it carries numerous attestations, like repair records, which significantly affect its valuation, safety, and insurance properties. These attestations are in the seller's wallet, but an Aircraft maintenance provider must add a hash of such an attestation each time the engine undergoes maintenance. Buyers would not purchase it if they are not presented with these attestations that match the blockchain records.

To illustrate a case when the *revocation* of an attestation must occur on the blockchain, consider a FIFA ticket. Issued by the event's organizer, it attests the owner's right to enter the venue. If a ticket's owner decides to sell his tickets on the blockchain following the corresponding smart contract rules, the ticket has to be used as the input of such a transaction and considered consumed, while a blockchain token representing the same entitlement would be created and traded. This concept was tested in a FIFA ticket experiment in mid-2018, and internally we refer to such an attestation as "a spawnable" as its use spawns a blockchain token. The details of that experiment can be found in our other publications.

In the context of Smart Tokens, these concepts of fungible, non-fungible, and attestations take on new dimensions, enabling a more nuanced and flexible approach to token categorization and usage in the next-generation web.

The Components of TokenScript

TokenScript, when viewed through the lens of smart tokens, presents a more nuanced and comprehensive framework for token interaction and management. This framework is composed of several key components, each playing a distinct role in the overall functionality of the token ecosystem.



Actions

Actions in TokenScript are the operations that can be performed with a token. These actions can be broadly classified into three categories:

- Utilizing the token to access a web service
- Employing the token to control IoT devices

- Conducting transactions with the token

For instance, a car token might have actions such as 'Unlock', 'Authorize', and 'Lend', each with its own specific functionality and requirements. Not all actions are inherently provided by the token. Some actions, like 'Transfer', might be provided by a generic token's TokenScript, while others, like 'Auction' or 'List for sharing', might be provided by external entities.

Magic Links

Magic links serve as shortcuts to actions on specific assets. They are typically sent to the asset owner and come with the necessary attestations for a transaction, such as an atomic swap. These links streamline the process of performing complex actions, making the user experience more seamless and efficient.

Attestations

Attestations are akin to tokens, but they are non-transferable. They are essentially cryptographically signed messages that testify something about an object. These attestations can have significant implications for the object they are associated with, affecting its transferability and functionality on the blockchain. Attestations can be seen as a form of asset that validates or verifies something about another asset or entity, adding an extra layer of security and trust to the token ecosystem.

Assets

In the context of TokenScript, an asset is anything of value that can be owned. This definition is broad and does not necessarily require the asset to produce a return. Assets can be tangible, like a piece of armor in a video game, or intangible, like the right to a bottle of wine. The concept of assets in TokenScript extends beyond the traditional understanding of assets in the financial world, encompassing a wide range of items that hold value in the context of the next-generation web.

These components together form the backbone of TokenScript, providing a robust and flexible framework for token interaction and management in the era of smart tokens.

Conclusion

As we reach the end of this document, it's clear that the landscape of digital tokens has evolved significantly over the years. The introduction of smart tokens has not only expanded the possibilities of what can be achieved with tokens but also redefined our understanding of their potential applications.

The concepts and ideas presented in this document, from the nuanced categorization of tokens to the intricate workings of TokenScript, attest to the transformative power of smart tokens. They have the potential to reshape the next-generation web, making it more secure, efficient, and user-friendly.

However, the journey doesn't end here. The world of smart tokens is still evolving, and there's much more to explore and understand. As we continue to delve deeper into this fascinating realm, we look forward to discovering new possibilities and pushing the boundaries of what can be achieved.

Thank you for joining us on this journey. We hope that this document has provided you with valuable insights into the world of smart tokens and sparked your curiosity to learn more. As we continue to explore and innovate, we invite you to join us in shaping the future of the next-generation web.