STAS

White paper

Scope of Document

The white paper provides an overview and introduction to Bitcoin, the protocol of Bitcoin SV, and knowledge required to explore the STAS tokenization solution. The actual detailed explanation over the STAS tokenization life cycle and the functionalities can be found in the technical documentation that can be obtained once registered on the TAAL Console or by contacting TAAL through the commercial team.

Further this is a living document and TAAL reserves the right to publish a new version as deemed necessary as we further develop the STAS as a token solution.

The white paper describes STAS and not the licensing model behind the token solution as such.

Audience

The audience of the white paper is intended for developers and end users, who are evaluating and interested in implementing a tokenization solution build on the Bitcoin native script.

Document revisions

Version	**Date**	**Author**
Initial draft	10. October 2020	Jerry Chan
Updating and final version 1.0	14. January 2022	Simon Giselbrecht

Acronyms, Abbreviations, and Definition

Acrondyms/Terms	**Abbreviations/Definition**
API	Application Programming Interface
Blockchain	It is a form of distributed database (or ledger) that acts as a record of all the valid transactions, which are transmitted to the blockchain network.
BSV	Bitcoin Satoshi Vision
BSV	Bitcoin Core
CPU	Central Processing Unit
ECDSA	Elliptic Curve Digital Signature Algorithm
F-NFT	Fractionalized Non-fungible Token
NFT	Non-Fungible Token
PKH	Public Key Hash
P2PKH	Pay-to-Public Key-Hash
SDK	Software Development Kit
SIT	Safe Instant Transactions
Sats	Satoshis It is the smallest division of a Bitcoin and the base unit of exchange in the BSV network. There are 100,000,000 Satoshis in 1 Bitcoin.
SSD	Solid-State Drive
STAS	Substantiated Tokens from Actualized Satoshis

	It is the tokenization standard described in this paper and allows to assign properties to Bitcoins or Satoshis.
Token	It is a type of digital property or control element that can be transferred using a blockchain.
TXID	Transaction Identifier
USD	United States Dollar
UTXO	Unspent Transaction Output It is an accounting method used in Bitcoin, and it refers to a balance of a BSV coin address that is controlled by a specific token.
VOUT	Selects an output of the previous transaction

Contents

Table of figures

1 Introduction

TAAL aims to offer a token solution that enables companies, individuals, and organizations to mint, transfer, and redeem tokens using our native Bitcoin-based STAS scripts.

The STAS script uses the Bitcoin (BSV) protocol, a permissionless public blockchain protocol that makes any asset decentralized once processed and removes the need for any third parties to be involved in moving said asset back and forth. A token can represent a contractual claim over an asset and allow any type of information and / or data to be tokenized.

The STAS digital asset tokenization is an on-chain native Bitcoin script-based solution. It enables you to build token solutions for any number of use cases. TAAL provides a Software Development Kit (SDK) that wraps the STAS script, which locks or unlocks Satoshis enabling you to create and send tokens in the form of raw transactions to the BSV blockchain, and to read the tokens using the Application Programing Interface (API) by using whatsonchain.com.

1.1 Bitcoin SV

The vision of Bitcoin Satoshi Vision (short BSV) is to facilitate a world money on the model outlined in the original Bitcoin white paper. To achieve this goal the BSV ecosystem forked BCH, increased the maximum block size, and restored certain functionalities (including the ability to deploy smart contracts and thus allow the creation of tokens) of the Bitcoin scripting language. BSV is a peer-to-peer electronic cash blockchain that allows various use cases to be built on top of this programmable money.

One of the main advantages of using BSV is the ability to create a large number and variety of transactions at low fees. As of January 17, 2022, the fees per transaction are $0.00065 USD.

Another key advantage of the BSV blockchain technology over others is the underlying Bitcoin protocol that removes trusted intermediaries from the payment process and allows competition between the miners to eternalize the transaction. That feature allows for fair pricing of the transaction fees. Blockchain's ability to process a large volume of transactions for a fraction of a penny in near real-time is one of the technology's primary promises that BSV fulfills. BSV will roll-out Teranode, which will lower the transaction cost even further and allow for the processing of millions of transactions per second.

The main benefits for developers and businesses to adopt BSV are best summarized as follows:

- Professional Standardization: Provided by the BSV Technical Standard Committee
- Stability: Enables businesses to plan years in advance and commit significant resources to build on a stable protocol
- Scalability: Delivers capacity increases through scalable architecture and the miner-configurable block sizes
- Security: Provides the best practice change management processes, external security audits, and lucrative bug-bounty program
- Safe Instant Transactions (SIT): Unlocks the bricks-and-mortar merchant market and enables new business models with micropayments and nano services.

1.2 UTXO accounting method

In blockchain there are two main types of record-keeping models in use. The first method is called the UTXO (**Unspent Transaction Output**) model and the second one is the Account/Balance Model. The UTXO model is employed by Bitcoin while Ethereum uses the Account/Balance Model.

- In **UTXO**, a user's wallet keeps track of a list of unspent transactions associated with all addresses owned by the user, and the balance of the wallet is calculated as the sum of those unspent transactions.
- The **Account/Balance Model**, on the other hand, keeps track of the balance of each account as a global state. The balance of an account is checked to make sure it is larger than or equal to the spending transaction amount.

To understand the mechanics of the STAS script logic and its ability to create "smart contracts" from UTXOs, it is important to delve into the preferred method.

A simple way of understanding the UTXO model is to compare it to a triple-entry-bookkeeping ledger. For simple illustration purposes let's use the following logic:

Bob wants to go shopping and wants to visit five shops, but he only has a single $50 bank note. He can go to the bank and exchange the $50 note for five $10 notes and pay a fee for the transaction, or he can go to the first store and pay for his purchase with his $50 note, get the change back, and continue shopping in the other four stores on his list.

At the start of his shopping spree, the entire balance of $50 can be viewed in his wallet and is categorized as unspent cash. After he visits the first store, his purchase is recorded and the unspent amount he holds becomes smaller. It continues to dwindle as he visits each subsequent store until he has spent the entire amount in his wallet. On the other side of the transactions, the shops all have added to the total amount in their cash register.

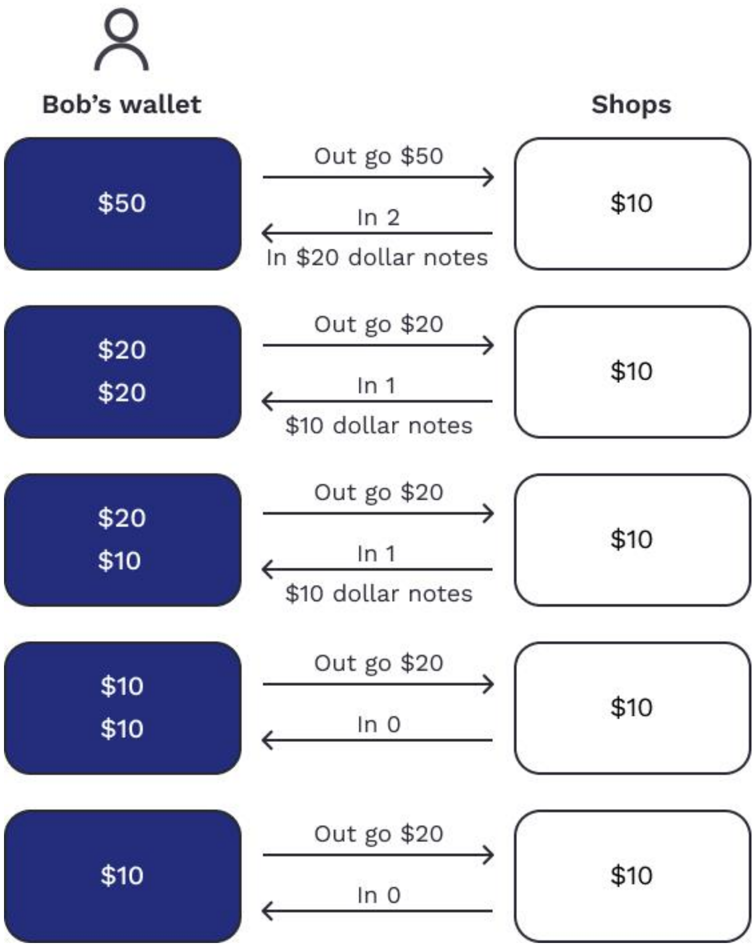


Figure 1: Showing an example of spending of tokens

With the more sophisticated logic of the UTXO model, a transaction becomes an output that has locked an amount of the digital currency in a way that can only be solved or unlocked by the possessor of the correct digital key — most likely, Bob the shopper. If Bob spends a certain amount, then that amount is removed from the UTXO set. In this case, UTXOs are processed continuously and are responsible for starting and ending each transaction. The confirmation of transaction results in the removal of coins spent from the UTXO set, but a record of the coins spent still exists on the ledger. Such as shown in Figure 1 above, the overall balance of cash out of Bob’s wallet lowers his balance.

A UTXO database is used to store output from the cryptocurrency transactions. This database is initially set as zero. As transactions take place, the database is populated with the various transactions outputs. Such as showing in Figure 1 above, Bob has spent his cash as outputs from his wallet, whilst the shops have received a payment as an input that can be spent for buying new things. The ledger in this case is not a publicly visible one. However, in the concept of BSV and our STAS token solution, the balances can all be viewed on the public ledger over a blockchain explorer. A blockchain explorer allows anybody to view the content to prove the accuracy of transactions that have been processed by a miner who validated and verified the transaction before it was mined and stored immutably on the public ledger (which can also be referred to as a public cloud consisting of information and data).

Bitcoins can be transacted in fractions of up to 10^-8 Bitcoin. The spending does not take place using single bytes, rather the multiple chunks (UTXOs) are retrieved by the wallet logic to fulfill a spending request. For example, a purchase worth 1 Bitcoin can use one UTXO with a value of 0.7 BT and another with a value of 0.5 BSV. The change from each of these fractions is a new output, which is added to the UTXO set and can be spent later.

1.3 Token

BSV reinstated the original Bitcoin protocol and has now made it possible to build tokens and smart contracts directly on Bitcoin.

Bitcoin allows tokens or smart-contract solutions to be built using on-chain or off-chain applications (e.g. run.network or tokenized.io).

In terms of this white paper, we define a token as a type of digital property or control element that can be transferred using blockchain. In blockchain technology, you can create and define a token using specific data fields within a transaction. The users of the blockchain or a particular token protocol can interpret the additional data (token data). The embedded data usually contains contractual information, which is legally binding, for the person minting (or issuing) the token but also essential information for others to use and redeem back for the underlying value the token may have. Most tokens represent a form of unilateral contract.

The validity of a token transaction in STAS is defined by a valid Bitcoin script and no other outside oracles. The user to whose address the output is locked, owns the token. If you are the user, you can use, sell, or trade the token for a particular purpose according to the token protocol.

As example:

- A cinema company can issue cinema tokens that represent tickets o users, who can pay for each token in return for watching a film at the cinema. The token in this case is essentially a ticket that can be collected and redeemed again by the issuer.
- A bank can issue tokens in exchange for its clients' USD deposits, which can be used by the clients to make regular payments and later exchanged back at the bank for USD by third parties.

1.4 STAS (STAS tokens)

Tokenization is the concept of issuing a digital token that represents a real-world asset such as stocks, bonds, real-estate holdings, as well as event tickets and travel cards, loyalty points, casino chips, and much more that can be issued and redeemed in a fungible (e.g. be traded on exchanges as a standardized good) or non-fungible token (representing a unique digital property). STAS is an ideal token solution for tokens for two reasons: BSV allows scaling based on the throughput capacity needed to support token requirements at the enterprise level while keeping transaction fees extremely low. Both the STAS token solution and the BSV ecosystem are compliant-compatible, making it easier for token issuers and professional users to comply with relevant laws and regulations. A security token or asset token can be issued but at the same time the solution can be used for countless other purposes, thus requiring less regulatory scrutiny.

STAS enables an issuer to convert a generic amount of unsubstantiated Satoshis (Bitcoin) into a minted token in a transparent and open fashion. It is a contract stored in the public BSV blockchain that can be transferred through the BSV network peer-to-peer. If the contract represents a redeemable claim, it can be used to collect this claim from the issuer.

2 Overview of tokenization

The popularity of Bitcoin has revolutionized how investments and assets are issued, managed, and transacted.

Blockchain enables an asset to break down into smaller units, representing ownership, encouraging the accessibility of investment and ease of use in handling them. This attribute leads to a transparent and fair market. It also allows assets that previously had no global market to become liquid. Everything can be tokenized and issued on a distributed ledger, such as fractions of paintings, digital media platforms, real-estate property, company shares, loans, collectibles, and many more.

Tokenization is a process by which an issuer creates digital tokens on a distributed ledger or blockchain. The tokens represent digital or physical property, or a control element. Blockchain confirms that once you buy tokens representing an asset, any single authority cannot erase or change your ownership, and it remains immutable.

For example, consider you have a property worth $500,000 in Vienna. Tokenization can convert ownership of the property into 500,000 tokens, and each one represents a tiny percentage (0.0002%) of the property. Consider you need to borrow $50,000, and you do not want to sell your property as you need a place to live. Instead of selling the house, you issue tokens on a public distributed ledger, like BSV, using the STAS tokenization scripts, which enables people to freely buy and sell on different exchanges. When someone buys a token, they buy 0.0002% of the ownership in the asset. It is required to buy 500,000 tokens to become 100% owner of the property. In the case of BSV the distributed ledger is immutable, and hence no one can erase the ownership of the investor, who has bought the tokens.

Tokenization offers for both fractional ownership and proof-of-ownership. You can tokenize assets, like venture capital funds, bonds, commodities, sports teams, racehorses, artwork, real-estate properties, celebrities, and anything worth tokenizing. In these cases, Bitcoin acts like “smart paper”. The Bitcoins are a commodity that can be used to change the Bitcoin ledger. STAS scripts are like taking that commodity and printing something onto it with smart ink. When the “paper” is returned it can be fully recycled by the issuer.

2.1 Token types

There are various types of definitions of token classifications. For simplicity, we will categorize the tokens into the following main types: fungible and non-fungible tokens.

2.1.1 Fungible tokens

In a Fungible token, each unit of the tokenized asset has the same market value and validity. For example, in a Bitcoin, all units of 1 BSV are the same, and have the same market value and are interchangeable. It does not matter from whom a BSV was purchased, as all BSV units have the same functionality and are part of the same network. You can swap one-fourth of a BSV with someone else’s one-fourth of a BSV, with confidence that your one-fourth of a BSV holds the same value, even if it is one-fourth of a different coin. A fungible cryptocurrency can be divided into as many decimal places, which are configured during its issuance. Each of the smallest units will have the same value and validity.

2.1.2 Non-fungible token (NFT)

Non-fungible tokens cannot be replaced with tokens of the same type because each token represents a unique instance. Each token differs from another token of the same type and has unique information and attributes. The tokens can have a high degree of ingenuity within each one. They can also be a digital ticket that is valid for a specific day and seat and will carry the properties required to create this distinctiveness. NFTs are not divisible, although the Fractionalized NFTs (F-NFTs) offer fractional ownership of NFTs, a feature that can be attractive for buyers of expensive fine art or commercial real estate.

3 Creating and submitting transactions

To fully understand how STAS works it is important to gain some level of understanding over how a Bitcoin transaction is created, signed, validated, and submitted to the blockchain.

3.1 Components of a transaction

As previously stated, a blockchain is a form of distributed database (or ledger) that acts as a record of all the valid transactions, which are transmitted to the blockchain network once validated by node.

The valid transactions broadcasted on the network are recorded on the blockchain by miners (or mining nodes) in blocks. A blockchain transaction is used to transfer control or custody of an amount of a digital asset.

Each transaction includes:

- **Input:** An input includes a reference to an UTXO from a previous transaction. A transaction uses UTXOs as inputs and distributes their value to the new outputs.
- **Output:** An output includes a locking condition that locks the value of the output, and certain data (e.g.: a set of signatures), to be provided in an input of a new next transaction to be unlocked. Outputs can also be used to inscribe data, such as text, images, and so on, onto the ledger.

An input of a transaction includes a digital signature that signs over at least part of the transaction. Therefore, a chain of transactions includes a chain of digital signatures that map the entire history of valid exchanges of the digital asset all the way back to its origin. The blockchain begins with a genesis block, which is the first block ever created. The n^{th} block references the $n - 1^{th}$ block, the $n - 1^{th}$ block references the $n - 2^{th}$ block, and so on, back to the genesis block.

A block contains an ordered list of blockchain transactions and a block header. The block header includes:

- a Merkle root, which is generated by hashing the ordered list of blockchain transactions into a Merkle tree
- a timestamp
- a reference to the previous block, which the present block builds upon and the means to validate the proof-of-work required for other miners to accept the block as valid. The validation is a hash puzzle, which is unique to each block.

The blockchain protocol run by mining nodes of the blockchain network uses a hashing algorithm that requires the miners to pre-build their candidate block before trying to solve the hash-puzzle. The new blocks cannot be submitted to the network without the correct answer. The process of mining is the process of competing to be the next to find the answer that solves the current block. The hash puzzle in each block is difficult to solve, but once a valid solution is found, it is very easy for the rest of the network to confirm that the solution is correct. There are multiple valid solutions for any given block, only one of the solutions needs to be found for the block to be solved.

3.2 Mine a new block to the blockchain

When a blockchain transaction is transmitted to a mining node, it is first validated according to the consensus rules of the blockchain network. If the transaction is valid, it is added to a pool of unconfirmed transactions. This pool is called a mempool. The mempool acts as a temporary store of transactions to be mined into the next block. Each mining node has its own mempool, and any given transaction can be included in more than one mempool if it is broadcast to more than one mining node (most transactions are also shared between miners).

A miner takes the transactions and includes it in the next block. The miner hashes the transactions into a Merkle tree structure and includes the resulting Markle root within a candidate block header. The miner then hashes this candidate block header to find a valid proof- of-work. A Merkle tree is a data structure in the form of a tree of hash values. In the context of the blockchain, a transaction is hashed to form a leaf node of the tree. The pairs of leaf nodes are concatenated and hashed to form a node in a higher layer of the tree. The pairs of nodes in this layer are further concatenated and hashed to form a node in a higher layer of the tree. This process is repeated until a single node — called a root node or the Merkle root — remains.

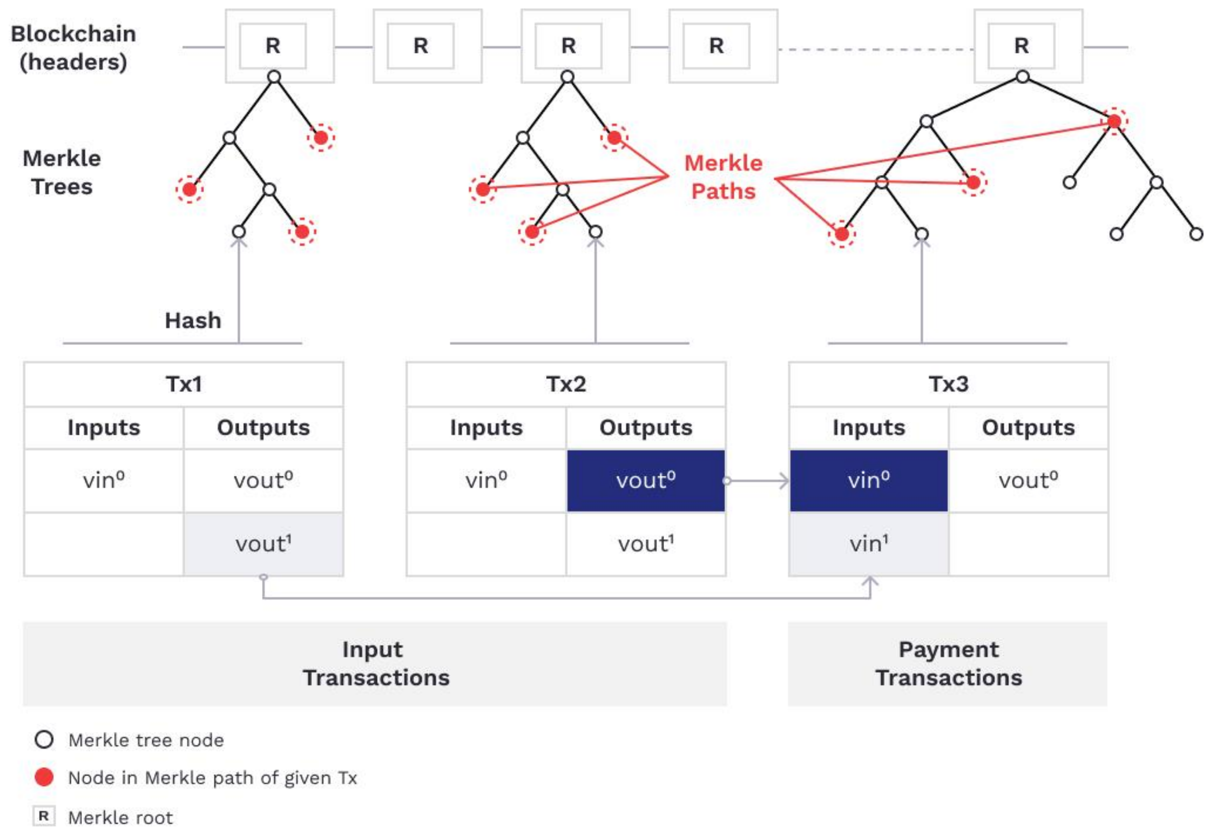


Figure 2: Merkle tree example[1] (Source: see Bitcoinsv.io)

[1] https://wiki.bitcoinsv.io/index.php/Simplified_Payment_Verification

A hash function converts a string of data of arbitrary length into a fixed length with unique value that is called the hash value or a hash digest. Hashing is a one-way function. That is, it is not feasible to determine what the input data is by looking at the hash value produced from it. Also, it is important to run the same input data through the same hash function and reproduce the same hash. The blockchain protocols use the SHA-256 hashing algorithm. The certain blockchain protocols use the SHA-256 hashing algorithm twice. That means the candidate block header is passed through the same hashing algorithm two times.

A valid transaction is found by hashing the candidate block header (in combination with other data) until the result is less than another value (called the target value).

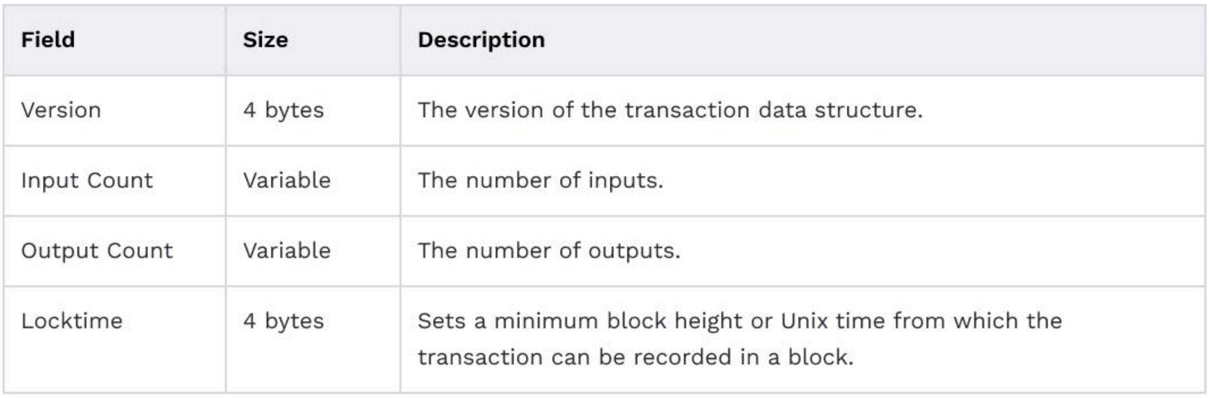
A mining node must add the additional information to the candidate block header to change the hash value. The mining nodes use two nonce fields to alter the value to be hashed, and thus alter the resulting hash value. A coinbase transaction is a transaction created and included in the candidate block by the mining node. Each field includes a counter parameter that can be incremented. The hash function cycles through all values of the first nonce field, and then increments (or otherwise changes) the second nonce field before going through all permutations of the first nonce field again. Incrementing the second nonce field involves recomputing the Merkle root as it modifies the hash of the coinbase transaction, which is included in the Merkle tree.

When a mining node finds a valid hash for a block (that is, a candidate block header that hashes to a value less than the target value), it broadcasts the new block to the rest of the blockchain network. The other nodes on the network accept the new block only if all the transactions in it are valid UTXOs that have not been consumed in an earlier block. Every block is timestamped and references the hash of the block preceding it, thus resulting in a chain of blocks called a blockchain.

3.3 Blockchain transaction components

The following table is the schematic representation of the structure of a transaction according to the Bitcoin protocols. A transaction is made up of the serialized set of data fields, which are represented in hexadecimal.

Table 1: Structure of a transaction



Field	Size	Description
Version	4 bytes	The version of the transaction data structure.
Input Count	Variable	The number of inputs.
Output Count	Variable	The number of outputs.
Locktime	4 bytes	Sets a minimum block height or Unix time from which the transaction can be recorded in a block.

3.3.1 Input

A transaction has one or more inputs, each input references an output of a previous transaction. Each transaction consumes as input 1-n outputs of one or several previous transaction(s). Each input includes an unlocking script. If the unlocking script includes the correct data, it will unlock the referenced output and consume it. An unlocked output of a previous transaction or the amount of the digital asset previously locked to the output can be spent by an output of the current transaction. The unlocked amount of the digital asset can be spent in its entirety by a single output or distributed across more than one output of the current transaction.

Table 2: Structure of input(s)

Field	**Size**	**Description**
TXID	32 bytes	The reference to the previous transaction.
VOUT	4 bytes	Selects an output of the previous transaction.
ScriptSigSize	Variable	The length of the unlocking scripts.
ScriptSig	Variable	A script that unlocks the selected output.
Sequence	4 bytes	

3.3.2 Output

A transaction has one or more outputs, which together distribute the total amount of the digital asset unlocked by the inputs. The sum of the output values is usually less than the sum of the input values, and the difference is the fee to the mining node that records the transaction in a new block.

An output can be one of the following:

- Spendable output (i.e. Spend money): A spendable output includes a locking script (referred as ScriptPubKey), which defines one or more conditions that must be satisfied to unlock an input in the future transaction.

- Unspendable output (i.e. Op_Return): An unspendable output does not contain a locking script that can be unlocked. If the unspendable output contains a locking script, it can cause failure in the execution of the locking script.

Table 3: Structure of output(s)

Field	**Size**	**Description**
Value	8 bytes	The value of the output.
ScriptPubKeySize	Variable	The length of the locking script.
ScriptPubKey	Variable	A script that locks the output.

4 Token “Smart Contract” Deployment Model

To understand BSV token solutions, it is required to provide an overview first, before diving into the details of STAS.

4.1 Token Contracts on BSV

BSV network supports a wide range of smart contracts that are based on using the scripting language known as Bitcoin script. Many concepts of tokenization or issuance of smart contracts follow a common logic where tokens that are issued effectively create new token units on layers on top of BSV and require different layers of logic to be run on top of it (see appendix for examples of token architectures).

In the past, Bitcoin script was in many ways perceived as an unfriendly base layer blockchain to deploy smart contracts on as it has limited capabilities over the state and is mainly used to create, store, and transfer an asset using Bitcoin. This is what was supposed to be solved with Ethereum. Some contracts are inherently stateful in that they require contracting parties to interact in multiple stages and depend on time-varying states, such as on-chain corporate action voting for security tokens.

One of the most powerful inventions on the BSV blockchain was the introduction of the concept for the usability of OP_PUSH_TX that allows deployment of smart contracts. An OP_PUSH_TX contract in a locking script is divided into code and data allowing it to become readable on-chain through a blockchain explorer such as whatsonchain.com. OP_PUSH_TX allows inspection of the entire transaction inside a contract, including all inputs and outputs. This opens boundless possibilities for all kinds of smart contracts on Bitcoin and can be built in using any token Layer architecture (Layer-0: on-chain or native Bitcoin script and all information is stored on-chain; Layer 1: integration layer that uses some off-chain features to store information; Layer 2: application layer that uses agents or virtual machines to store information off-chain). A more detailed explanation is given in the Appendix section. This is the case for Layer 1 and Layer 2 solutions where the token logic is represented off-chain by running proprietary code on proprietary servers, and thus a certain amount of trust needs to be assigned to the platform itself.

4.2 STAS Tokenization Script

The benefits of STAS tokens are that they do not require any additional processing logic beyond that of basic Bitcoin script itself. Given the fact that the tokens themselves are Satoshis we can have many things that previously required third-party actions to be performed automatically by revealing the right set of keys. Those keys would allow the locking / unlocking of a token from being issued, transferred, traded, and redeemed. With the proper identity / key management systems a third-party accounting and auditing firm could work directly off the public blockchain to do a company's year-end filing and tax reporting.

Additionally, since STAS operates as a regular Bitcoin output, wallets can implement standard protocols for creating digital invoices such as BIP270[2], which allows merchants to create invoices by way of a customized payment template. As STAS uses Bitcoin as a ledger, complex payment options can be made using STAS-based stable coins. For example, one feature of STAS allows for portions of a payment to be made in BSV or a token representing fiat. This innovation in payment flexibility allows for maximum useability in the real world.

Finally, because the STAS standard uses Bitcoin tokens, issuers do not need to create tokens outside of Bitcoin. Instead, tokens implemented using STAS have the same durability, inter-operability, and global scalability that Bitcoin does. They stand to be the only tokens that will outlive the token platform providers and the issuers, too. That durability is because the tokens are on the public ledger, allowing anybody to view them with a blockchain explorer. In the case of STAS-enabled transactions, the blockchain explorer used would be whatsonchain.com from TAAL.

STAS is most suited for use in the creation of digital twins. Even if the issuer of the token (custodian of the asset) were to become insolvent, the STAS token, its rightful owner, its chain of digital signatures, and historical ownership would still exist, just like the real-world asset that it represents. Just in the form of a token, rather than a paper-based contract.

The STAS token is not a token layer for a specific use case, it is the technology that will work for all tokenized use cases. And because it requires no specific server code, and only small modifications to wallets and block explorers as well as APIs to support them, they stand to be the most widely adopted standard, and a compelling use case for the utility value of the Bitcoin network itself.

[2] https:/tsc.bitcoinassociation.net/standards/invoice_based_payments/

4.3 How to use STAS

Implementing STAS successfully follows a two-step procedure. First is the implementation of the token schematics, also known as tokenomics. Tokenomics refers to the possible movements and uses of Sats that have been “transformed” (or actualized) from native Bitcoin denominations into a “minted” token with a legal issuance contract. A user who wants to mint / issue a STAS token must lock up the defined amounts of Sats in a wallet address that represents the actual minted STAS tokens. Thus, making it impossible for any wallet to remove or double-spend the native Sats. It is the same principal with a working model that so many colored coins in the past have failed in achieve. (“Colored Coins" loosely refers to a class of methods that represent and manage real-world assets on top of the Bitcoin Blockchain.)

The second part of the standard governs the issuance transaction and how it is sequenced. The sequencing occurs such that all supporting wallet and public node infrastructures understand how to interpret and support the STAS token ecosystem, by providing the needed authenticity (origin) checks.

As mentioned STAS is implemented by native Bitcoin script. These scripts are in the form of transaction templates, which impose restrictions on the usability of a UTXO or unspent transactional output, containing a certain number of Sats.

4.3.1 Special Restrictions on Outputs via Smart Token Scripts

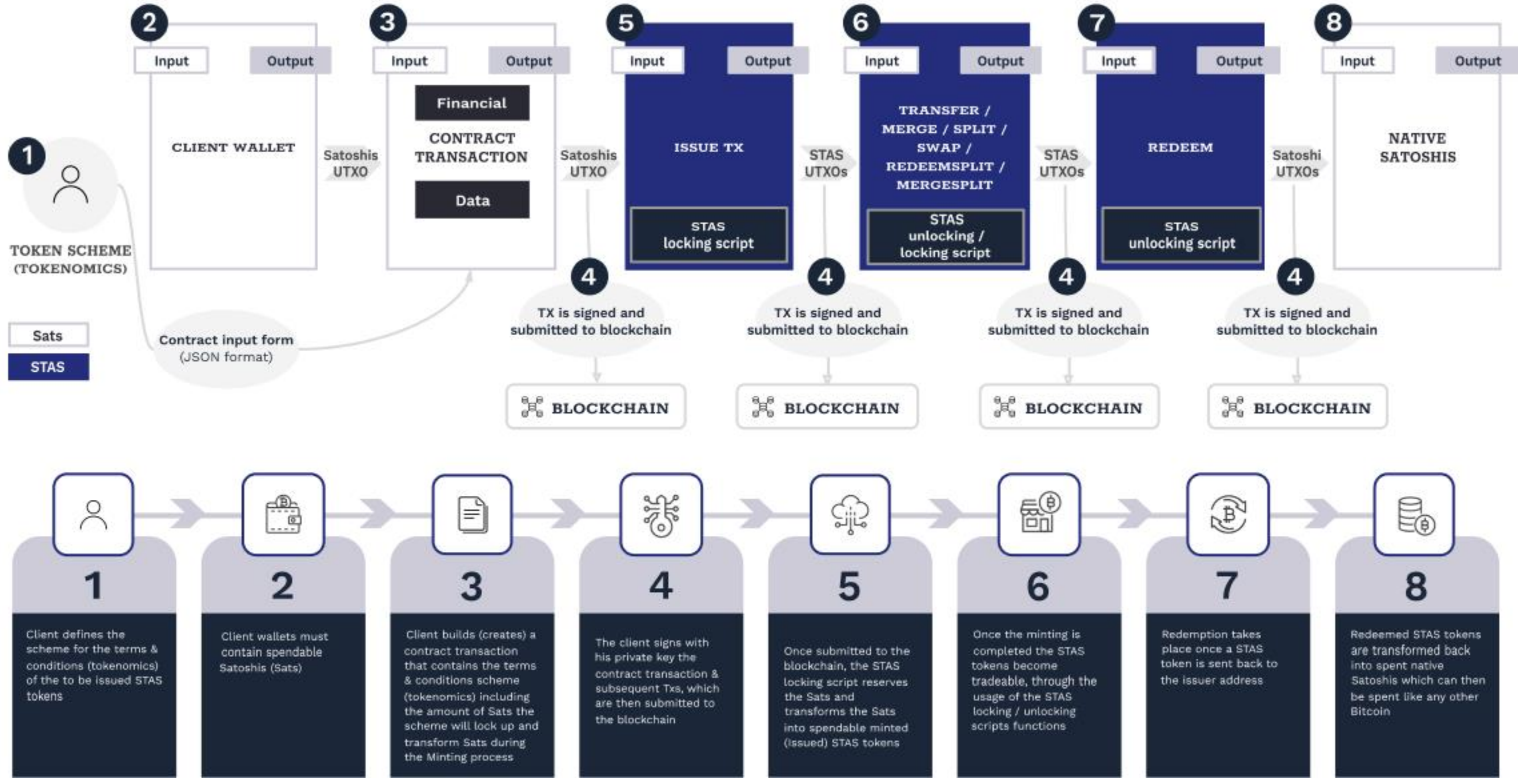
The basic notion is to lock or restrict the usage of transaction outputs through Bitcoin script. That protocol requires these outputs to continually persist some associated token meta-data. In addition, these restrictions mean the outputs cannot be unlocked unless they are sent to a pre-determined redemption address. The metadata simply consists of a link to the information of the issuer, the token ID, and a redemption address.

More complex types of STAS templates will be able to facilitate more sophisticated transfer modes and token logic. Those additions will be introduced as STAS grows. Some of the coming features requires enterprises and regulated companies to include the following token functionalities: auto-expire, callability, drag-along contract, token recovery, collective signatures, allow lists, exchangeability, whitelisting/blacklisting, voting, authenticity of a person's identity, integration with other token standards, token swapability of SH-256-based tokens (from BSV, BCH, and BTC, as well as other tokenization protocols), wrapping and interoperability with other blockchains in the future.

4.3.2 STAS Token lifecycle explained

To use the STAS script, TAAL has developed an SDK in Go and JavaScript that allows the functionalities of the STAS script to be integrated and can be utilized to build a transaction to issue, transfer, trade, or redeem a token using the STAS locking and unlocking script.

The SDK gives developers the ability to build any type of application on top, using the high-level functions rather than the low-level locking and unlocking scripts.


1
TOKEN SCHEME (TOKENOMICS)
Sats
STAS
Contract input form (JSON format)
2
Input
Output
CLIENT WALLET
Satoshis UTXO
3
Input
Output
Financial
CONTRACT TRANSACTION
Data
Satoshis UTXO
5
Input
Output
ISSUE TX
STAS locking script
STAS UTXOs
6
Input
Output
TRANSFER / MERGE / SPLIT / SWAP / REDEEMSPLIT / MERGESPLIT
STAS unlocking / locking script
STAS UTXOs
7
Input
Output
REDEEM
STAS unlocking script
Satoshi UTXOs
8
Input
Output
NATIVE SATOSHIS
4
TX is signed and submitted to blockchain
BLOCKCHAIN
4
TX is signed and submitted to blockchain
BLOCKCHAIN
4
TX is signed and submitted to blockchain
BLOCKCHAIN
4
TX is signed and submitted to blockchain
BLOCKCHAIN

1 Client defines the scheme for the terms & conditions (tokenomics) of the to be issued STAS tokens

2 Client wallets must contain spendable Satoshis (Sats)

3 Client builds (creates) a contract transaction that contains the terms & conditions scheme (tokenomics) including the amount of Sats the scheme will lock up and transform Sats during the Minting process

4 The client signs with his private key the contract transaction & subsequent Txs, which are then submitted to the blockchain

5 Once submitted to the blockchain, the STAS locking script reserves the Sats and transforms the Sats into spendable minted (issued) STAS tokens

6 Once the minting is completed the STAS tokens become tradeable, through the usage of the STAS locking / unlocking scripts functions

7 Redemption takes place once a STAS token is sent back to the issuer address

8 Redeemed STAS tokens are transformed back into spent native Satoshis which can then be spent like any other Bitcoin

Figure 3: STAS Token Lifecycle

A typical lifecycle to issue a STAS token consists of the creation, issuance, transfer, and redemption of a token. However, there are multitudes of use cases that may require the ability to also merge and split tokens as a whole or in a fractionalized form. The above (Figure 2) illustrates an example lifecycle for any type of token issuance.

1. The client defines the token scheme for the terms and conditions of the STAS tokens to be issued.
2. The system ensures that the client's wallets contain the spendable Sats.
3. The client builds a contract transaction that contains the terms and conditions that includes the amount of Sats that the script will lock and transform into tokens during the minting process.
4. The client signs the contract and subsequent transactions with a private key, which are submitted to the blockchain.
5. Once submitted to the blockchain, the STAS locking script locks the Sats. That step transforms them into spendable, minted (issued) STAS tokens.
6. Once the minting is completed, the STAS tokens are transferable using the STAS locking or unlocking scripts functions.
7. The redemption takes place once a STAS token is sent back to the issuer address.
8. The redeemed STAS token is unlocked from the STAS locking script and returned to a native Satoshi, which can be spent in the normal way or minted again under a new contract transaction.

For a full list and explanation of the various functionalities of the STAS SDK, please refer to the appendix, where each functionality is explained in more detail.

To date there is still no common language when it comes to what comprises a Layer 0 (native blockchain or "on-chain" solutions), 1 (Integration), 2 (Applications) or even Layer 3 (digital apps also called dApps) architecture and why any of them should be built on a certain token solution and / or protocol. Depending on the website visited [3]BSV, ETH, SOL, ADA, and others tend to only be referred to as existing as a Layer 1 solution. Herein, we refer to the fact of BSV forming the native Bitcoin scripts that are "on-chain", which allows the building of token solutions on top of it. In other words, BSV is the underlying protocol that makes it possible to build token solutions on which smart contracts can be issued. For this purpose, we explain the different types of Layers.

- **Layer 2**: All the token logics are executed on external servers, and only a proof-of-execution or events is on-chain. The Bitcoin is used only as a data-carrier. For example, see Tokenized.com and Run.sv.
- **Layer 1**: The token logic is on-chain; however, the balance representation and tokens exist in additional data structures, unrelated to and decoupled from the Bitcoin. For example, see sCrypt.
- **Layer 0**: This layer is like Layer 1, except that the tokens are the Bitcoins. The conferring meaning to a Bitcoin native token (Sats) is an open standard, and restrictions are applied to Sats through the native Bitcoin script. For example, STAS.

In this white paper we have included the various types of technology architectures that can be utilized to build out a blockchain use case and explain the difference of the layers in more details, please refer to the appendix.

[3] https://zycrypto.com/blockchain-layers-explained-what-are-they-and-why-do-we-need-layer-solutions/

4.3.3 STAS Layer 0 Architecture

In the Layer 0 architecture, the token logic (that governs the transferability of tokens) exists in Bitcoin. Thus, there is no need for private servers running the token transfer logic, and the wallet applications need simple modification to recognize the balance of a specific token.

The balance and accounting logic, and all the need to synchronize the balances depend on the public blockchain as the ledger. Bitcoin has been shown as a scalable way to ensure a global synchronized ledger.

In this layer, there is a need for of an authenticity check, which is an extraction of evidence of genuineness from the blockchain, which can run by installing local open-source software by the applications or wallets, or for profit by the third-party businesses of validation through public data extraction. In the case of TAAL clients it is possible to use the toolsets provided (SDK) and the whatsonchain.com explorer. By using our API endpoints, it is possible to fetch information related to ensuring if in fact a token is valid and if it represent a token from the desired origin. The technology of such a validator is the same as the current blockchain explorers and it is expected that most blockchain explorers will optionally provide this service to the ecosystem, alongside enterprise solution providers.

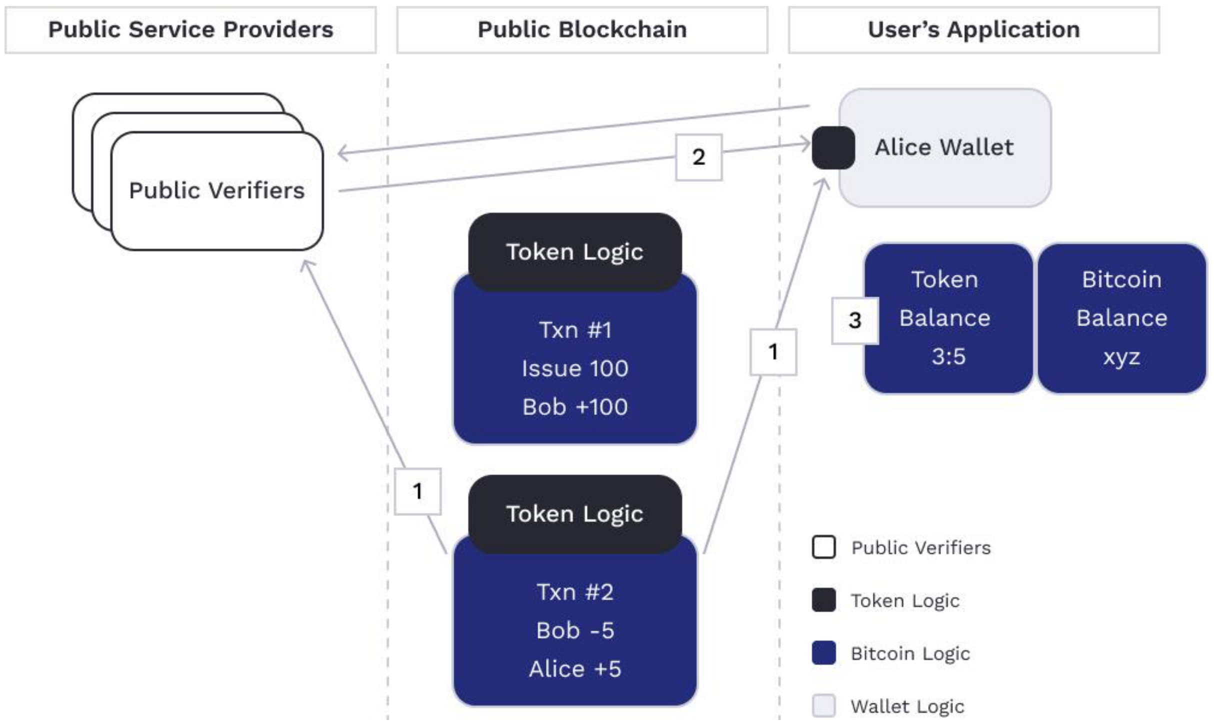


Figure 4: Layer 0 architecture of STAS

4.3.4 Locking TX Outputs

Bitcoin transactions use locking and unlocking scripts, which are executed together to verify a transaction. A locking script is a spending condition specified in the transaction output, and an unlocking script satisfies this condition when the two scripts are executed together.

To fully understand the STAS locking / unlocking scripts we want to explain the components and how they work together to allow a user to work with the STAS token scripts.

As outlined already, restrictions are implemented by way of creating outputs using a certain concept also known in Bitcoin language as a "template". For STAS we implemented the concept / template named P2STAS or "Pay to STAS". This is similar to the P2PKH concept. The P2PKH concept stands for "Pay to Public Key Hash". The Public Key Hash is one of many formats of the Bitcoin address, the other being "Pay to Script Hash". At the most basic level, P2PKH means "pay to this Bitcoin address"4.

[4] https://wiki.bitcoinsv.io/index.php/Bitcoin_Transactions#Pay_to_Public_Key_Hash_.28P2PKH.29

Fundamentally, the template for a P2STAS Token output, consists of the following components:

- variable component (see example below 202)
- constant component (see figure below 204)

The rules are enforced by code in the constant component called the token mechanics subcomponent.

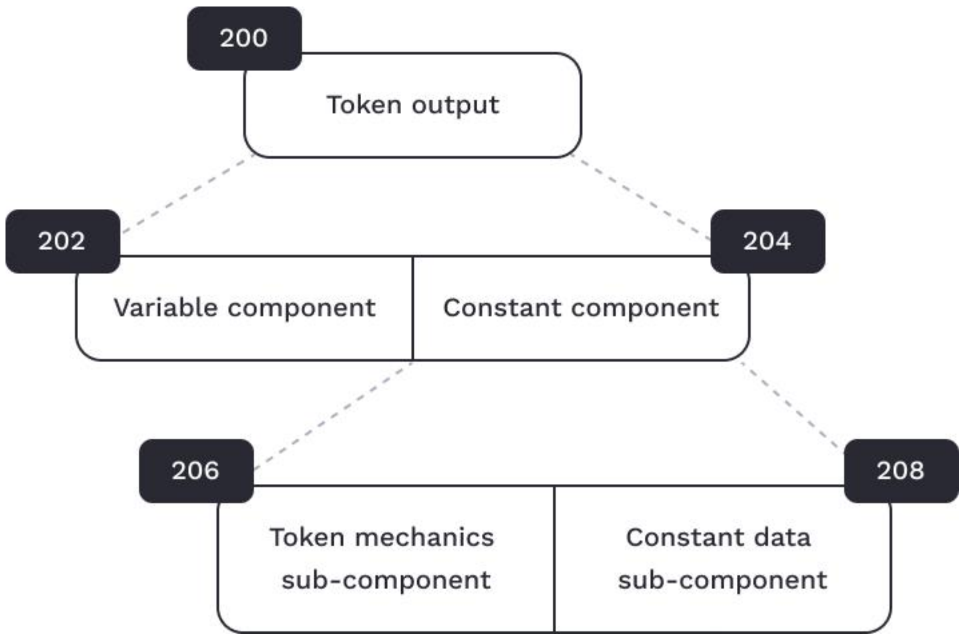


Figure 5: Components of a STAS “Smart Token” locking script; a P2STAS template

The **variable component** is identical to regular Bitcoin’s payment template that passes on possession of the tokens through ECDSA usage (Elliptic Curve Digital Signature Algorithm (ECDSA) and offers a variant of the Digital Signature Algorithm (DSA), which uses elliptic curve cryptography.[5]

The constant component as shown in the figure above consists of two sub-components:

- token mechanics sub-component (See figure above), it contains the rules enforcing the code; and

[5] https://wiki.bitcoinsv.io/index.php/Elliptic_Curve_Digital_Signature_Algorithm

- constant data sub-component (See figure above), which includes the token meta-data that must now be persisted and carried over through token transfers. The constant data-sub-component prevents a token information from being 'lost". Tokens are lost when a wallet that does not support the protocol strips the metadata off the coins on spending, thus losing the information and destroying the token.

According to one aspect disclosed here, a computer-implemented method of sending digital tokens using blockchain transactions is provided. Each token is represented by a single unit of an underlying digital asset (i.e.: native units) of the blockchain. The method comprises generating a first token transaction and transmitting the first token transaction to the blockchain network. The first token transaction comprises a first token output and the first token output comprises a first token locking script and a first token amount. The first token locking script comprises a variable component and a constant component.

The variable component comprises a first payment address embedded in a payment template.

The constant component comprises a token mechanics sub-component. When executed alongside an input script of a spending transaction (with the input script including a respective locking script and an amount locked in a previous transaction output that is being spent), the token mechanics sub-component is configured to perform the following operations:

A first operation comprises obtaining one or more data pairs from the input script of the spending transaction. Each data pair comprises: i) at least a respective payment address included in a respective locking script of the spending transaction outputs; and ii) a corresponding amount of the underlying digital asset locked by the respective locking script of that output.

Another operation includes verifying that one or more outputs of the spending transaction of a respective locking script comprises a) a respective payment script template that includes a predetermined payment address; or b) a respective variable component with a respective payment address other than the predetermined payment address, followed by the constant component.

For those one or more outputs of the spending transaction, another operation requires the verification that a total amount of the underlying digital asset locked by the respective locking scripts (of the one or more outputs) is equal to the first token amount. The token mechanics sub-component is configured to fail during execution if any of verification steps fail.

4.3.5 The Variable Component

The variable component is the part of the token locking script that allows tokens to be moved to another party. The variable component includes payment template containing (e.g., surrounding) a payment address. The payment address may be based on a public key of a recipient party. For instance, the payment address may be a public key hash (PKH) address, that is, a hash (or double hash) of a public key.

This part is intentionally at the beginning of the script and is just a regular P2PKH template. The reason for this placement is to allow for maximum possible compatibility with existing wallets and browsers, providing their searches by specific raw ECDSA addresses wrapped with this fixed template. This is the feature that designates the current holder of a token.

4.3.6 Constant Component (Token Mechanics)

The constant component is the token's code. It features include: 1) self-immutability; 2) impossibility of its own omission; and 3) most importantly, preservation of token amounts. That means it locks the tokens, letting their amount (whole or in part) neither to be leaked as miners' fees nor spent to any other than its own smart-locking script format outputs (unless redeemed to a specified address, encoded upon issuance).

For a token to be spent, assigned, or otherwise transferred (and, therefore, continue to function as a token), a spending transaction must have in its next output a locking script of the same format as a previous transaction output (UTXO) that is being spent. Like the token transaction output, which is being spent, the spending transaction, in order to be able to be successfully transmitted, must include an output having a locking script that has the same constant component of the new token output. This constant part can neither be changed nor omitted through a token's lifetime until its redemption. Meaning that if you want to spend a token, it's only doable if the next UTXO has the same locking script (P2STAS) as the UTXO being spent has, apart from new owner address update.

Spending by a transaction with a UTXO via regular Bitcoin locking scripts (P2PKH, P2PK, and so on) fails, unless it is a P2PKH type to the redemption address set in the contract upon its issuance. Meaning, only the issuer who determined the constant component of the token, can designate the redemption address that can convert an asset representing STAS back into its original native bitcoin sats (by releasing underlying assets, e.g., USD, gold, and so on), which removes the need for administrative servers.

In summary, it makes native tokens regular usage impossible while enforcing newly defined ones, thus changing their nature and behavior.

This effect means that each single unit of the underlying digital asset now represents a redefined single token and stops its regular functioning. To continue with the example of the Bitcoin blockchain, a single Satoshi is now redefined as a single token. The owner of the token(s) cannot move the token(s) to another user unless the locking script contains the same constant component, which enforces new token mechanics. All that can be changed in the locking script is the variable component, which may be a standard template (e.g.: used in the same manner as in native tokens of the underlying asset) responsible for transferring control. That is, it enables the current controller to move some or all the token(s) to the next controller, according to an address included in a standard template.

Put another way, unlike previous attempts that rely on metadata attachments to represent a token, the tokens of the STAS invention are single units of the underlying digital asset that have been reconfigured to function as distinct entities, operating by different rules (encoded in themselves).

The tokens can be converted back into the underlying digital asset, i.e. to be used once again as the native token, if and only if the tokens are subject to satisfying particular conditions encoded in the above constant part of the script (e.g.: a movement to predetermined redemption address). Only a hardcoded user or authority (such as the entity that has control over the redemption address) has the ability to turn the tokens back into the regular underlying digital asset's nature, restoring their original functioning, thus "destroying" the redeemed token.

The amount of Satoshis that represent a token upon spending must be one of the following:

- unaltered — in case of plain transfer (either an envelope or a single-token type).
- split in two — in case of spending TX having two outputs (partial spending with change) each with locking scripts identical to the one being spent apart from the owner address (envelope type only).

What constitutes the state pushed in these stateful TXs is the controlling address updated each hop AND (only in case of envelope-type with multiple tokens) the number of tokens (that may be reduced through splitting).

In other terms, the constant component data consists of a link to the issuer's information, the Token ID, and the redemption address.

Each Sat transformed to a STAS now represents a token and ceases its regular functioning, although it can represent something cheaper or more expensive than 1 Satoshi itself. A user cannot move tokens unless this exact template, preserved as a locking script in sending a transaction output, exists and then all that can be changed is the owner's address in the variable part.

This is unlike "colored coin" implementations, which make coins differ from the others only by carrying some metadata attachments on them. This technique locks Satoshis to a specific spending pattern and locks that metadata to a spending template. It can be considered a form of reversible burning of the Sats.

As P2STAS requires P2PKH at the beginning of the template, a wallet needs to concatenate (to be able to spend the transaction) to the sub-template of the whole constant component of the STAS token output that is being spent into the constant section of the new smart token output. If this cloning of the constant component of the previous STAS output is not done, then the transaction validation/transmission will fail. The script in the constant section includes code that checks that it is replicated and includes no modifications, and that the only way a STAS can be paid to an output which is not itself a STAS smart token is if it is sent to the redemption address, hardcoded immutably in script itself.

In summary, the movement of STAS outputs can be visualized below:

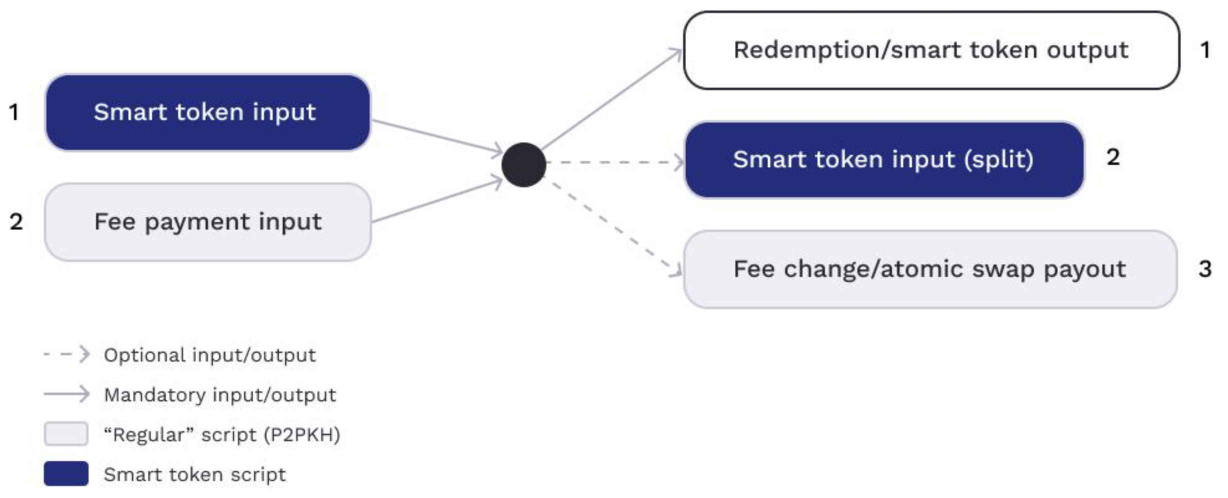


Figure 6: Valid transactions involving STAS outputs

4.3.7 STAS token Transaction Chain and Authenticity Checks

As mentioned earlier, the other required aspect of the standard is the way in which authenticity of any given STAS token can be verified by network services that support the STAS standard. This check can be performed by any service that runs a fully validating Bitcoin node (if not using specific software that keeps only specific token relevant parts of the blockchain) that are offered by TAAL, Mempool or Mattercloud and others.

This includes all miners and transaction processors, but the creation of blocks is not a requirement for such a service provider to provide the authenticity check for STAS tokens. As the STAS standard states any issuance transaction should follow a certain format and include certain information fields (such as the redemption address) and follow an initial contract transaction. It is a simple task for any STAS verifier to keep track of which outstanding UTXOs are genuine STAS (issuance of which occurred on the blockchain from the address and symbol combination that is immutably hardcoded in every token) out of the total UTXO set.

This can be done in one step (O(1) complexity). This simple method allows for the greatest scalability as it avoids any need for creation of additional indexing systems for wallets/services, which at scale will become an unnecessary and burdensome challenge. Instead, it extracts the data from the existing native system of Bitcoin itself, making it naturally compatible with Bitcoin's own scalability. Generally, for Bitcoin the scalability is achieved by the fact that a node only needs to keep track of the valid UTXO set, and not the entire history of the blockchain, to validate transactions.

Based on the STAS standard, it also becomes possible that a token is supported by any wallet that supports Bitcoin, with minimal addition of tagging a subset of the total UTXO set. This allows for the greatest possible interoperability between token ecosystems and platforms, as specialized wallets are not required.

4.3.8 STAS Transaction Sequence

STAS tokens are created by transforming (actualizing) Satoshis (Sats). This is done by creating a contract transaction, followed by an issuance transaction. The contract transaction is a normal transaction, and it simply pre-allocates the number of Sats to be transformed into STAS. In the contract transaction, the issuer sends the Sats to the issuance address, which is also part of the Token ID that becomes immutably set in the STAS tokens that are issued. Next, the issuance transaction, is one that is paid to the first recipient of the tokens, and must be made from that issuance address, which at the same time is the Token ID and symbol (e.g.: Protocol ID). This is also known as the "origin" transaction for the given minting of the token. This origin transaction must be the ancestor to any valid STAS token of this issuance. If a token cannot be traced back to an origin transaction from an address that is the same as the redeem address used in a token ID, then it is not an authentic STAS token.

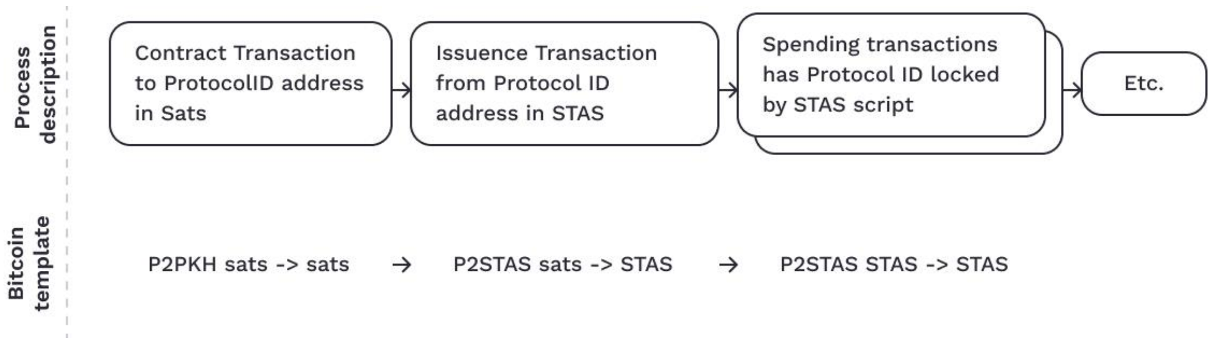

Process description
Contract Transaction to ProtocolID address in Sats
Issuence Transaction from Protocol ID address in STAS
Spending transactions has Protocol ID locked by STAS script
Etc.
Bitcoin template
P2PKH sats -> sats
P2STAS sats -> STAS
P2STAS STAS -> STAS

Figure 7: Creation of STAS tokens; transformation happens at the Issuance transaction

The below graphic shows on a high level a very basic flow of the transaction chains from the legal contract that was issued as a contract transaction and a separate issuance transaction in STAS tokens to Alice, who then allows Bob to purchase services by a merchant, who then redeems the tokens back.

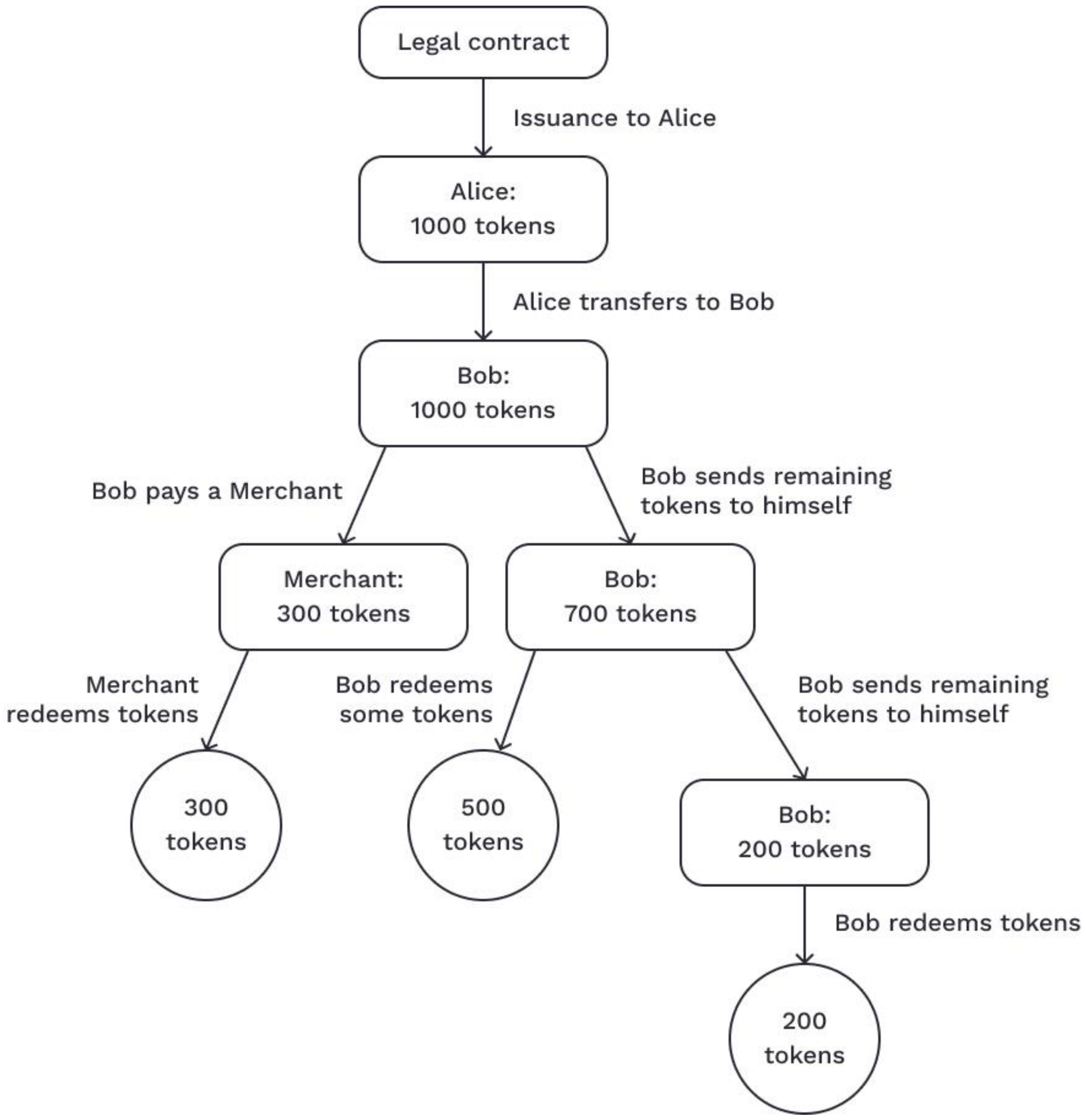


Figure 8: Typical flow of STAS contract and issuance (origin) transaction and transfers

4.3.9 A Public Overlay Network of Authenticity Verifiers

The final piece of the STAS infrastructure is a network of STAS-aware verifiers, or full nodes that can fill the requests of wallets to check whether any given STAS token is authentic. While the Bitcoin script governs the transferability of any authentically issued STAS tokens, it cannot by itself verify that it is a descendant of a valid issuance transaction.

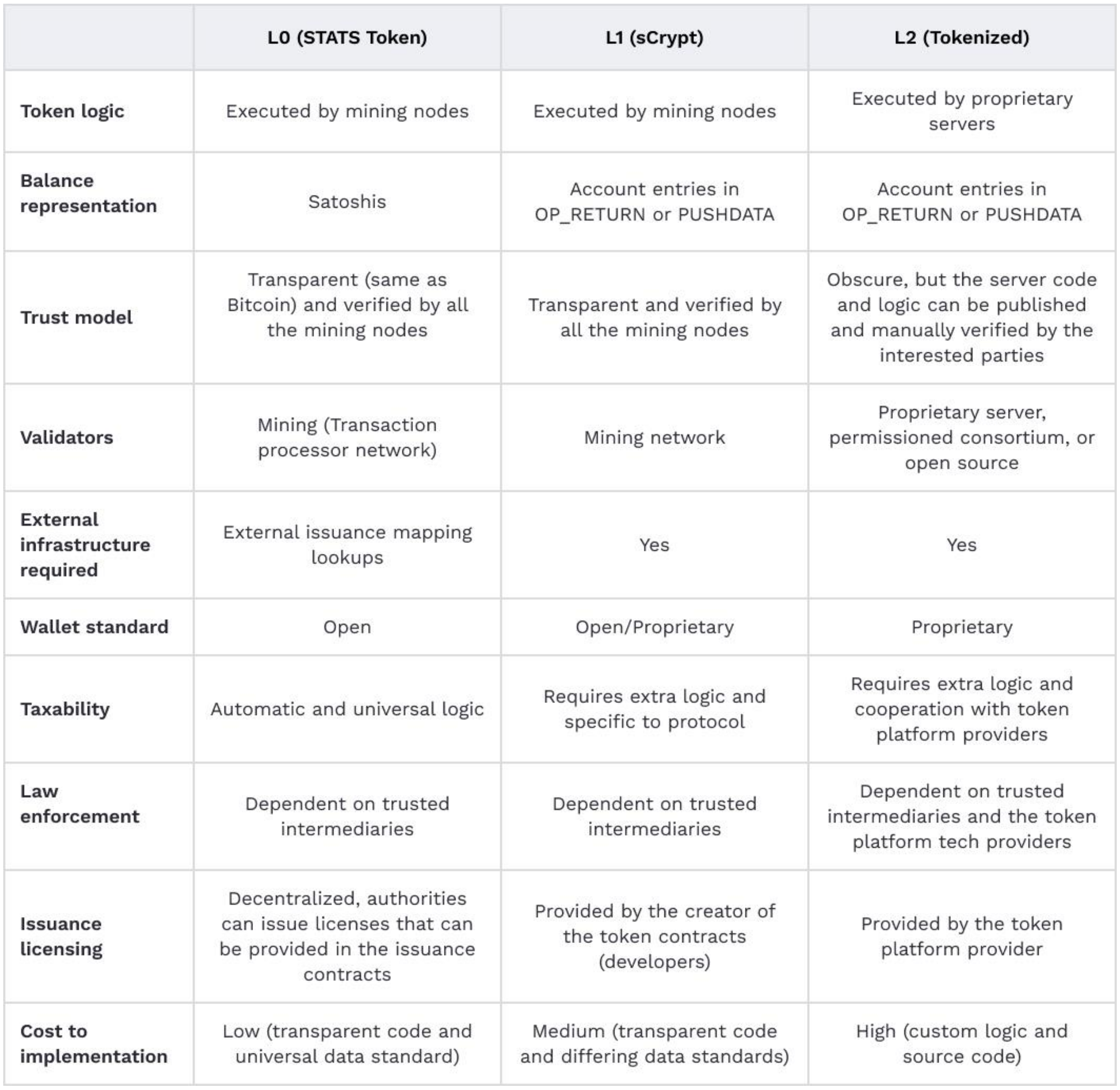
For example, take the case of tokens that represent car ownership used to operate smart cars in the future. A malicious agent may attempt to take some generic Satoshis, and voluntarily lock them with a copy of the same STAS locking script that the valid issuer of the car tokens used to represent the automobile ownership registrations. This perpetrator would then try to pass this token off to a buyer who he intends to trick into believing that it is truly the digital token linked to a specific automobile. The malicious agent is attempting to “steal” the asset. While the transaction that the malicious agent could construct would still be a valid Bitcoin transaction (nothing stops anyone from putting arbitrary limitations on the transferability of their own Bitcoins), this fake STAS token would not have a history that traces back to the original issuance transaction created with the public key of the registered issuer and at the same time be immutably written in token's script code. The system would recognize it as invalid.

4.3.10 Comparison of BSV Layer Token solutions

Opposed to other token blockchain protocols with one common smart contract standard, BSV allows that anyone can build their own token solutions on top of the native Bitcoin scripts. However, it has to be noted that for each different type of solution a deep understanding is required to develop a token solution, as it does follow a different logic than an account-based framework (e.g., Solana, Ethereum, Polkadot, Polygon, and so on.) over the blockchain protocol as explained in the introduction section of the white paper.

To ensure a broader understanding for the reader and to provide an overview over the different type of token solutions available on BSV we have compiled a comparison between the different token solutions. However, there is only one solution that does fulfill our definition of being a Layer 0 solution that directly utilizes the native Bitcoin script, namely STAS.

Table 4: Comparison of different Layer token solutions on BSV



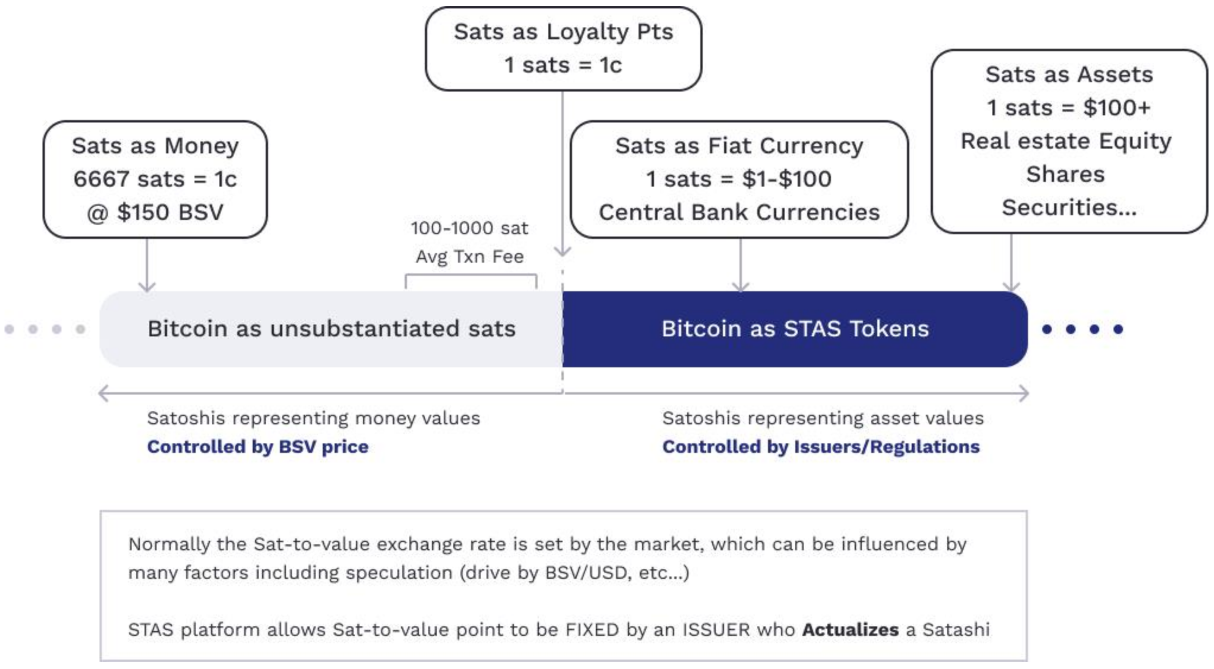
	L0 (STATS Token)	L1 (sCrypt)	L2 (Tokenized)
Token logic	Executed by mining nodes	Executed by mining nodes	Executed by proprietary servers
Balance representation	Satoshis	Account entries in OP_RETURN or PUSHDATA	Account entries in OP_RETURN or PUSHDATA
Trust model	Transparent (same as Bitcoin) and verified by all the mining nodes	Transparent and verified by all the mining nodes	Obscure, but the server code and logic can be published and manually verified by the interested parties
Validators	Mining (Transaction processor network)	Mining network	Proprietary server, permissioned consortium, or open source
External infrastructure required	External issuance mapping lookups	Yes	Yes
Wallet standard	Open	Open/Proprietary	Proprietary
Taxability	Automatic and universal logic	Requires extra logic and specific to protocol	Requires extra logic and cooperation with token platform providers
Law enforcement	Dependent on trusted intermediaries	Dependent on trusted intermediaries	Dependent on trusted intermediaries and the token platform tech providers
Issuance licensing	Decentralized, authorities can issue licenses that can be provided in the issuance contracts	Provided by the creator of the token contracts (developers)	Provided by the token platform provider
Cost to implementation	Low (transparent code and universal data standard)	Medium (transparent code and differing data standards)	High (custom logic and source code)

4.4 STAS Satoshi’s to Value explained

Bitcoin is an information and data carrier commodity. Its value has always been rooted in its ability to convey, store, record, attest, and trade information.

Instead of waiting until the utility of Bitcoin rises to the point where the value of BSV appreciates, trusted banking intermediaries could simply issue tokens and back them with real-world assets. These STAS tokens would be worth more than open-market Satoshi's but discounted by the credit risk of the issuer. The difference of the value of a fiat-pegged STAS over open-market STAS could also be the value of that trusted intermediary's credit.

Seen this way, transformed (actualized) Sats can have pegged values, and the value of Bitcoin appreciates based on its utility value of being a useful vessel for these substantiated tokens, after all there are only 2.1 quadrillion of these vessels available for actualization, and some of these vessels are busy being used as money itself. (Though at a significant risk-discounted value). The figure below illustrates the value spectrum of the different uses of Satoshi's, from fully as money to fully as asset vessels for substantiated tokens representing real-world assets.



Normally the Sat-to-value exchange rate is set by the market, which can be influenced by many factors including speculation (drive by BSV/USD, etc...)

STAS platform allows Sat-to-value point to be FIXED by an ISSUER who **Actualizes** a Satashi

Figure 9: The value representation spectrum of Bitcoin's Satoshi's (Sats)

Table 5: Comparison of value regimes for Satoshi use cases

Use Case	Metric	Satoshi Value	Comment	Notes
Representing Money	Value of 1 cent	6667 Sats	Effectively this means that we can charge as low as 1/6000 of a USD cent for a microtransaction (as scale moves the market price per transaction down)	As BSV/USD price appreciates this ability to represent microtransactions go down, at $10'000 BSV the smallest USD fee per transaction can only be 1/100 of a USD cent For Satoshi's value to cross into STAS, BSV needs to be > $650,000
Paying for transaction	Average Transaction fee	100 to 1000 sats	This means that most current market transactions are from USD cent 0.05 to 0.5	Looking objectively at block average transactions based loosely on market of 200 to 500 sat / kB
Representing Value	Value of 1 token	1 Satoshi	This means 1 Satoshi could represent a very high value item	N/A

4.5 Tokenization with STAS benefits

Utilizing the STAS script provides many benefits over the usage of other token solutions in BSV or compared to other blockchain protocols.

- Single toolset that offers read/write ability to create and sign transactions that wraps the STAS script through a tested and audited SDK that can issue, transfer, redeem, swap, create, and submit financial and/or data transaction.
- Deliver efficiency gains through the transfer of value without the need for trusted centralized intermediaries:
  - Smart contracts can reduce the cost of issuing and administering securities, and further reduces the cost of transactions.
  - STAS token contracts solution can keep track of and allow an infinite trace over the following non-complete list to facilitate the following: corporate actions (coupon or dividend payments, voting), escrow arrangements (release of funds), and collateral management (exchange of ownership interest).

- No infrastructure required for the issuer to use the technology.
    - STAS solution allows the Sat-to-value point to be fixed by an issuer who transforms (“actualizes”) a Satoshi.
    - With a small amount of effort any existing BSV wallet can be modified to recognize STAS tokens.
    - Reduces the risks of dependency to external servers (connectivity issues, hacking, or should the supporting company cease operations) by using STAS.
- In case of a company that has issued tokens using STAS becomes insolvent, it provides proof of ownership that can help to redeem the underlying values as the proof of ownership, based on the deployed tokenomics and legal claim details available on the blockchain.
- Any use case can be built on top with minimal requirements of integration or third-party dependencies as the token solution is simple to use and well-integrated to all services offered by TAAL.
- TAAL ensures that the STAS script, SDK, and APIs follow best practices and have undergone a third-party audit, performed by Trail of Bits. Please note that the audit report also entails internal relevant details, and thus the detailed report cannot be shared.

5 Appendix

5.1 Tokenization solution layers

Below are different types of tokenization layer architecture.

5.1.1 Layer 2 architecture

The difference between the main architectures is that the Layer 2 solutions require off-chain (private) servers to run the token logic and maintain balances. However, the Layer 0 solutions require only public validators to service an authenticity check that wallet users will require, which anyone can run.

The balance logic depends on Bitcoin. The wallet applications that support Layer 2 tokens need extensive changes to maintain balances and intimate knowledge of the L2 token protocol. The wallet applications depend on the private token servers for authenticity, balance updates, and trust.

The following image shows a Layer 2 architecture.

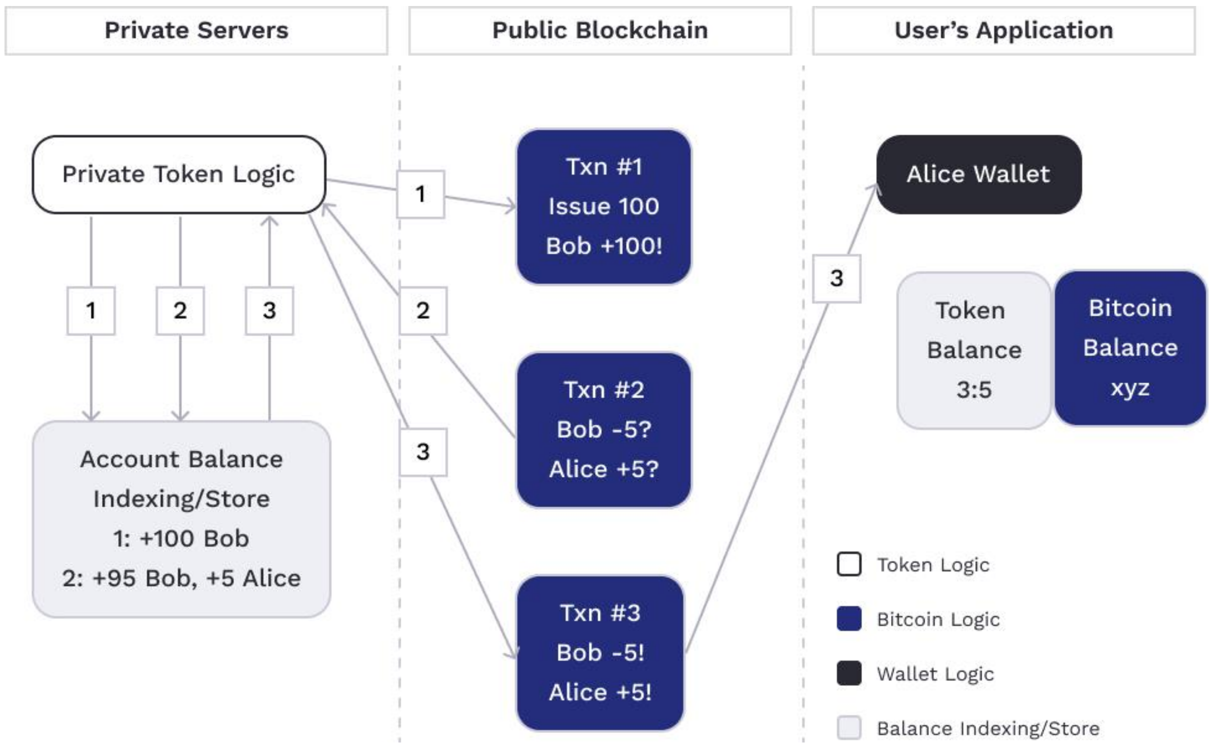


Figure 10: Layer 2 architecture

5.1.2 Hybrid architecture

The hybrid architecture has a private server executing the complex business token logic, which uses Layer 0 in the backend for the token representation. It combines the characteristics of both architectures and provides a migration path for the existing platforms with an existing UI and wallet ecosystem. This process can be implemented using threshold signatures to control the movement of the base Layer 0 token with the private token servers having a part of the signing keys.

The following image shows the hybrid architecture.

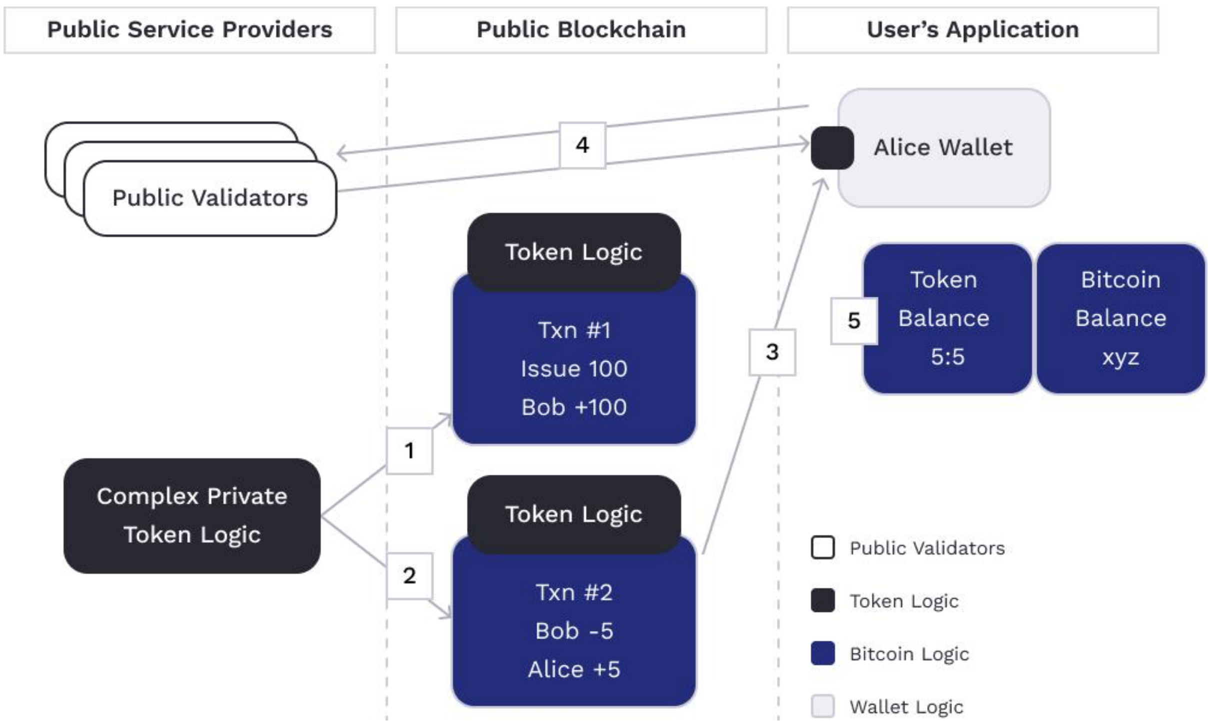


Figure 11: Hybrid architecture

5.1.3 Layer 1 architecture

In the Layer 1 architecture, the token logic (that governs the transferability of tokens) exists in the Bitcoin. Thus, there is no need for private servers running the token transfer logic, and the wallet applications need simple modification to recognize the balance as a specific token ID.

The difference in balances is stored in the transaction as data, and the wallets are required to decode and store their own balances. The wallets must be customized like the Layer 2 architecture.

The following image shows the Layer 1 architecture.

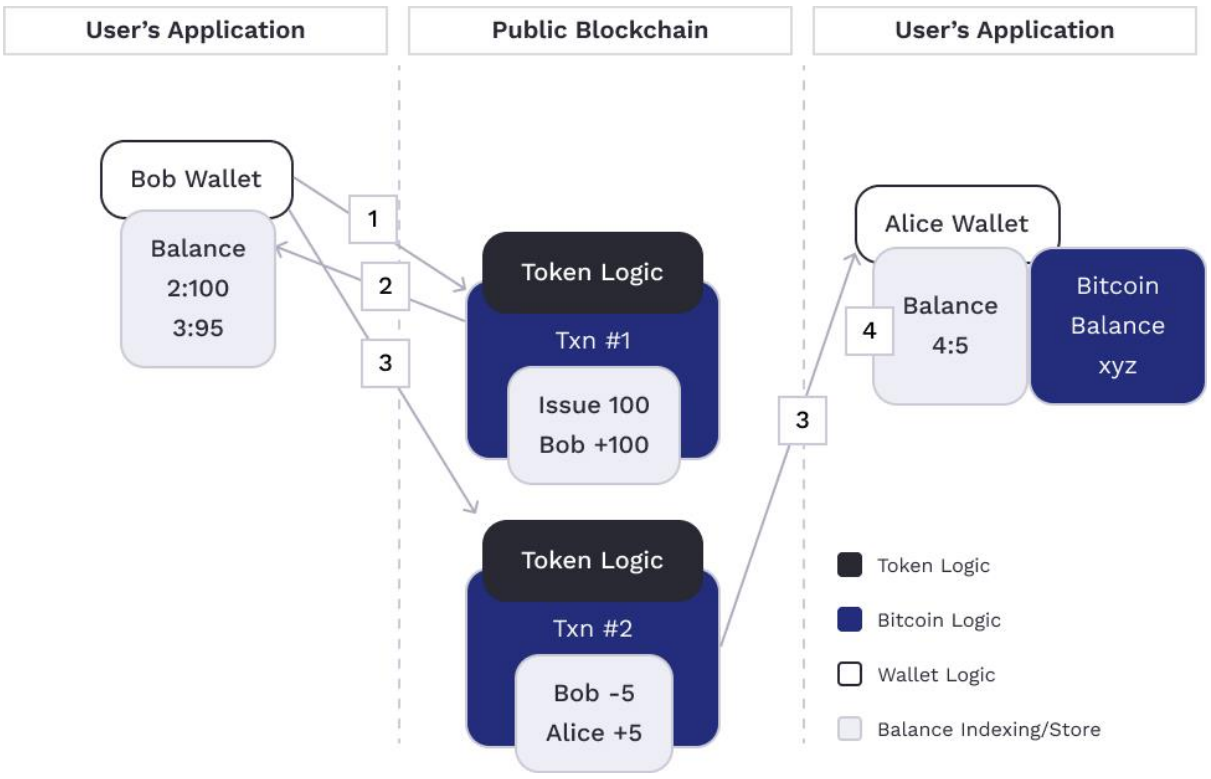


Figure 12: Layer 1 architecture

5.2 STAS functionality

The following table describes the STAS functionalities on a high level. The detailed usage of the functionalities can be found in the technical documentation.

Table 6: STAS functionality

Functionality	Description
STAS token scheme	Defines in a JSON file the Minting terms, which enables the minting of STAS tokens/coins, create business logic, reward structure, data validations, and so on. This must include: Name, TokenId, Description of the token, Issuer name, Country, Legal form, Email address, Website, Governing law, Terms of use, Icon, and Ticket symbol.
Contract transaction to BSV network	Parses the defined Minting token JSON scheme into a legal text document, which is published in the BSV blockchain network, and enables you to view the token terms, company details, and so on.
Issuance of STAS	The issuance blocks the Satoshis and mint STAS token with the defined value from the Token schemes. The Sats (input) are converted to STAS tokens/coins (output), which are sent (unspent) to 1 (Transferred) or 1:n (Split) to various BSV addresses using the BSV accounting method (called UTXO). Any token can be Split into 1:n, expect an NFT. The NFT tokens cannot be split or merged, and data can only be added as part of the issuance process. However, an NFT token can be transferred, swapped, or redeemed.
Swap	The atomic swap functionality is a trustless exchange of different assets without counterparty risk within the BSV eco-system. It enables a swap between any chosen token/coin against each other in using the same underlying BSV native coins within the network.
Split of STAS 1:n	When you send STAS to more than 1 addresses of the people, it triggers the splitting of the STAS, which can be in whole or fractional amounts to multiple recipients.
Transfer of STAS 1:1	In a STAS transfer transaction, the STAS are sent from a different address (spent) to a new address (becoming an unspent) STAS, which can be used to create new transactions (outputs).
Merge of STAS	The merge functionality enables you to send (input) and split multiple STAS tokens to one address (output). Instead of sending 10 separate tokens in 10 individual transactions, this functionality merges these into one transaction.
Redemption	The holder of a STAS token can redeem the STAS to the BSV native coins by sending back the STAS to the original address.
Mergesplit of STAS	The merge split functionality is required, if a STAS token address wants to send STAS tokens to multiple participants, which can be inefficient and trigger multiple transactions in using the transfer functionality,
Redemption split	The redemption split function enables you to redeem a portion of STAS token back in the native BSV coins by sending an amount to the original redemption address, and the remaining STAS tokens stay on the address.
API	An API and associated key enable you to broadcast the tokens/coins across the BSV network by using the transaction services.