

BSV-20

“first-is-first” style fungible token specification

First is first deployment and minting allows for a single use-case where a token is publicly mintable, by anyone, outside of the control of any token issuer.

Abstract

This proposal introduces a fungible token standard based off of the BRC20[1] standard on BTC introduced by Domo [here](#) but customized to work on BSV (Bitcoin Satoshi Vision). In order to avoid confusion with the BRC-20 standard on BTC, we are calling this standard BSV-20.

Motivation

The purpose of this proposal is to provide a standard that offers the same functionality that BRC20-BTC offers that works on BSV instead of BTC. A guiding motivation behind this standard proposal is to try and keep the standard as simple as possible.

Specification

This specification is meant to be as similar to the BTC BRC20 standard so it also follows the first is first approach. In order to deploy or mint a token, the process is almost identical to the protocol on BTC: you would create a transaction output with the data fields below and transactions are indexed on a 'first is first' approach with duplicates or overflows being invalid and ignored. The main difference between bsv-20 (this protocol) and bsv-20 protocol (on BTC) is:

- A content type of application/bsv-20 is used in place of text/plain. This change means a bsv-20 indexer does not need to parse every text inscription to test for embedded JSON content (and failing most of the time) in order to determine if an inscription is BSV-20 related.

Deploy

Notes

- The first deployment of a ticker is the only one that has claim to the ticker. Tickers are not case sensitive (DOGE = doge)
- If two events occur in the same block, prioritization is assigned via order they were confirmed in the block. (first to last)
- The first mint to exceed the maximum supply will receive the fraction that is valid. (ex. 21,000,000 maximum supply, 20,999,242 circulating supply, and 1000 mint inscription = 758 balance state applied)
- Number of decimals cannot exceed 18 (default)
- Maximum supply cannot exceed `uint64_max`

In order to deploy an BSV-20 token, you must make sure that that token ticker has not already been deployed and then create a TXO with the following data present in the script. It does not matter whether this UTXO is spendable or not.

Key	Required?	Description
p	Yes	Protocol: bsv-20
op	Yes	Operation: deploy
tick	Yes	Ticker: 4 letter identifier of the bsv-20
max	Yes	Max supply: set max supply of the bsv-20
lim	No	Mint limit: If letting users mint to themselves, limit per ordinal. If omitted or 0, mint amt is unlimited.
dec	No	Decimals: set decimal precision, default to 0

Example

To deploy the `ordi` token, you would create an inscription with the following json (with `ContentType: application/bsv-20`):

```
{
  "p": "bsv-20",
  "op": "deploy",
  "tick": "ordi",
  "max": "21000000",
  "lim": "1000"
}
```

Mint

In order to mint tokens of a specific BSV-20 token, you must make sure that that token ticker has already been deployed and then create a UTXO with 1 satoshi value as well as with the following data present in the script. This UTXO should be spendable in order for you to be able to transfer these minted tokens.

Key	Required?	Description
p	Yes	Protocol: bsv-20
op	Yes	Operation: mint
tick	Yes	Ticker: 4 letter identifier of the bsv-20
amt	Yes	Amount to mint: States the amount of the bsv-20 to mint. Has to be less than "lim" above if stated

Example

To mint `ordi` tokens, you would create an inscription with the following json (with `ContentType: application/bsv-20`):

```
{
  "p": "bsv-20",
  "op": "mint",
  "tick": "ordi",
  "amt": "1000"
}
```

Transfer

Tokens in BSV-20 are held in UTXOs, similar to native bitcoins. This is different from BRC20, which holds balance in an account model. In order to transfer tokens, you spend that specific UTXO and create new outputs the same way you spend regular Satoshis, but the output(s) must contain transfer inscriptions.

If more tokens are transferred in the output(s) than are available in the input(s) then the transaction is considered invalid and the tokens are burned. If less tokens are created in the outputs than are available in the input(s), the unallocated tokens are burned.

Using the same procedure as regular Satoshi transfers allows us to benefit from the parallelisation Bitcoin benefits from, where you can split a specific UTXO with a large amount into smaller UTXOs and spend those in parallel (the same way you could exchange a \$100 bill into \$1 bills and spend those in parallel) with no sequential bottlenecks that something like ERC20 (Ethereum) suffers from.

Key	Required?	Description
p	Yes	Protocol: bsv-20
op	Yes	Operation: transfer
tick	Yes	Ticker: 4 letter identifier of the bsv-20
amt	Yes	Amount of tokens transferred in this output.

Example

To transfer the ordi tokens that you minted as shown above, you would create a transaction spending the minting UTXO (providing the signature and public key normally to spend the P2PKH script) with an output (or many) with similar scripts with the following json (with `ContentType: application/bsv-20`), as shown below:

To mint ordi tokens, you would create an inscription with the following json:

```
{
  "p": "bsv-20",
  "op": "transfer",
  "tick": "ordi",
  "amt": "1000"
}
```

Inputs	Outputs
Signature Public_key (spending mint of 1000 ordi)	<pre>inscription({"p":"bsv-20", "op":"transfer", "tick":"ordi", "amt":"100"})</pre>
	<pre>inscription({"p":"bsv-20", "op":"transfer", "tick":"ordi", "amt":"500"})</pre>
	<pre>inscription({"p":"bsv-20", "op":"transfer", "tick":"ordi", "amt":"400"})</pre>

Grandfathering in older inscriptions

There are several inscriptions on the blockchain prior to the release of this spec that use text/plain as the content type instead of application/bsv-20. We will continue to index these up to block height 793000. Starting with this block, BSV-20 inscriptions must have a content type of application/bsv-20 to be considered valid by indexers.

Implied Transfers Deprecated

V1 of this spec, supported functionality of implied transfers. An implied transfer was achieved by transferring a mint or transfer inscription without the creation of a new transfer inscription. This functionality was put in place to support users who transferred their inscriptions before there were any publicly accessible tools to create transfer inscriptions, and was always a temporary solution. This functionality will be discontinued at block height 807000.

References

- 1: [BRC20 on BTC](#)