BUIP086 – bitcoincash: URI format update

Layer: Applications
Title: BUIP086 – bitcoincash: URI format update
Author: Brendan Lee <brendanjlee@protonmail.com>
Comments-Summary: No comments yet.
Comments-URI:
Status: Draft
Type: Standards Track
Created: 2018-02-05

Contents

Abstract

This BUIP describes a revision of the bitcoin: URI scheme (BIP 20/BIP21) to enable the creation of Bitcoin transactions with multiple and complex outputs including scripts and data.

New features proposed to be added in version 2.0 include:

Creating a new URI format allowing multiple outputs to be defined in a single URI

Adding functionality to allow multiple URLs to be scanned together and condensed into a single transaction

Create mechanisms that allow scripts and smart contracts to be applied within the standard URI formatting guidelines

Motivation

This revision allows merchants to create more flexible transaction URIs that can split single payments into multiple outputs. For example, a merchant may have an agreement to pay some part of all gross revenue to a landlord, some part to a franchise organisation and some part as GST/VAT/Sales tax. This BUIP allows these additional transactions to be handled instantaneously as part of the merchant's regular commercial operations, thereby simplifying their accounting and reporting processes. It also enables the third parties to receive their payments instantaneously rather than waiting until monthly or quarterly accounting processes can be undertaken.
This revision also allows users to condense multiple URIs into a single set of outputs by scanning them consecutively before executing the transaction. For example, a merchant could attach QR codes to each item in the store directing the wallet to pay different amounts to a single common address. The new 'multiscanparam' allows a compatible SPV wallet to enter multiple URLs together to create a smart contract. Smart contracts can be used to perform token passing using coloured coins, or through OP_RETURN outputs.

The addition of the ability to create unique scripts for each output in the URI gives merchants and users a high degree of flexibility and allows new application layers to be built and executed using methods already widely used for bitcoin transactions.
Specification
The bitcoincash: URI scheme is updated to version 2.0. This new version allows multiple new features including:

Nesting of multiple transaction outputs within a single URI

Addition of optional transaction outputs

Processing of multiple URIs into a single transaction

Specification of unique redeem scripts for each output defined in the transaction

The URI scheme maintains compatibility with version 1.0 through the use of compatible labels, but re-structures the URI outputs into a set of transaction data arrays which contain information relating to each transaction.
Separate arrays exist for each transaction parameter (bitcoinaddress, amountparam, messageparam, labelparam), with the elements of each array being linked to the corresponding transaction (e.g. amount[1], label[1] and message[1] would all relate to transaction[1]).
Additional parameters are added to the scheme including:

multiscanparam – Boolean value allowing a merchant to allow payers to merge multiple URIs into a single transaction

bitcoinscript[ ] – An array containing script data for each output

txidparam[ ] and voutparam[ ] – arrays used by scripts to identify specific UTXO that the script is trying to spend

Note:

To maintain compatibility with URI standards, all reserved characters must be transmitted as percent encoded values. E.g. %5B = [and %5D =].

For further information on reserved characters, see the following link: https://tools.ietf.org/html/rfc3986#section-2.2

For the purposes of readability, this specification uses the reserved symbols, however all examples have been represented using RFC3986 compliant notation.

Parameters

The URI and its parameters are defined as follows:

Bitcoincashurn = "bitcoincash:" bitcoinaddress [";version=" bitcoinversion] [“?” bitcoinparam] ["?" bitcoinparam[]] [“?” otherparam]
bitcoinaddress = cashaddress | legacyaddress

cashaddress = cashadd *cashaddr

legacyaddress = base58 *base58
bitcoinversion = "2.0"
bitcoinparam = amountparam | labelparam | messageparam | sendparam | multiscanorderparam | otherparam
amountparam = “amount=” *amount
amount = amountdecimal | amounthex
amountdecimal = *digit ["." *digit] ["X" *digit]
amounthex = "x" *hexdigit ["." *hexdigit] ["X" *hexdigit]
labelparam = “label=” *pchar
messageparam = “message” *pchar
sendparam = "send=" *pchar
multiscanorderparam = “order[?]=” *char
otherparam = pchar *pchar "=" *pchar
bitcoinparam[] = bitcoinaddress[] | amountparam[] | labelparam[] | messageparam[] | bitcoinscrip[]
bitcoinaddress[] = “address[?]=” cashaddress | legacyaddress

cashaddress = cashaddr *cashaddr

legacyaddress = base58 *base58
bitcoinaddress[0] = bitcoinaddresss
amountparam[] = "amount[?]=" amount
amountparam[?] = amountdecimal | amounthex
amountdecimal = *digit ["." *digit] ["X" *digit]
amounthex = "x" *hexdigit ["." *hexdigit] ["X" *hexdigit]
amountparam[0] = amountparam
labelparam[] = "label[?]=" *pchar
labelparam[0] = labelparam
messageparam[] = "message[?]=" *pchar
messageparam[0] = messageparam

txscript[] = “txscript[?]=’ *pchar
bitcoinscript[] = "script[?]=" *pchar

bitcoinscript[0] = bitcoinscript
bitcoinscript[] = "script[?]=" *pchar

bitcoinscript[0] = bitcoinscript

txidparam[] = “txid[?]=” *base64

voutparam[] = “vout[?]=” *digit

Compressed Labels
Due to the size constraints of some URL presentation methods the following compressed labels will also be considered valid for transaction construction:

“v” = “version”

“ad” = “address”
“am” = “amount”
“l” = “label”
“m” = “message”
“o” = “order”
“ts” = “txscript”

“bs” = “bitcoinscript”

“tx” = “txid”

“vo” = “vout”

Compatibility
The new URI format maintains compatibility with the current version 1.0 format through the inclusion of the bitcoinparam labels used in version1.0. This means that POS systems that do not need features associated with version2.0 URLs do not need to upgrade to be compatible with wallets that support version2.0

Capabilities
The version 2.0 format allows merchants to create structured URIs that can send multiple complex outputs where each output can contain a non-default redeem script and be sent to a unique destination. This is achieved by using the bitcoinparam[ ] array structures.

The minimum amount of information needed by a wallet to add an output to a transaction is a bitcoin address. For each output being requested, the URI must add a bitcoin address to a position in the bitcoinaddress[ ] array. The position of the address in the bitcoinaddress[ ] array determines the position of the ancillary data in the amountparam[ ], labelparam[ ], messageparam[ ], scriptparam[ ] and txid[ ] arrays. E.g. amount data for the output to the address located in bitcoinaddress[1], is entered into the amountparam[1], messageparam[1] and labelparam[1] array positions.

A URI that defines more than one element in any bitcoinparam[N] field would be rejected as invalid. E.g. if a URI defines an amount in the amount field and then defines a conflicting amount in the amount[0] array position, the URI would be rejected as invalid. Similarly, if there are two separate values attached to the amount[2] label, the URI would be rejected as invalid.

For tokens and sidechains, a script can identify a particular UTXO which is to be spent using the txidparam[ ] and voutparam[ ] arrays. This can be used to specify a UTXO which is known to contain a coloured coins or a certain script. For the transaction to be valid, the selected UTXO

Requirements for URL reader
The URI provides a medium through which complex transaction structures can be defined between two parties. For a transaction to be completed, the seller must meet all the

requirements stipulated in the URL.

Where amount[N] is defined, the wallet can assume that the requested value is agreed upon between sender and receiver. It is up to the user's wallet to determine how best to structure that payment using the funds has access to.

Where script[N] is the wallet can assume that any outputs sent to that address that do not contain the correct redeem script will not meet the requirements of the contract and will not complete the transaction. It is up to the sending wallet to ensure that amounts and scripting requirements imposed by the receiver are met.

Multiscan contracts

The intent of the Multiscan function is to allow for the creation of a new type of smart contract. A customer assembles a contract by using their SPV wallet to scans URLs that include a contract order.

A multiscan contract is started when a wallet receives a URL with a non-zero value in the multiscanorderparam field. The wallet sees this and goes into multiscan mode.

In multiscan mode, the wallet should allow the user to enter multiple different URLs. URLs do not need to be scanned in order however the wallet user will not be able to terminate the contract without a URL labelled order[FF]. The order[FF] URL may contain financial transactions and parameters. If OP_GROUP is used in this Script[0], any URLs that are part of the contract will be able to use coloured coins as well.

When the wallet assembles the transaction, the transactions in address[X>0] of the order[FF] URL, ascending transaction numbers from zero will be set up to be the first when the contract is executed. Scripts in the remaining URLs will be appended in ascending order. Once all the URLs have been concatenated, the transactions with address[X<0] as a parameter in the order[FF] URL are appended in ascending order.

Any URLs that have the same multiscanorderparam must be concatenated in no particular order. If a zero value transaction

Where there are multiple transactions directed to a particular address in the contract, from multiple URLs with the same multiscanorderparam, those amounts can be aggregated.

When additional URLs are scanned, the wallet will expect multiscanparam to be non-zero. If a URL without a non-zero multiscanparam is scanned, it is either rejected from the contract, or the contract is terminated.

Example: Alice visits Bobmart to buy a few things.

When Alice arrives, she can choose three baskets. Blue, green and red. Each has a NFC chip in the handle containing an order[FF] URL.

Alice picks the blue basket and scans the handle with her phone. Its zero script sets up the transaction to support colored coins using OP_GROUP. Her basket will now reward her with 1 Bobpoint for every bit she spends. Her phone now goes into scan mode.

The green basket's zero script sets up the transaction to use the Bob's charities colored coin system. Any transactions would carry discounts that apply to Bobmart Rewards members, but instead of receiving Bob's her BobPoints, Bobmart gives 1 satoshi to a local charity.

The red basket has no points or charity, and is a basic payment script.

Alice buys some fruit, (order[12]), milk(order[2]), cheese (order[2]) and bread (order[10]), scanning the code on each as she puts them in her basket. walks to the packing area and hits ‘Pay Now’.

Her wallet sets up a contract that applies bob’s rewards to be automatically applied to her wallet. The transaction then pays 500 bits to the dairy who made the milk and cheese, 250 to the Bob’s dairy transport and storage, 1200 bits to various orchards for the fruit and 1000 bits to bob’s fruit department and 500 bits to Bob’s bakery. The wallet can send this onto the network and Bob and all of his suppliers are paid at once.

Bitcoinscript
The version2.0 URI format allows URLs to be created which fully or partially define complex smart contracts. By using the bitcoinscript[] parameters, a URL can be constructed that defines smart contract scripts to be applied to some or all of the outputs in the transaction.

This can be used to apply non-default redeem scrips such as timelocks, tokenisation scripts or evaluation of multiple signatures to outputs containing funds, or it can be used to define the contents of an OP_RETURN output that needs to be included in the transaction.

It is up to the sending wallet to ensure that scripting requirements specified in the URL are met.

Appendix
Version 1.0 Syntax

The version 2.0 implementation is still fully compatible with version 1.0 wallets. POS providers should ensure to always specify version=2.0 in situation where version 2.0 URLs are being presented. Otherwise a version 1.0 compatible wallet will look only at the parameters in the bitcoinparam fields and drop all other data.