```
BIP: 21
Layer: Applications
Title: URI Scheme
Author: Nils Schneider <nils.schneider@gmail.com>
        Matt Corallo <bip21@bluematt.me>
Comments-Summary: No comments yet.
Comments-URI: https://github.com/bitcoin/bips/wiki/Comments:BIP-0021
Status: Final
Type: Standards Track
Created: 2012-01-29
```

This BIP is a modification of an earlier BIP 0020 by Luke Dashjr. BIP 0020 was based off an earlier document by Nils Schneider. The alternative payment amounts in BIP 0020 have been removed.

Abstract

This BIP proposes a URI scheme for making Bitcoin payments.

Motivation

The purpose of this URI scheme is to enable users to easily make payments by simply clicking links on webpages or scanning QR Codes.

Specification

General rules for handling (important!)

Bitcoin clients MUST NOT act on URIs without getting the user's authorization. They SHOULD require the user to manually approve each payment individually, though in some cases they MAY allow the user to automatically make this decision.

Operating system integration

Graphical bitcoin clients SHOULD register themselves as the handler for the "bitcoin:" URI scheme by default, if no other handler is already registered. If there is already a registered handler, they MAY prompt the user to change it once when they first run the client.

General Format

Bitcoin URIs follow the general format for URIs as set forth in RFC 3986. The path component consists of a bitcoin address, and the query component provides additional payment options.

Elements of the query component may contain characters outside the valid range. These must first be encoded according to UTF-8, and then each octet of the

corresponding UTF-8 sequence must be percent-encoded as described in RFC 3986.

ABNF grammar

(See also a simpler representation of syntax)

```
bitcoinurn     = "bitcoin:" bitcoinaddress [ "?" bitcoinparams ]
bitcoinaddress = *base58
bitcoinparams  = bitcoinparam [ "&" bitcoinparams ]
bitcoinparam   = [ amountparam / labelparam / messageparam / otherparam / reqparam ]
amountparam    = "amount=" *digit [ "." *digit ]
labelparam     = "label=" *qchar
messageparam   = "message=" *qchar
otherparam     = qchar *qchar [ "=" *qchar ]
reqparam       = "req-" qchar *qchar [ "=" *qchar ]
```

Here, "qchar" corresponds to valid characters of an RFC 3986 URI query component, excluding the "=" and "&" characters, which this BIP takes as separators.

The scheme component ("bitcoin:") is case-insensitive, and implementations must accept any combination of uppercase and lowercase letters. The rest of the URI is case-sensitive, including the query parameter keys.

Query Keys

- label: Label for that address (e.g. name of receiver)
- address: bitcoin address
- message: message that describes the transaction to the user (see examples below)
- (others): optional, for future extensions

Transfer amount If an amount is provided, it MUST be specified in decimal BTC. All amounts MUST contain no commas and use a period (.) as the separating character to separate whole numbers and decimal fractions. I.e. amount=50.00 or amount=50 is treated as 50 BTC, and amount=50,000.00 is invalid.

Bitcoin clients MAY display the amount in any format that is not intended to deceive the user. They SHOULD choose a format that is foremost least confusing, and only after that most reasonable given the amount requested. For example, so long as the majority of users work in BTC units, values should always be displayed in BTC by default, even if mBTC or TBC would otherwise be a more logical interpretation of the amount.

Rationale

Payment identifiers, not person identifiers

Current best practices are that a unique address should be used for every transaction. Therefore, a URI scheme should not represent an exchange of personal information, but a one-time payment.

Accessibility (URI scheme name)

Should someone from the outside happen to see such a URI, the URI scheme name already gives a description. A quick search should then do the rest to help them find the resources needed to make their payment. Other proposed names sound much more cryptic; the chance that someone googles that out of curiosity are much slimmer. Also, very likely, what he will find are mostly technical specifications - not the best introduction to bitcoin.

Forward compatibility

Variables which are prefixed with a req- are considered required. If a client does not implement any variables which are prefixed with req-, it MUST consider the entire URI invalid. Any other variables which are not implemented, but which are not prefixed with a req-, can be safely ignored.

Backward compatibility

As this BIP is written, several clients already implement a bitcoin: URI scheme similar to this one, however usually without the additional "req-" prefix requirement. Thus, it is recommended that additional variables prefixed with req- not be used in a mission-critical way until a grace period of 6 months from the finalization of this BIP has passed in order to allow client developers to release new versions, and users of old clients to upgrade.

Appendix

Simpler syntax

This section is non-normative and does not cover all possible syntax. Please see the BNF grammar above for the normative syntax.

[foo] means optional, <bar> are placeholders

```
bitcoin:<address>[?amount=<amount>][?label=<label>][?message=<message>]
```

Examples

Just the address:

```
bitcoin:175tWpb8K1S7NmH4Zx6rewF9WQrcZv245W
```

Address with name:

`bitcoin:175tWpb8K1S7NmH4Zx6rewF9WQrcZv245W?label=Luke-Jr`

Request 20.30 BTC to "Luke-Jr":

`bitcoin:175tWpb8K1S7NmH4Zx6rewF9WQrcZv245W?amount=20.3&label=Luke-Jr`

Request 50 BTC with message:

`bitcoin:175tWpb8K1S7NmH4Zx6rewF9WQrcZv245W?amount=50&label=Luke-Jr&message=Donation%20for%20`

Some future version that has variables which are (currently) not understood and required and thus invalid:

`bitcoin:175tWpb8K1S7NmH4Zx6rewF9WQrcZv245W?req-somethingyoudontunderstand=50&req-somethingel`

Some future version that has variables which are (currently) not understood but not required and thus valid:

`bitcoin:175tWpb8K1S7NmH4Zx6rewF9WQrcZv245W?somethingyoudontunderstand=50&somethingelseyoudon`

Characters must be URI encoded properly.

Reference Implementations

Bitcoin clients

- Bitcoin-Qt supports the old version of Bitcoin URIs (ie without the req- prefix), with Windows and KDE integration as of commit 70f55355e29c8e45b607e782c5d76609d23cc858.

Libraries

- Javascript - https://github.com/bitcoinjs/bip21
- Java - https://github.com/SandroMachado/BitcoinPaymentURI
- Swift - https://github.com/SandroMachado/BitcoinPaymentURISwift