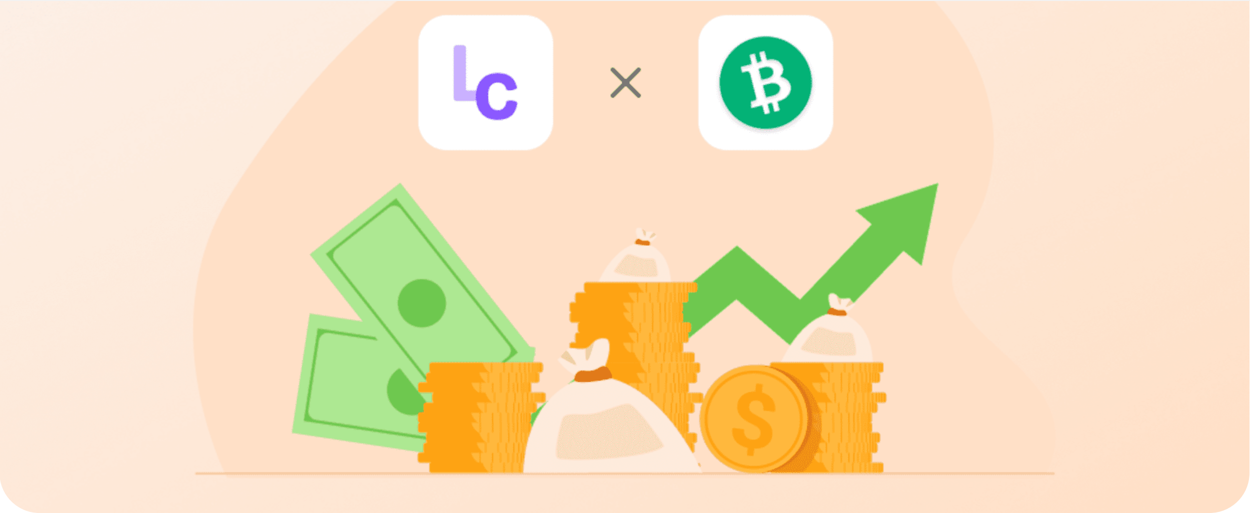
09 September 2021

LocalCryptos launches Bitcoin Cash trading as its 5th crypto

Users can now buy and sell Bitcoin Cash (BCH) on the most popular non-custodial peer-to-peer crypto marketplace.



Almost a year has passed since we added a new cryptocurrency to LocalCryptos, and many of you have been requesting Bitcoin Cash as the next flavor of our non-custodial formula. The wait is finally over.

Beginning today, **peer-to-peer Bitcoin Cash trading is available on LocalCryptos**.

How will the non-custodial escrow

Bitcoin, Litecoin, and Dash, **LocalCryptos leverages the “OP_CHECKDATASIG” op-code** which is only available in Bitcoin Cash transactions.

A regular user won’t notice a difference between this new escrow type, as the outcome is virtually identical to LocalCryptos’ non-custodial BTC escrow script. However from a programmer’s perspective, the non-custodial Bitcoin Cash P2SH script is more simplistic and intuitive, and it comes with some advantages.

As with all escrows on LocalCryptos, it is technically impossible for us to spend BCH in escrow. We only become involved when there is a payment dispute, and once involved we only have the ability to allow the BCH to be redeemed by the buyer or seller. Click here to learn more about the cryptographic magic that makes this possible.

Of course, your Bitcoin Cash wallet on LocalCryptos is also **self-custodial**. Your keys—your coins. Developers can view our annotated Bitcoin Cash Script templates at the bottom of this announcement.

- It was created out of the demand for a variant of Bitcoin to **process a higher number of transactions** per block which, according to supporters, would require increasing the blockchain's block size.
- As a result of this approach to the scalability issue, Bitcoin Cash **doesn't support Segregated Witness** transactions (unlike Bitcoin and Litecoin).
- Despite the differences, both coins still share many similarities — they both mine coins under the Proof-of-Work consensus algorithm, their total coin supply is capped at 21 million, and both support **pay-to-public-key-hash** (P2PKH) and **pay-to-script-hash** (P2SH) addresses.
- The average Bitcoin Cash transaction fee is usually less than 1¢ — **practically free**!

How Bitcoin Cash non-custodial escrow works

transaction, except that the parties involved don't need to agree upon how and when the Bitcoin Cash outputs are spent. The oracle — which can be the seller, buyer, or arbitrator, depending on the circumstances of the trade — doesn't have the ability to place conditions on the transaction, unlike with traditional multi-signature wallets.

This is due to the fact that with a traditional multi-signature wallet, all parties must sign a full transaction including all outputs and inputs, whereas with a non-custodial escrow transaction that uses `OP_CHECKDATASIG`, the oracle simply needs to give the winner a signature which they can use at any time to unlock the BCH in any way they choose.

This type of on-chain escrow mechanism gives the buyer and seller the ability to exchange without permission, and the arbitrator the ability to intervene as a non-custodial mediator in the case of a payment dispute.

Seller's deposit into escrow

To move Bitcoin Cash into escrow, the seller generates a transaction with two outputs. One output is the escrow to the buyer and the other is a output. This doesn’t require our intervention. Similarly, if the buyer chooses to cancel the trade on their own accord, the seller can spend the output without our help. The first scenario is a “release” and the second is a “return”.

In the event of a payment dispute, an arbitrator can step in and act as a mediator. The arbitrator can only allow either the seller or the buyer to spend, by design. The fee script allows the arbitrator to collect a fee after a released escrow, or for seller to claim a refund if the trade is unsuccessful.

Escrow output template

```
OP_DUP # We need to use the byte again afterwards
# Get the hashed public keys we need to compare against (ours, and the oracle)
OP_1
OP_EQUAL
OP_IF
  <hash160(SellerPubKey)> # Oracle pub key
  <hash160(BuyerPubKey)> # Spender pub key
OP_ELSE
  OP_DUP
  OP_ELSE
    OP_DUP
    OP_3 # = return from buyer
    OP_EQUAL
    OP_IF
      <hash160(BuyerPubKey)> # Oracle pub key
      <hash160(SellerPubKey)> # Spender pub key
    OP_ELSE
      OP_DUP
      OP_4 # = return from arbitrator
      OP_EQUALVERIFY # must be true, else the message is unknown
      <hash160(ArbPubKey)> # Oracle pub key
      <hash160(SellerPubKey)> # Spender pub key
    OP_ENDIF
  OP_ENDIF
OP_ENDIF
# Put the hashed public keys on the alt stack
OP_TOALTSTACK
OP_TOALTSTACK # Stack is effectively reset to the input
# On the alt stack we have: [ hash160(SpenderPubKey), hash160(OraclePubKey) ]
<EscrowKey> # Append the nonce to the escrow key to make the message
OP_CAT # Stack is [ ..., <OraclePubKey>, <0x01 || EscrowKey> ]
OP_SWAP # Use this later; verify the oracle public key hash first
```

```
OP_DUP
OP_HASH160
OP_FROMALTSTACK
OP_EQUALVERIFY
OP_CHECKSIG
```

Fee output template

This is the fee portion of the trade. If the trade is unsuccessful, the fee can be reclaimed by the seller; if the trade is successful, the fee output will be claimed by LocalCryptos.

```
OP_DEPTH # Count stack size
OP_2
OP_EQUAL # Does the input stack only have two items?
OP_IF # If yes, this is the owner collecting fee; simple PKH
  OP_DUP
  OP_HASH160
  <hash160(ArbPubKey)>
  OP_EQUALVERIFY
  OP_CHECKSIG
OP_ELSE # Seller is spending a "returned" (i.e. canceled) escrow
```

```
OP_ELSE
  OP_DUP
  OP_4 # = return from arbitrator
  OP_EQUALVERIFY # must be true, else the message is unknown
  <hash160(ArbPubKey)> # Oracle pub key
OP_ENDIF
<hash160(SellerPubKey)> # Spender pub key
# Put the hashed public keys on the alt stack
OP_TOALTSTACK
OP_TOALTSTACK # Stack is effectively reset to the input
# On the alt stack we have: [ hash160(SpenderPubKey), hash160(OraclePubKey) ]
<EscrowKey> # Append the nonce to the escrow key to make the message
OP_CAT # Stack is [ ..., <OraclePubKey>, <0x01 || EscrowKey> ]
OP_SWAP # Use this later; verify the oracle public key hash first
OP_DUP
OP_HASH160
OP_FROMALTSTACK # Grab hashed pub key from alt stack
OP_EQUALVERIFY # Public key checks out; now verify the oracle signature
OP_CHECKDATASIGVERIFY # Verify the sender
OP_DUP
OP_HASH160
OP_FROMALTSTACK
OP_EQUALVERIFY
```

To spend an escrow output, the spender must provide in their Bitcoin Cash transaction's `scriptSig`:

```
<Sig> <SpenderPubKey> <OracleSignature> <OraclePubKey> <ActionByte>
# Example: <Sig> <OwnPubkey> <SignatureFromSeller> <SellerPubKey> OP_1
```

1. `<ActionByte>` is a byte corresponding with the situation being executed.

- 1: Escrow is being released by the seller
- 2: Escrow is being released by the arbitrator
- 3: Escrow is being returned by the buyer
- 4: Escrow is being returned by the arbitrator

2. `<OraclePubKey>` is the public key of the person signing the release/return message.

- 1: `<OraclePubKey>` = `<SellerPubKey>`

3. `<OracleSignature>` is a signature from the oracle of `ECDSA(<ActionByte> || <EscrowKey>)`. The `<EscrowKey>` is unique so that signatures cannot be re-used across escrows.

4. `<SpenderPubKey>` is the buyer's public key if a release, otherwise the seller's public key.

5. `<Sig>` is the transaction signature from the spender.

You might be interested in

More articles in this category

See all articles

After 5 years, LocalCryptos is saying goodbye

21 October 2022

Updates

Zero-conf escrow enabled for Bitcoin Cash on LocalCryptos

Use LocalCryptos