

Claims:

1. A computer implemented method for implementing an event stream associated with a blockchain, the method implemented by a platform processor associated with an application programming interface (API), the method comprising the steps of:

receiving a request from a client relating to an event stream ES, the request received based on a Hypertext Transfer Protocol (HTTP) transmission protocol format;

determine a current length n of the event stream E;

wherein if $n = 0$, then the method comprises:

identifying an event E_n for the event stream in the received request as a first event to create the event stream ES;

processing the received event E_n by creating a blockchain transaction including a first unspent output $UTXO_{0_dust}$ that is a dust output;

wherein if $0 < n < N$, where N is the final or maximum value for, then the method comprises:

identifying an event E_n for the event stream in the received request a current event to amend the event stream ES;

processing the received event E_n by creating a blockchain transaction including:

a first input spending a dust output associated with a previous transaction for the event stream ES;

a first unspent transaction output $UTXO_{n_dust}$ being a dust output; and

a final unspent transaction output $UTXO_{n_data}$ associated with event data representing the current event E_n ;

wherein if $n = N$, then the method comprises:

identifying an event E_n for the event stream in the received request as a final event to create the terminate the stream ES; and

processing the received event E_n by creating a blockchain transaction including:

a first input spending a dust output associated with a previous transaction for the event stream ES; and

a final unspent output $UTXO_N$ that is associated with a digital asset that more than a defined dust output value;

the method comprising:

submitting the created transaction to the blockchain; and

sending a result associated with the event stream ES in TX_0 to the client, the result being provided based on the HTTP transmission protocol format.

2. A computer implemented method for creating an event stream associated with a blockchain, the method implemented by a platform processor associated with an application programming interface (API), the method comprising the steps of:

receiving a request from a client relating to a new event stream ES, the request received based on a Hypertext Transfer Protocol (HTTP) transmission protocol format;

determining a hierarchical deterministic key chain K to be used with the new event stream ES, wherein the key chain K includes a series of private/public key pairs derived from a selected parent key pair, such that $K = K_{n=0 \text{ to } N}$, whereby n is an integer from 0 to N, each integer n representing a current length or current number of events associated with event stream ES, and N represents a maximum or final value of n;

creating a first blockchain transaction TX_0 for the new event stream ES where $n = 0$, the blockchain transaction including:

a first unspent transaction output ($UTXO_{0_dust}$) being a dust output, said dust output associated with a locking script secured with a first derived key pair K_0 from the key chain K;

submitting the transaction TX_0 to the blockchain; and

sending a result associated with the created event stream ES in TX_0 to the client, the result being provided based on the HTTP transmission protocol format.

3. A computer implemented method for updating an event stream associated with a blockchain, the method implemented by a platform processor associated with an application programming interface (API), the method comprising the steps of:

receiving a request from a client to update an existing event stream ES associated with the blockchain, the request received based on a Hypertext Transfer Protocol (HTTP) transmission protocol format, and wherein the event stream ES is associated with a key chain K such that $K = K_{n=0 \text{ to } N}$, where n is an integer from 0 to N, each integer n representing a current length or current number of events associated with the event stream ES, where N represents a maximum or final value of n;

determining a current length n of the event stream ES;

from the request, obtaining a current event E_n to be appended to the event stream ES;

identifying a previous blockchain transaction TX_{n-1} associated with the event stream ES;

obtaining a key pair K_{n-1} associated with the identified previous transaction TX_{n-1} ;

deriving a key pair K_n for the current event E_n ;

creating a current blockchain transaction TX_n for the current event E_n to be appended to the event stream ES where $0 < n < N$, the current blockchain transaction including:

a first input spending a dust output associated with the previous transaction TX_{n-1} , said spend authorised with the obtained key pair K_{n-1} for the previous transaction;
 a first unspent transaction output ($UTXO_{n_dust}$) being a dust output for the current transaction TX_n , said dust output associated with a locking script secured with the derived key pair K_n ; and
 a final unspent transaction output ($UTXO_{n_data}$) associated with event data representing the current event E_n ;
 submitting the transaction TX_n to the blockchain; and
 sending a result associated with the updated event stream ES to the client, the result being provided based on the HTTP transmission protocol format.

4. The method as claimed in claim 3 where the event data for the event E_n in the final unspent transaction output ($UTXO_{n_data}$) for the current blockchain transaction TX_n includes a hash of said event data.

5. The method as claimed in claim 4 where the hash of said event data is applied by the platform processor.

6. The method as claimed in claim 4 where the hash of said event data is applied by the client device prior to being included in the request that is received by the platform processor.

7. The method as claimed in claim 3 where the event data for the event E_n in the final unspent transaction output ($UTXO_{n_data}$) for the current blockchain transaction TX_n includes raw event data.

8. A computer implemented method for terminating an event stream associated with a blockchain, the method implemented by a platform processor associated with an application programming interface (API), the method comprising the steps of:

receiving a request from a client to terminate an existing event stream ES associated with the blockchain, the request received based on a Hypertext Transfer Protocol (HTTP) transmission protocol format, and wherein the event stream ES is associated a key chain K such that $K = K_{n=0 \text{ to } N}$, where n is an integer from 0 to N, each integer n representing a current length or current number of events associated with the event stream ES, where N represents a maximum or final value of n;

determining a current length $n = N$ of the event stream ES;

from the request, obtaining a final event E_N to be appended to the event stream ES;

identifying a previous blockchain transaction TX_{N-1} associated with the event stream ES;

obtaining a key pair K_{N-1} associated with the previous transaction TX_{N-1} ;

deriving a key pair K_N for the final event E_N ;

creating a final blockchain transaction TX_N for the final event E_N to be appended to the event stream ES, the final blockchain transaction including:

a first input spending a dust output associated with the previous transaction TX_{N-1} , said spend authorised with the obtained key pair K_{N-1} for the previous transaction;

a first unspent transaction output ($UTXO_N$) associated with a digital asset that is above a defined dust output limit;

submitting the transaction TX_N to the blockchain; and

sending a result associated with the terminated event stream ES to the client, the result being provided based on the HTTP transmission protocol format.

9. The method as claimed in any one of claims 1 to 8 wherein the result associated with the event stream ES includes a certificate confirming at least one of the following :

- a transaction identifier within which the event E_n was submitted to the blockchain;
- a Merkle inclusion proof of the transaction to a header in the blockchain
- a copy of the block header in which said transaction was included

10. The method as claimed in any one of claims 1 to 9 wherein the step of submitting comprises including the created transaction in a subsequent block associated with the blockchain to be mined.

11. The method as claimed in any one of claims 1 to 10 wherein the created blockchain transaction further comprises

an input associated with a digital asset; and

one or more change outputs associated with the digital asset.

12. The method as claimed in claim 11 wherein the digital asset is associated with an operational float.

13. The method as claimed in any one of claims 1 to 12 comprising the step of identifying the event stream ES based on a transaction identifier associated with the submitted blockchain transaction.

14. The method as claimed in any one of claims 1 to 13 comprising the step of identifying a state associated with the event stream ES based on a transaction identifier associated with the submitted blockchain transaction.
15. The method as claimed in any one of claims 1 to 14 wherein the event stream is used to represent a smart contract on the blockchain.
16. The method as claimed in claim 15 wherein the smart contract is implemented as a Finite State Machine(FSM).
17. The method as claimed in any one of claims 1 to 16 comprising storing a copy or a record or a log that is based on the result(s) for each event of the event stream in an off-chain storage resource.
18. A computer implemented method for writing data associated with an event stream in a blockchain, the method implemented by one or more processors of a given client among a plurality of clients, the method comprising the steps of:
obtaining or identifying an application programming interface (API) endpoint associated with one or more processors associated with a platform for providing a data writing service;
sending a request for one or more events E_n pertaining to an event stream, wherein the request is sent based on a Hypertext Transfer Protocol (HTTP) transmission protocol format;
and
receiving a result associated with an output script of a blockchain transaction pertaining to the requested event E_n , said result being received based on the HTTP transmission protocol format.
19. The method as claimed in claim 18 comprising applying a hash function to event data for the event E_n , such that the request includes the hashed event data for the event E_n .
20. A computing device comprising a processor and memory, the memory including executable instructions that, as a result of execution by the processor, causes the device to perform the computer-implemented method as claimed in any one of claims 1 to 17, the computing device pertaining to a platform processor.

21. A computing device comprising a processor and memory, the memory including executable instructions that, as a result of execution by the processor, causes the device to perform the computer-implemented method as claimed in any one of claims 18 or 19, wherein the computing device is pertaining to a client.

22. A computer system comprising:

at least one platform processor communicatively coupled to at least one client via a wireless communication network, the at least one platform processor associated with an application programming interface(API) endpoint for processing HTTP requests from the at least one client, the at least one platform processor implemented in accordance with the computing device as claimed in claim 20 ; and the at least one client being implemented in accordance with the computing device as claimed in claim 21.

23. A computer-readable storage medium having stored thereon executable instructions that, as a result of being executed by a processor of a computer, cause the computer to perform the method of any one of claims 1 to 19.