

Claims

1. A computer implemented method for synchronising a plurality of event streams 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 a plurality of existing event streams ES_n $n=1$ to M associated with the blockchain, n being an integer from 1 to M , where $M \geq 2$
 - from the request, obtaining a current event E_n to be appended to each event stream ES_n among the plurality M of event streams $ES_{n=1 \text{ to } M}$;
 - for each event stream ES_n , identifying a previous blockchain transaction TX_{n-1} associated with the event stream ES_n ; and
 - creating an atomic blockchain transaction TX_n for the current event E_n to be appended to each event stream ES_n among the plurality of M event streams $ES_{n=1 \text{ to } M}$ in order to synchronise the plurality M of event streams $ES_{n=1 \text{ to } M}$, the atomic blockchain transaction TX_n comprising:
 - $n = M$ inputs, each n^{th} input spending a dust output associated with the previous transaction TX_{n-1} of the respective event stream ES_n ,
 - for each of the n inputs, a respective unspent transaction output ($UTXO_{n_dust}$) being an n^{th} dust output for the atomic transaction TX_n associated with the respective event stream ES_n , and
 - an unspent transaction output ($UTXO_{n_data}$) associated with event data representing the current event E_n .
2. The method as claimed in claim 1, wherein for each event stream ES_n among the plurality of M event streams $ES_{n=1 \text{ to } M}$, the method comprises mapping an input associated with dust to a respective output based on an arbitrary or regular pattern, to track a state of the respective event stream ES_n .
3. The method as claimed in claim 2, wherein when x is an index associated with the n^{th} input for the respective stream ES_n , the regular pattern is provided by the steps of:
 - mapping the dust of n^{th} input at index x with an output stored at index $2x$; and
 - identifying the state of the event stream ES_n based on an output stored at index $2x+1$;

wherein, the input indices x are provided from 0 to $M-1$ for the atomic blockchain transaction.

4. The method as claimed in any preceding claim wherein the API is associated with a Hypertext Transfer Protocol (HTTP) endpoint, and wherein the request received from the client is based on a HTTP transmission protocol format.
5. The method as claimed in any preceding claim wherein the endpoint is a separate and/or specific endpoint for synchronising the plurality of event streams.
6. The method as claimed in any preceding claim comprising the step of identifying the plurality of event streams $ES_{n=1 \text{ to } M}$ based on an identifier for each event stream ES_n among the plurality provided in the request from the client.
7. The method as claimed in any preceding claim comprising the step of obtaining a target index for one or more event stream ES_n among the plurality of event streams $ES_{n=1 \text{ to } M}$ for appending the current event E_n .
8. The method as claimed in claim 7 wherein the target index for the one or more event streams ES_n among the plurality is provided in the request from the client.
9. The method as claimed in any one of claims 4 to 8 wherein the request is associated with a digital signature.
10. The method as claimed in any one of claims 7 or 8 wherein, based on a determination that an actual next available index value for an event stream ES_n is the same as the target index for the respective event stream ES_n , the method includes :
 appending the current event E_n to the respective event stream ES_n ;
otherwise, the method includes generating an error notification.
11. The method as claimed in any preceding claim comprising the step of locking each event stream ES_n associated with the request from the client until the current event E_n has been appended to each of the plurality of M event streams $ES_{n=1 \text{ to } M}$, or until an error message is generated for one or more of the event streams in said plurality.

12. The method as claimed in any preceding claim wherein a dust input index for each of the $n = M$ inputs of the atomic blockchain transaction is recorded in the respective event stream ES_n associated with the n^{th} input.
13. The method as claimed in claim 10 wherein a dust output index and/or an event data index for each of the $n = M$ inputs of the atomic blockchain transaction is recorded in the respective event stream ES_n associated with the n^{th} input.
14. The method as claimed in any preceding claim wherein, for each event stream ES_n among the plurality M of event streams, the method comprises creating a key pair specific to each index associated with the respective event stream ES_n .
15. The method as claimed in any preceding claim wherein, each event stream ES_n among the plurality M of event streams is associated with a key chain K such that $K = K_{0 \text{ to } i}$, where i is an integer representing a current index value or length or current number of events associated with the given event stream ES_n , the method comprising the steps of :
- obtaining a key pair K_{i-1} associated with the identified previous transaction TX_{n-1} for the given event stream ES_n ;
 - deriving a key pair K_i for the current event E_n for the given event stream ES_n ;
 - wherein, the spend of the n^{th} input for the atomic blockchain transaction TX_n is authorised with the obtained key pair K_{i-1} for the previous transaction of the respective event stream ES_n ; and
 - wherein each n^{th} dust output is associated with a locking script secured with the derived key pair K_i for the respective event stream ES_n .
16. The method as claimed in any one of the preceding wherein the atomic blockchain transaction further comprises:
- an input associated with a digital asset; and
 - one or more change outputs associated with the digital asset.
17. The method as claimed in claim 15 wherein the digital asset is associated with an operational float.
18. The method as claimed in any preceding claim, comprising the step of:
- responsive to appending the current event E_n to each of the plurality of M event streams $ES_{n=1 \text{ to } M}$, obtaining a new current index value for each of the event streams ES_n ;

providing a response array of the obtained new index values for the plurality M of event streams.

19. The method as claimed in any preceding claim comprising :
submitting the atomic blockchain transaction TX_n to the blockchain; and
sending a result associated with each of the plurality of synchronised event streams $ES_{n=1 \text{ to } M}$ to the client.
20. The method as claimed in claim 19 wherein the step of submitting comprises including the atomic transaction in a subsequent block associated with the blockchain to be mined.
21. The method as claimed in any one of claims 19 or 20, wherein the result associated with each of the plurality of event stream $ES_{n=1 \text{ to } M}$ includes a certificate confirming at least one of the following :
- the atomic blockchain transaction identifier within which the event E_n was submitted to the blockchain;
- a Merkle inclusion proof of the atomic blockchain transaction to a header in the blockchain
- a copy of the block header in which said atomic blockchain transaction was included.
22. The method as claimed in any one of claims 19 to 21 wherein the result further includes the response array of claim 17.
23. The method as claimed in any one of claims 19 to 22 wherein the result is sent to the client based on a Hypertext Transfer Protocol (HTTP) transmission protocol format.
24. The method as claimed in any one of the preceding claims wherein the event data associated with the current E_n for synchronising the plurality of event streams $ES_{n=1 \text{ to } M}$ is the same for all event streams in the plurality.
25. The method as claimed in any one of claims 1 to 23 wherein the event data associated with the current E_n for synchronising the plurality of event streams $ES_{n=1 \text{ to } M}$ is different for at one or more of the event streams in the plurality of event streams .

26. The method as claimed in any one of the preceding claims wherein the event data associated with the current E_n for synchronising the plurality of event streams $ES_{n=1 \text{ to } M}$ comprises metadata.

27. The method as claimed in any one of claims 1 to 23 wherein the event data associated with the current E_n for synchronising the plurality of event streams $ES_{n=1 \text{ to } M}$ contains only metadata.

28. A computer implemented method for requesting synchronisation of a plurality of M event streams in a blockchain, where $M \geq 2$, 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 for synchronising event streams;

sending a request for appending an event E_n to the plurality of 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 an atomic blockchain transaction pertaining to the requested event E_n , said result being received based on the HTTP transmission protocol format.

29. The method as claimed in claim 28 wherein identifiers for each of the plurality of M event streams are included in the request.

30. the method as claimed in claim 28 or 29 comprising including a target index value for each of the plurality of M event streams in the request.

31. 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 27, the computing device pertaining to a platform processor.

32. 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 28 to 30, wherein the computing device is pertaining to a client.

33. 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 31 ; and the at least one client being implemented in accordance with the computing device as claimed in claim 32.

34. 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 27.