
<< Java Class >>
ClientDriver
org.bitcoinj.channels.htlc.client
- PARAMS: NetworkParameters
- SERVER_PORT: int
- appKit: WalletAppKit
- client: ClientChannelConnection
+ run(): void
+ main(String []): void
- client
0...1
<< Java Class >>
ClientChannelConnection
org.bitcoinj.channels.htlc.client
- wireParser: ProtobufParser<TwoWayChannelMessage>
- channelClient: ClientPaymentChannel
- channelOpenFuture: SettableFuture<ChannelClientConnection>
+ ClientChannelConnection(InetSocketAddress, int, Wallet, TransactionBroadcastScheduler, ECKey, ECKey, Coin, long)
+ buy(String, String, Coin): ListenableFuture<PaymentReceipt>
+ settle(): void
<< Java Interface >>
IPaymentChannelClient
org.bitcoinj.channels.htlc.client
+ receiveMessage(TwoWayChannelMessage): void
+ connectionOpen(): void
+ connectionClosed(): void
+ settle(): void
+ buy(String, String, Coin): ListenableFuture<PaymentReceipt>
- channelClient
0...1
- conn
0...1
<< Java Interface >>
IClientConnection
org.bitcoinj.channels.htlc.client
+ sendToServer(TwoWayChannelMessage): void
+ acceptExpireTime(long): boolean
+ channelOpen(boolean): void
+ destroyConnection(CloseReason): void
<< Java Class >>
ClientPaymentChannel
org.bitcoinj.channels.htlc.client
- CLIENT_MAJOR_VERSION: int
- CLIENT_MINOR_VERSION: int
- SERVER_MAJOR_VERSION: int
- clientPrimaryKey: ECKey
- clientSecondaryKey: ECKey
- timeWindow: long
- blockingQueue: HTLCBlockingQueue<HTLCPayment>
- htlcRound: HTLCRound
- currentBatch: List<HTLCPayment>
- paymentAckFutureMap: Map<String, SettableFuture<PaymentReceipt>
- statsResponseFutureMap: Map<String, SettableFuture<FlowResponse>>
- paymentInfoFutureMap: Map<String, SettableFuture<List<PriceInfo>>
- broadcastScheduler: BroadcastScheduler
- wallet: Wallet
- value: Coin
- conn: IClientConnection
- state: ClientChannelStateMachine
+ ClientPaymentChannel (Wallet, TransactionBroadcastScheduler, ECKey, ECKey, Coin, long, IClientConnection)
+ receiveMessage(TwoWayChannelMessage): void
+ connectionOpen(): void
+ settle(): void
+ buy(String, String, Coin): ListenableFuture<PaymentReceipt)>
<< Java Enumeration >>
HTLCRound
org.bitcoinj.channels.htlc.client
OFF: HTLCRound
WAITING_FOR_ACK: HTLCRound
CONFIRMED: HTLCRound
SERVER: HTLCRound
- state
0...1
<< Java Class >>
ClientChannelStateMachine
org.bitcoinj.channels.htlc.client
- wallet: Wallet
- broadcastScheduler: TransactionBroadcastScheduler
- clientPrimaryKey: ECKey
- clientSecondaryKey: ECKey
- serverMultisigKey: ECKey
- multisigContract: Transaction
- refundTx: Transaction
- teardownTx: Transaction
- totalValue: Coin
- totalValueInHTLCs: Coin
- expireTime: long
- htlcSettlementExpiryTime: long
- htlcRefundExpiryTime: long
- teardownForfeitureTxMap: Map<Sha256Hash, List<Transaction>>
- walletListener: AbstractWalletEventListener
- state: ChannelState
- htlcMap: Map<String, HTLCClientStateMachine>
+ ClientChannelStateMachine (Wallet, TransactionBroadcastScheduler, ECKey, ECKey, Coin, long)
+ initiate(): void
+ provideRefundSignature(byte []): void
+ updateTeardownTxWithHTLC(String, Coin): int
+ getSignedTeardownTx(String, Coin): SignedTransaction
+ getHTLCSettlementTx(String, Sha256Hash): SignedTransaction
+ getHTLCForfeitTx(String, Sha256Hash): SignedTransaction
+ attemptSettle(String, String): boolean
+ attemptBackoff(String, Transaction, TransactionSignature): void
- state
0...1
<< Java Enumeration >>
ChannelState
org.bitcoinj.channels.htlc.client
NEW: ChannelState
INITIATED: ChannelState
WAITING_FOR_SIGNED_REFUND: ChannelState
SCHEDULE_BROADCAST: ChannelState
PROVIDE_CONTRACT: ChannelState
READY: ChannelState
EXPIRED: ChannelState
- broadcastScheduler
0...1
<< Java Class >>
TransactionBroadcastScheduler
org.bitcoinj.channels.htlc
- broadcastScheduleMap: Map<Transaction, TimerTask>
- timeoutHandler: Timer
- peerGroup: TransactionBroadcaster
+ TransactionBroadcastScheduler(TransactionBroadcaster peerGroup)
+ broadcastTransaction(Transaction): ListenableFuture<Transaction>
+ updateSchedule(Transaction, Transaction, long): void
+ scheduleTransaction(Transaction, long): void
+ removeTransaction(Transaction): void
- htlcMap
0...1
<< Java Class >>
HTLCClientStateMachine
org.bitcoinj.channels.htlc.client
- teardownTxHTLCOutput: TransactionOutput
- refundTx: Transaction
- settlementTx: Transaction
- forfeitTx: Transaction
- htlcMap: Map<String, HTLCStateMachine>
+ HTLCClientStateMachine(String, Coin, long, long)
+ addHTLCOutput(Transaction, HTLCKeys): Transaction
+ signAndStoreRefundTx(Transaction, TransactionSignature, ECKey, Sha256Hash): void
+ createSettlementTx(Sha256Hash, ECKey, ECKey): SignedTransaction
+ createForfeitTx(Sha256Hash, ECKey): SignedTransaction
+ verifySecret(String): boolean
+ getRefundTx(): Transaction
+ getSettlementTx(): Transaction
+ getForfeitTx(): Transaction
<< Java Enumeration >>
HTLCState
org.bitcoinj.channels.htlc.client
NEW: HTLCState
OUTPUT_INITIATED: HTLCState
REFUND_VERIFIED: HTLCState
SETTLE_CREATED: HTLCState
FORFEIT_CREATED: HTLCState
SETTLEABLE: HTLCState
- state
0...1
<< Java Class >>
HTLCStateMachine
org.bitcoinj.channels.htlc
- value: Coin
- settlementExpiryTime: long
- refundExpiryTime: long
- secretHash: String
+ HTLCState(Coin, long, long)
+ HTLCState(String, Coin, long, long)
+ setId(String): void
+ getId(): String
+ getValue()