Micropayments Interoperability with Blockchain and Linked Data to Improve Transaction Throughputs

Academic Seminar – 3.11.2020

Amiruddin, Azmi – TU Berlin

Agenda

1. Background (revisit)
2. Initial Situation
   - Current State
   - Transition State
   - Observations
3. Research Methodology
   - Multivocal Literature Review
   - Research Model
   - Contribution
4. System Architecture
   - Baseline Architecture
   - System Design
5. Proof of Concept
   - Channel lifecycle
   - Simple payment transfer (off-chain)
   - Simple payment transfer (on-chain)

Baseline State

Refer to "Incorporating Blockchain into RDF Store“ [20]

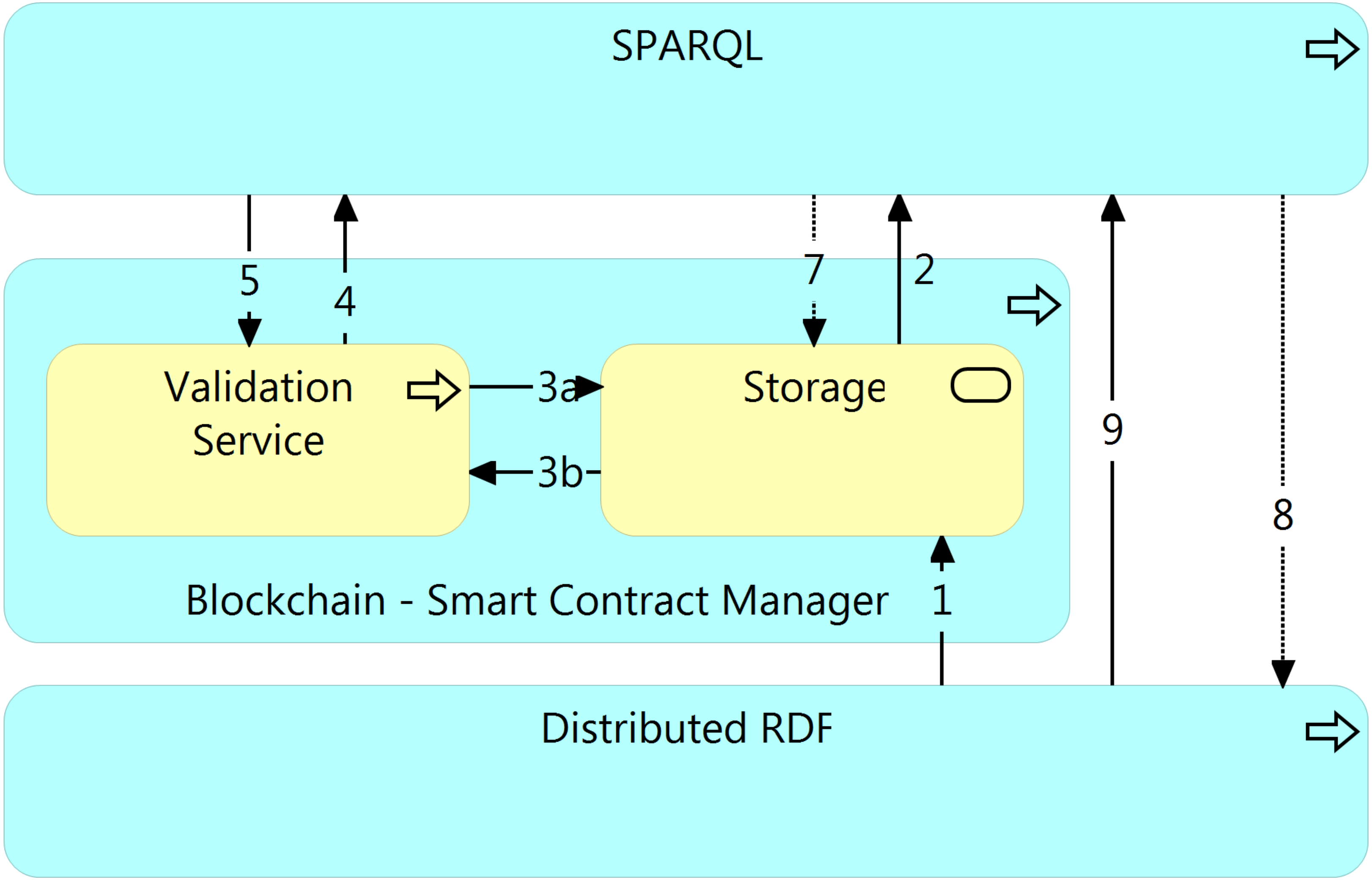
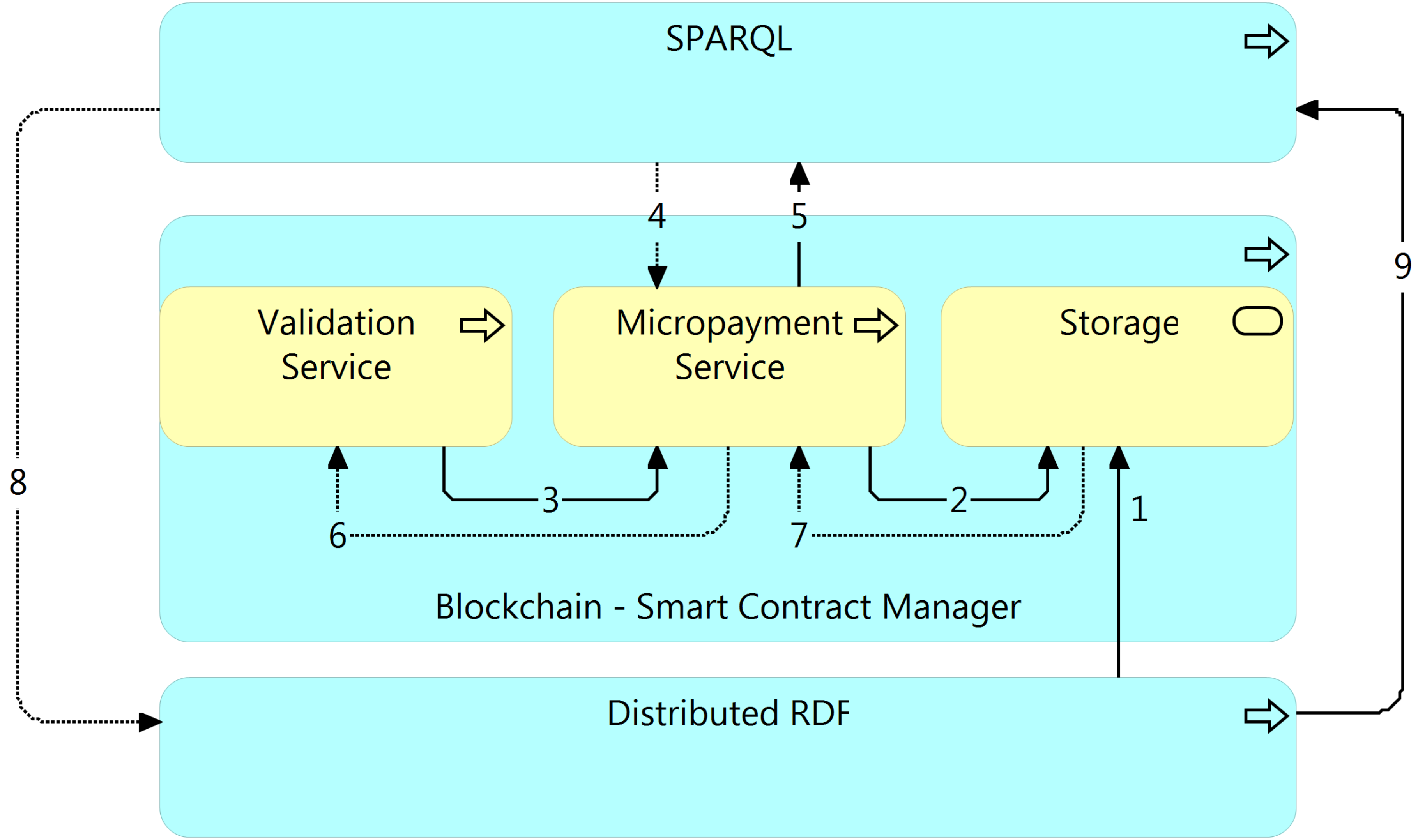


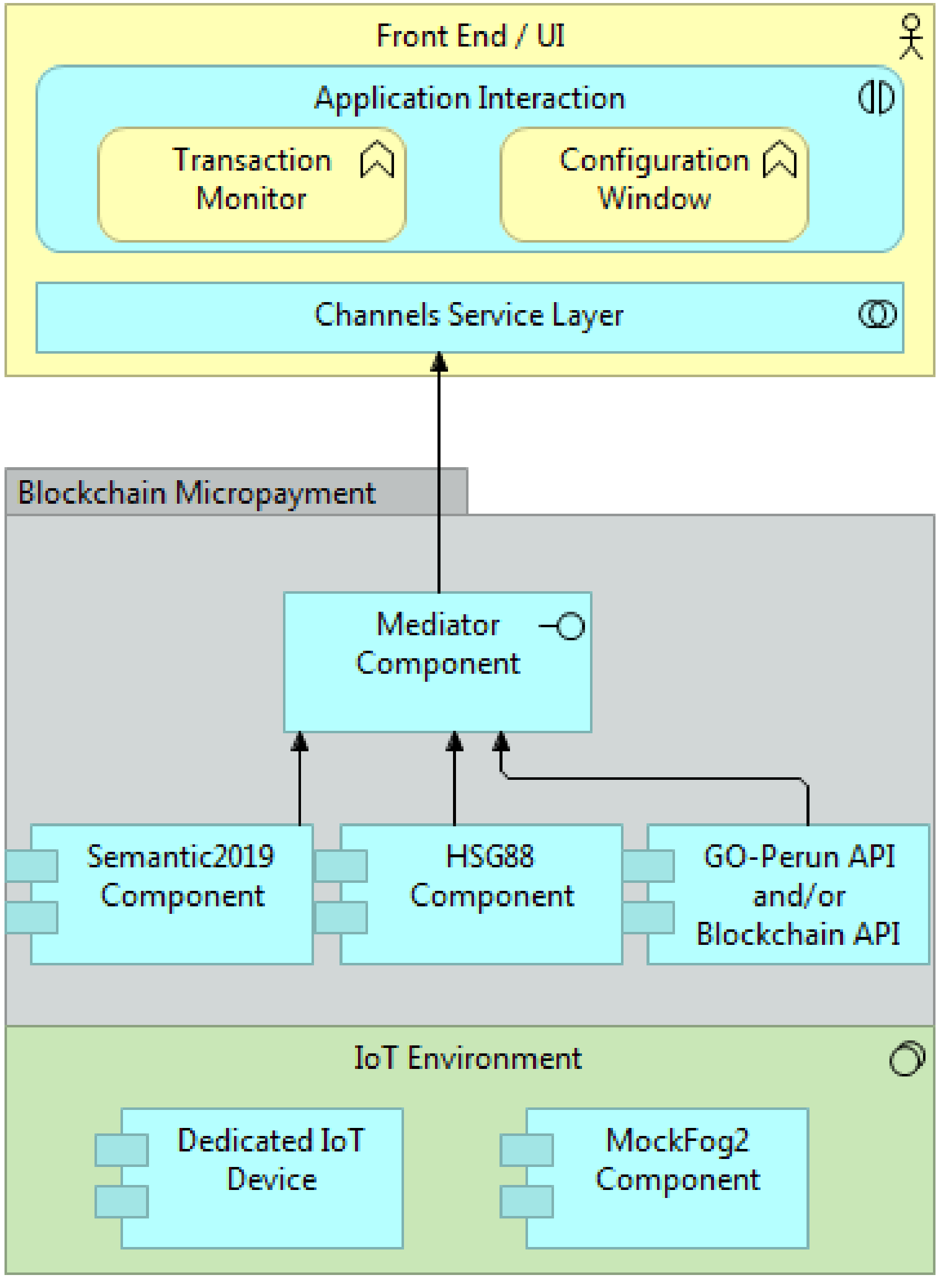
Image [27]

Transition State

Refer to "Incorporating Blockchain into RDF Store“ [27] and Enhancement HSG88 [17].



System Architecture



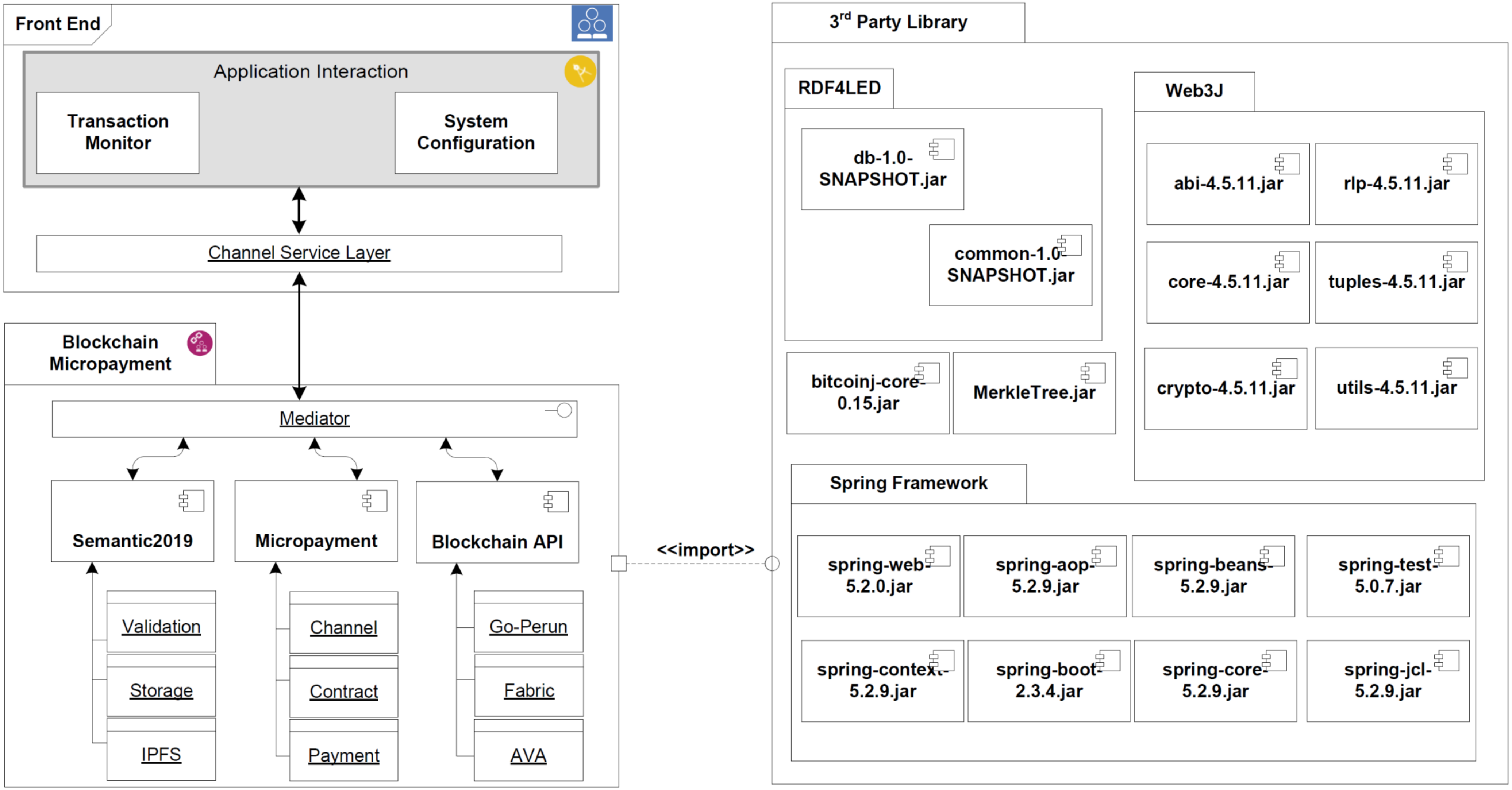
1. **Front End**
2. **Channel Service Layer**

   Channels are only a delivery layer, containing those functions necessary to manage delivery, while providing an appropriate client experience,

   All processing components should be available as services from underlying systems.
3. **Mediator Component**

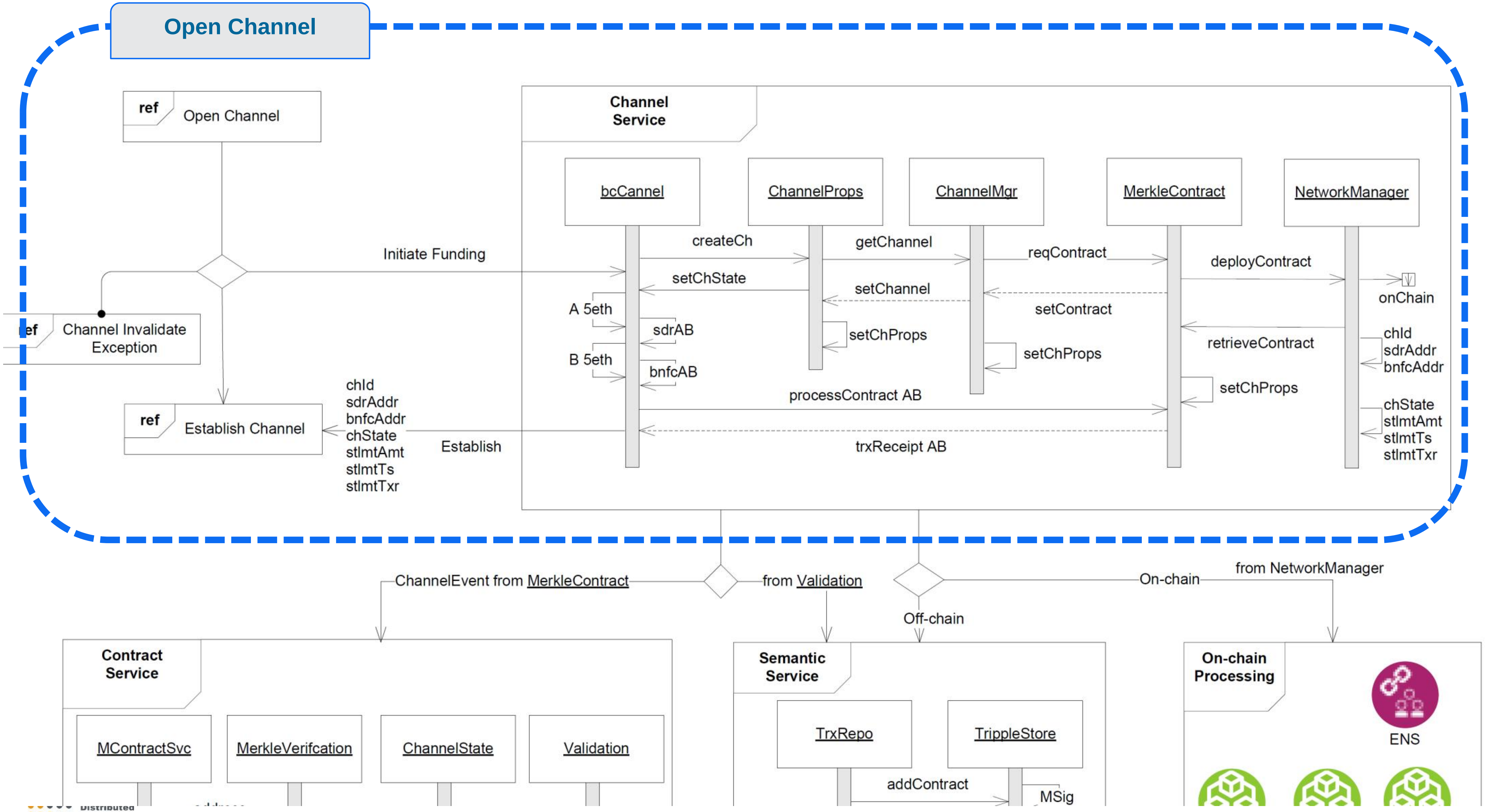
   Expose 2 core Services as APIs allowing multiple Front End to consume the same set of business services that covers enquiries, configuration and transaction services for Semantic and HSG88.
4. **Semantic2019 (without Blockchain)**
5. **HSG88 (enhance component)**

System Design - Component Diagram



Open Distributed Sytems
ODS

System Design - Interaction Diagram



Open Channel

ref Open Channel

Initiate Funding

ref Channel Invalidate Exception

ref Establish Channel

chId
sdrAddr
bnfcAddr
chState
stlmtAmt
stlmtTs
stlmtTxr

Establish

Channel Service

bcCannel

ChannelProps

ChannelMgr

MerkleContract

NetworkManager

createCh

getChannel

reqContract

deployContract

setChState

setChannel

onChain

A 5eth

setContract

sdrAB

setChProps

B 5eth

setChProps

retrieveContract

chId
sdrAddr
bnfcAddr

bnfcAB

setChProps

processContract AB

chState
stlmtAmt
stlmtTs
stlmtTxr

trxReceipt AB

ChannelEvent from MerkleContract

from Validation

On-chain

from NetworkManager

Off-chain

Contract Service

MContractSvc

MerkleVerifcation

ChannelState

Validation

Semantic Service

TrxRepo

TrippleStore

addContract

MSig

On-chain Processing

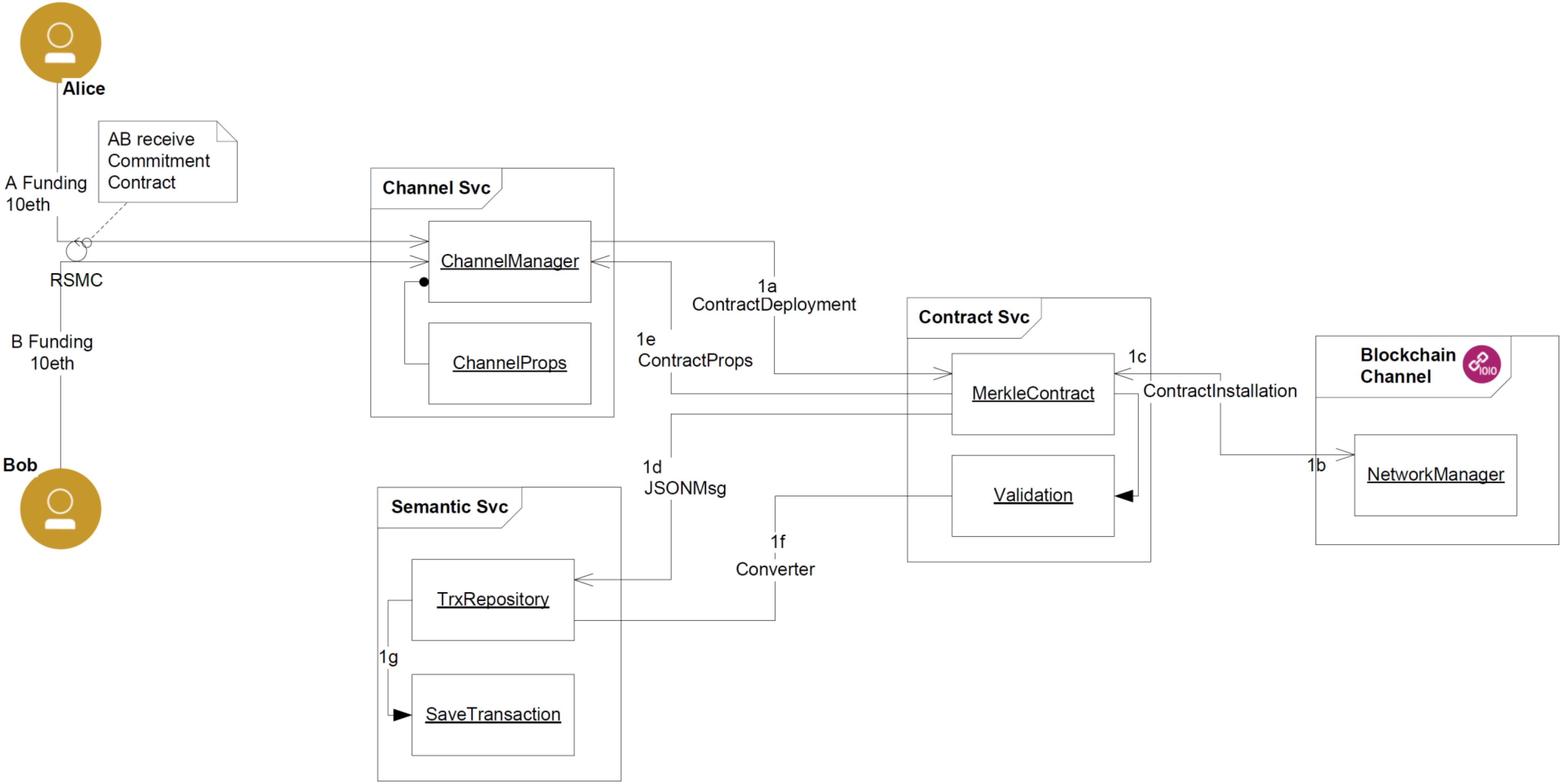
ENS

Distributed Sytems

ODS

System Design – Information Flow

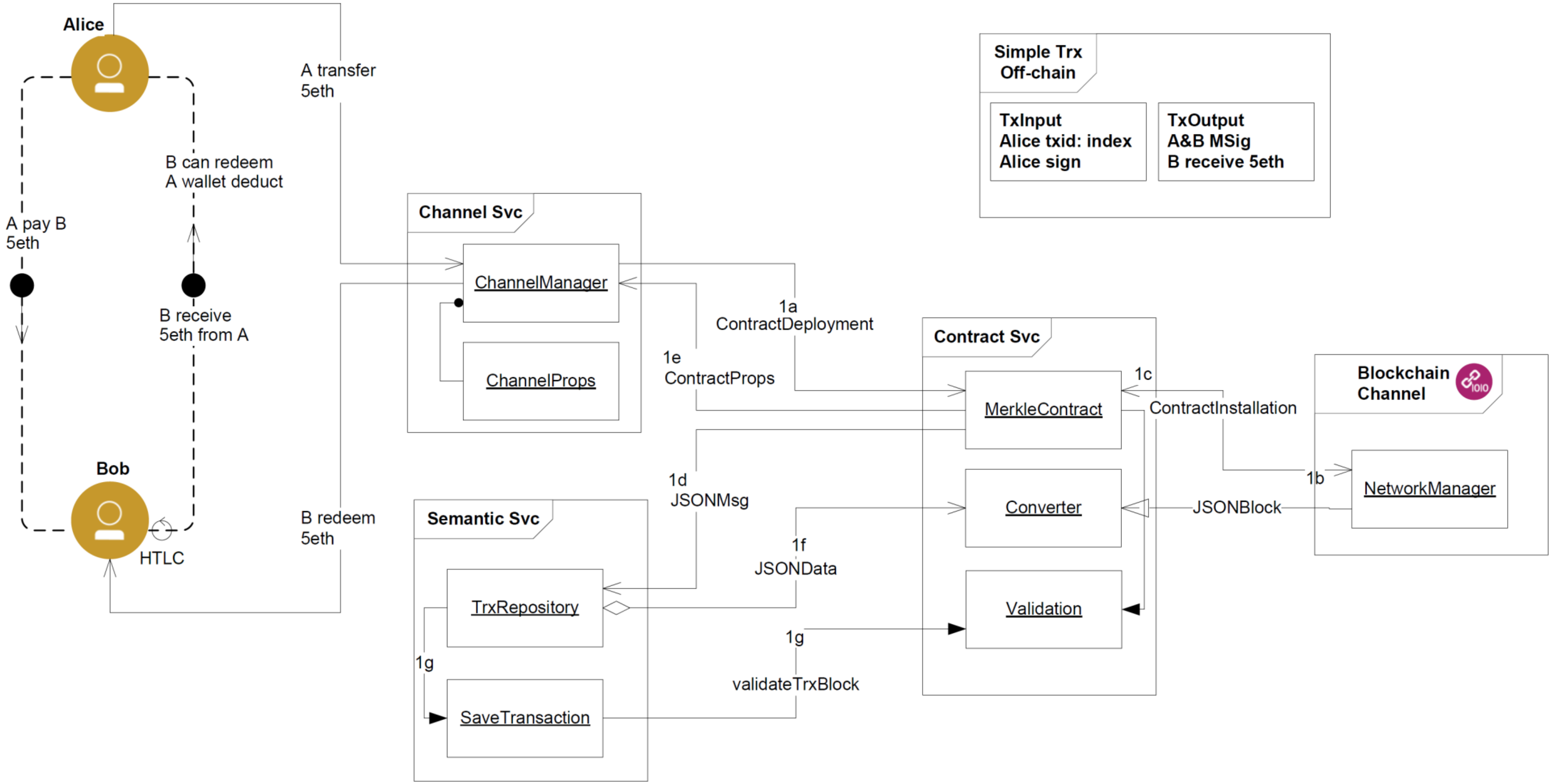
Open Channel – initial commitment



Open Distributed Sytems
ODS

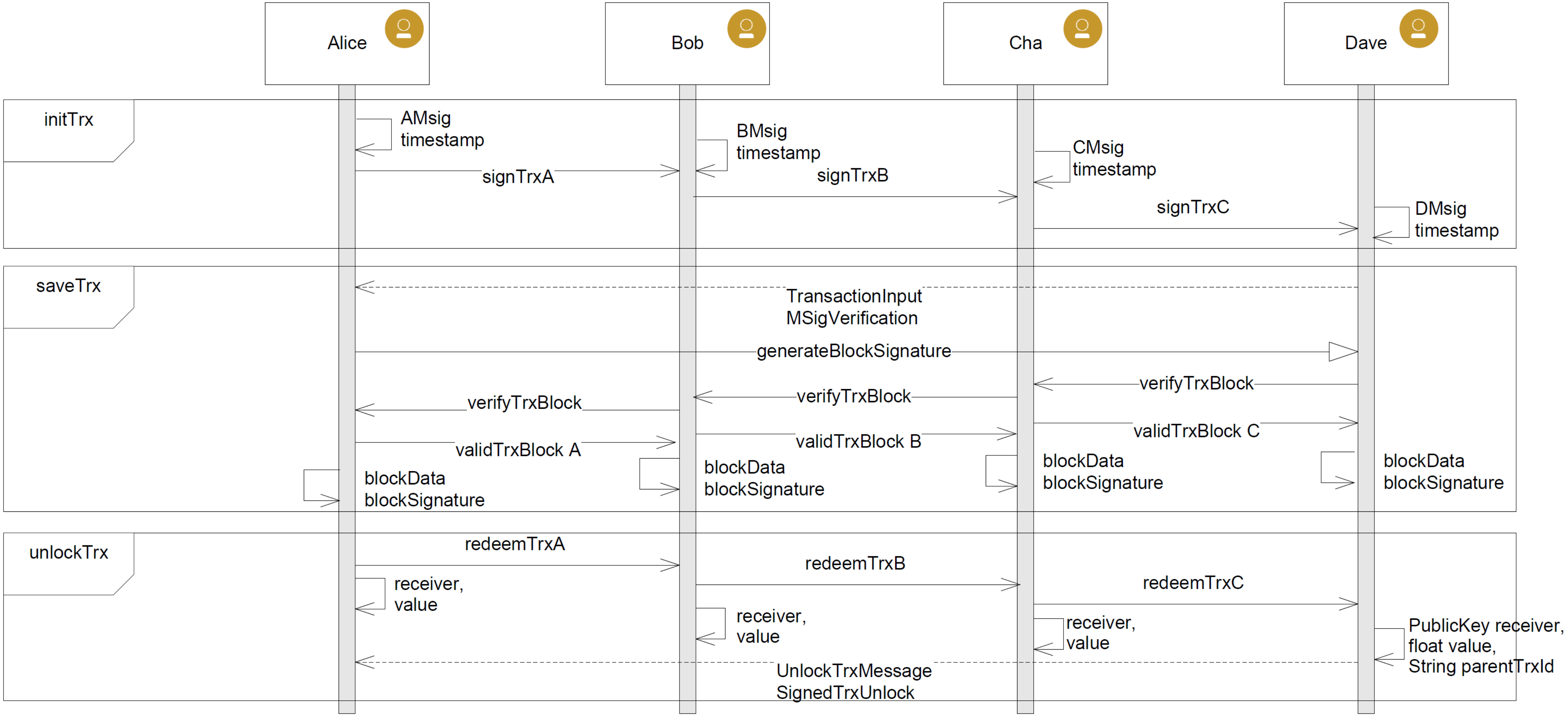
System Design – Information Flow

Transaction Flow: simple transaction – happy path (off-chain)



System Design – Interaction Flow

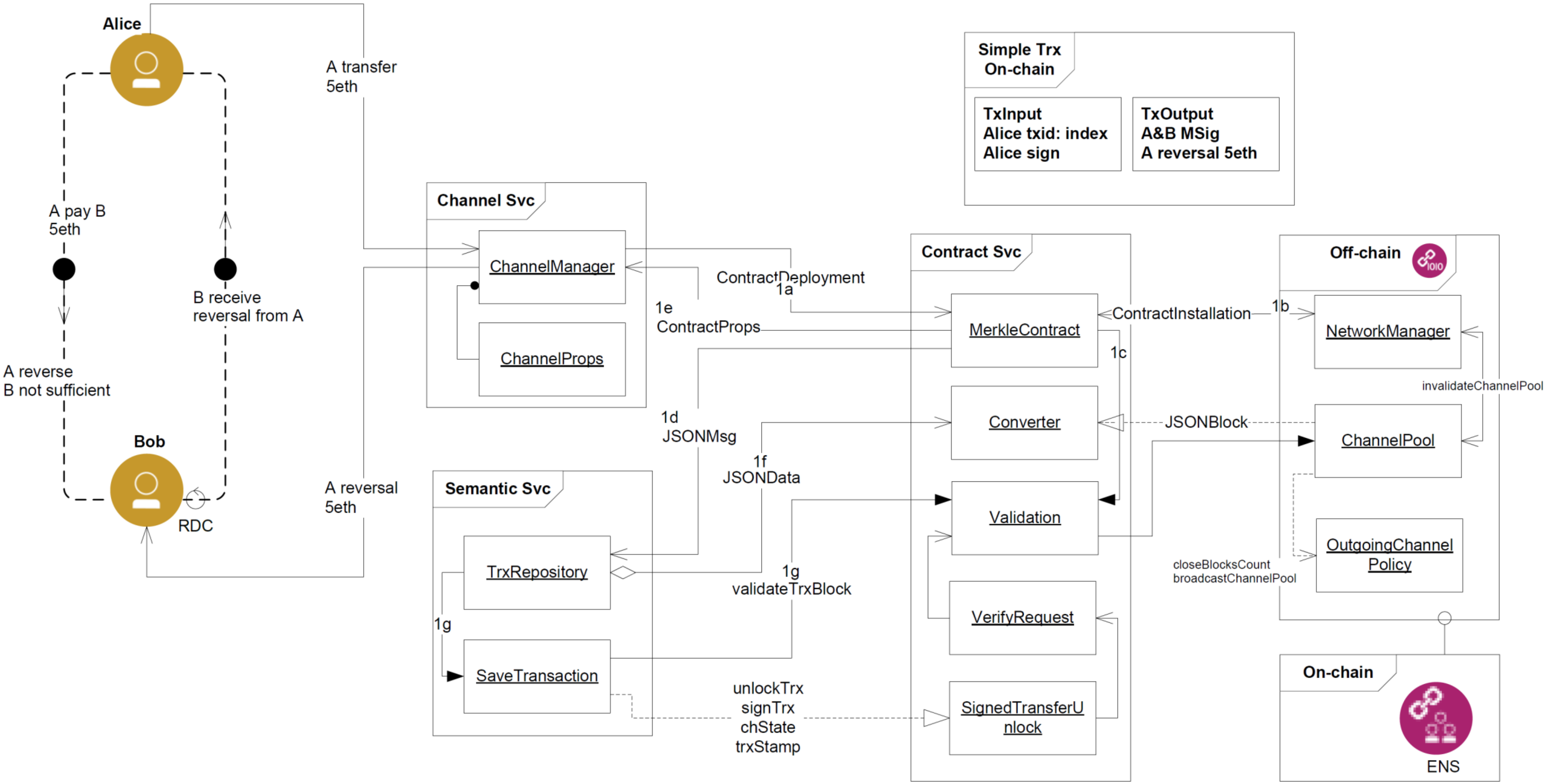
Sequence interaction: simple transaction – happy path (off-chain)



Open Distributed Sytems
ODS

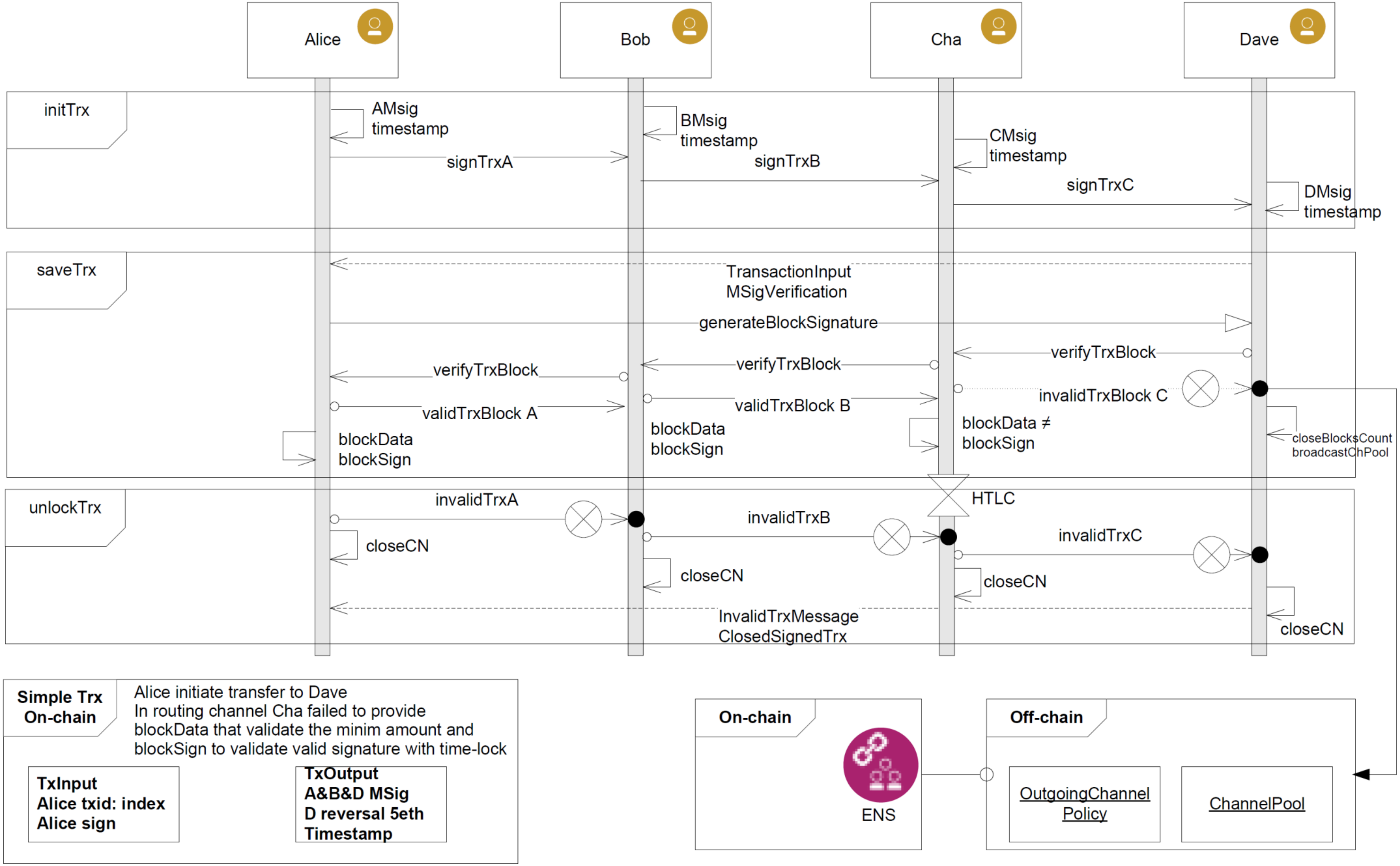
System Design – Information Flow

Creating a new commitment and reversal (on-chain)



System Design – Interaction Flow

Sequence interaction: new commitment and reversal (on-chain)



Alice

Bob

Cha

Dave

initTrx

AMsig
timestamp

signTrxA

BMsig
timestamp

signTrxB

CMsig
timestamp

signTrxC

DMsig
timestamp

saveTrx

TransactionInput
MSigVerification

generateBlockSignature

verifyTrxBlock

verifyTrxBlock

verifyTrxBlock

invalidTrxBlock C

validTrxBlock A

validTrxBlock B

blockData
blockSign

blockData
blockSign

blockData ≠
blockSign

closeBlocksCount
broadcastChPool

unlockTrx

HTLC

invalidTrxA

invalidTrxB

invalidTrxC

closeCN

closeCN

closeCN

closeCN

InvalidTrxMessage
ClosedSignedTrx

Simple Trx On-chain

Alice initiate transfer to Dave
In routing channel Cha failed to provide
blockData that validate the minim amount and
blockSign to validate valid signature with time-lock

TxInput
Alice txid: index
Alice sign

TxOutput
A&B&D MSig
D reversal 5eth
Timestamp

On-chain

ENS

Off-chain

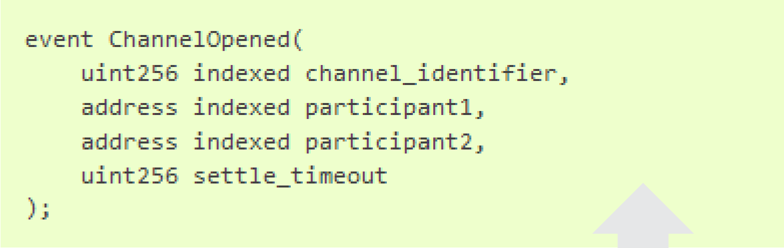
OutgoingChannel Policy

ChannelPool

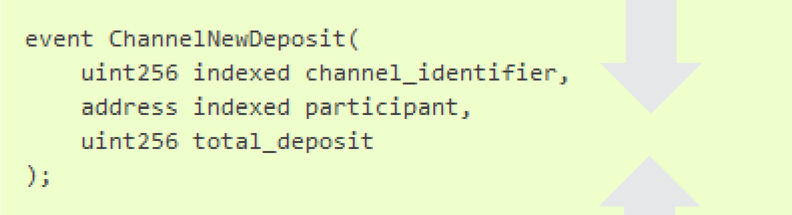
System Design – Data Structures

Raw message are used to convert existing field into persistence approach.

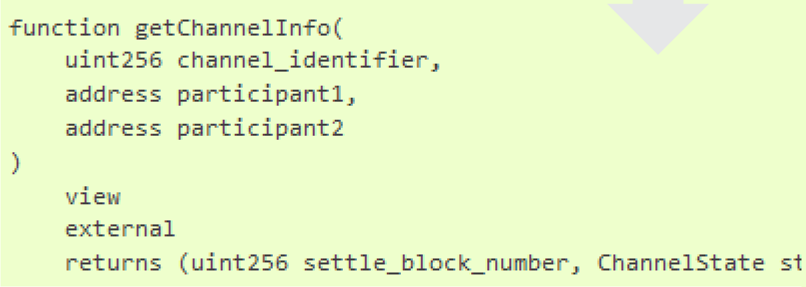
From On-chain Transaction → Converter → TrxRepository object



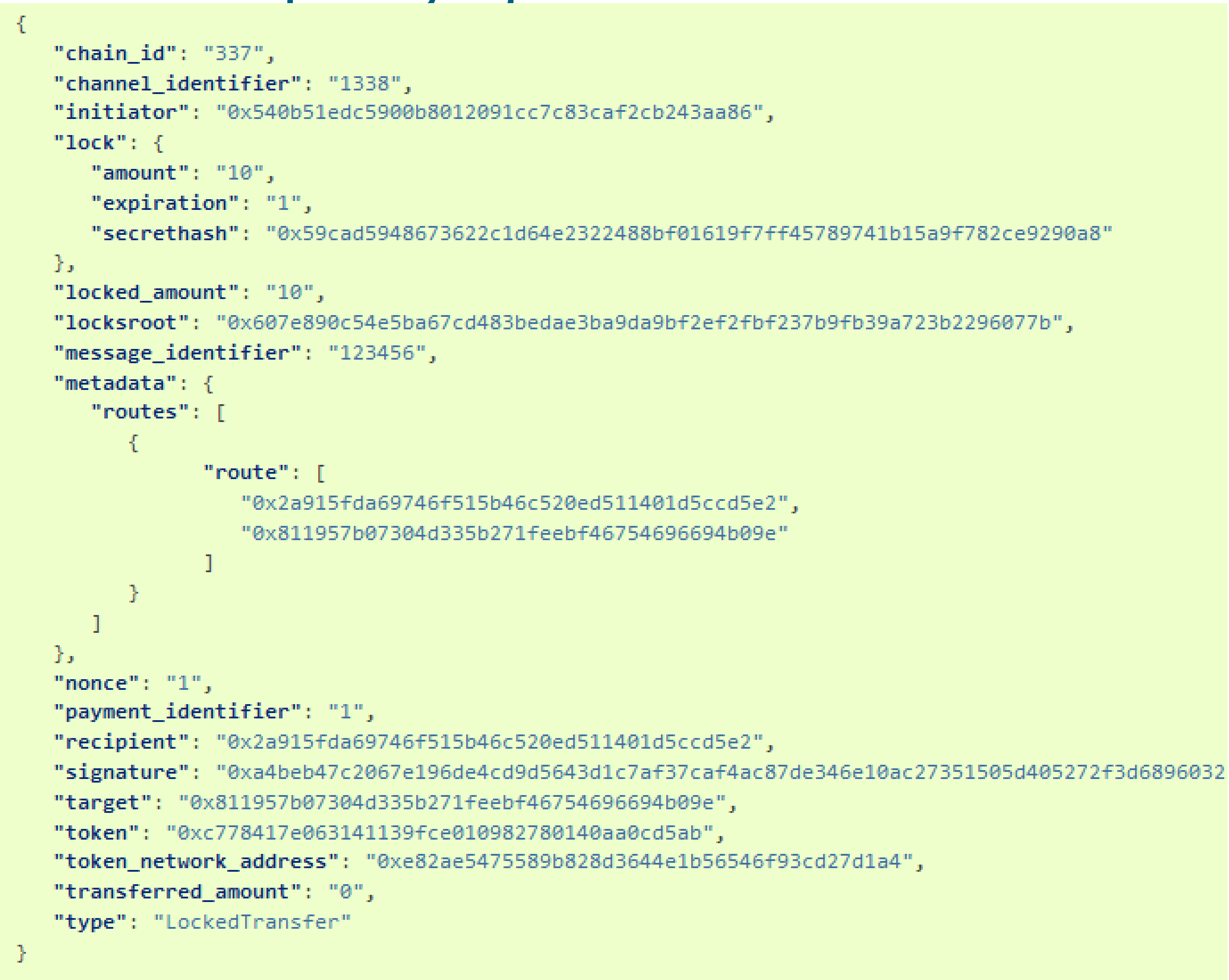
```
event ChannelOpened(
    uint256 indexed channel_identifier,
    address indexed participant1,
    address indexed participant2,
    uint256 settle_timeout
);
```



```
event ChannelNewDeposit(
    uint256 indexed channel_identifier,
    address indexed participant,
    uint256 total_deposit
);
```



```
function getChannelInfo(
    uint256 channel_identifier,
    address participant1,
    address participant2
)
    view
    external
    returns (uint256 settle_block_number, ChannelState st
```



```
{
   "chain_id": "337",
   "channel_identifier": "1338",
   "initiator": "0x540b51edc5900b8012091cc7c83caf2cb243aa86",
   "lock": {
      "amount": "10",
      "expiration": "1",
      "secrethash": "0x59cad5948673622c1d64e2322488bf01619f7ff45789741b15a9f782ce9290a8"
   },
   "locked_amount": "10",
   "locksroot": "0x607e890c54e5ba67cd483bedae3ba9da9bf2ef2fbf237b9fb39a723b2296077b",
   "message_identifier": "123456",
   "metadata": {
      "routes": [
         {
               "route": [
                  "0x2a915fda69746f515b46c520ed511401d5ccd5e2",
                  "0x811957b07304d335b271feebf46754696694b09e"
               ]
         }
      ]
   },
   "nonce": "1",
   "payment_identifier": "1",
   "recipient": "0x2a915fda69746f515b46c520ed511401d5ccd5e2",
   "signature": "0xa4beb47c2067e196de4cd9d5643d1c7af37caf4ac87de346e10ac27351505d405272f3d6896032
   "target": "0x811957b07304d335b271feebf46754696694b09e",
   "token": "0xc778417e063141139fce010982780140aa0cd5ab",
   "token_network_address": "0xe82ae5475589b828d3644e1b56546f93cd27d1a4",
   "transferred_amount": "0",
   "type": "LockedTransfer"
}
```

Open
Distributed
Sytems
ODS

PoC – Technical Demo

I. System Requirement

1. Oracle JDK or IBM JDK
2. Ethereum Test Network (e.g., AVA, Go-Perun, K-Channel, Raiden)
3. Minimum effort Ganache local setup
4. Redis
5. IPFS

II. PoC Scenario

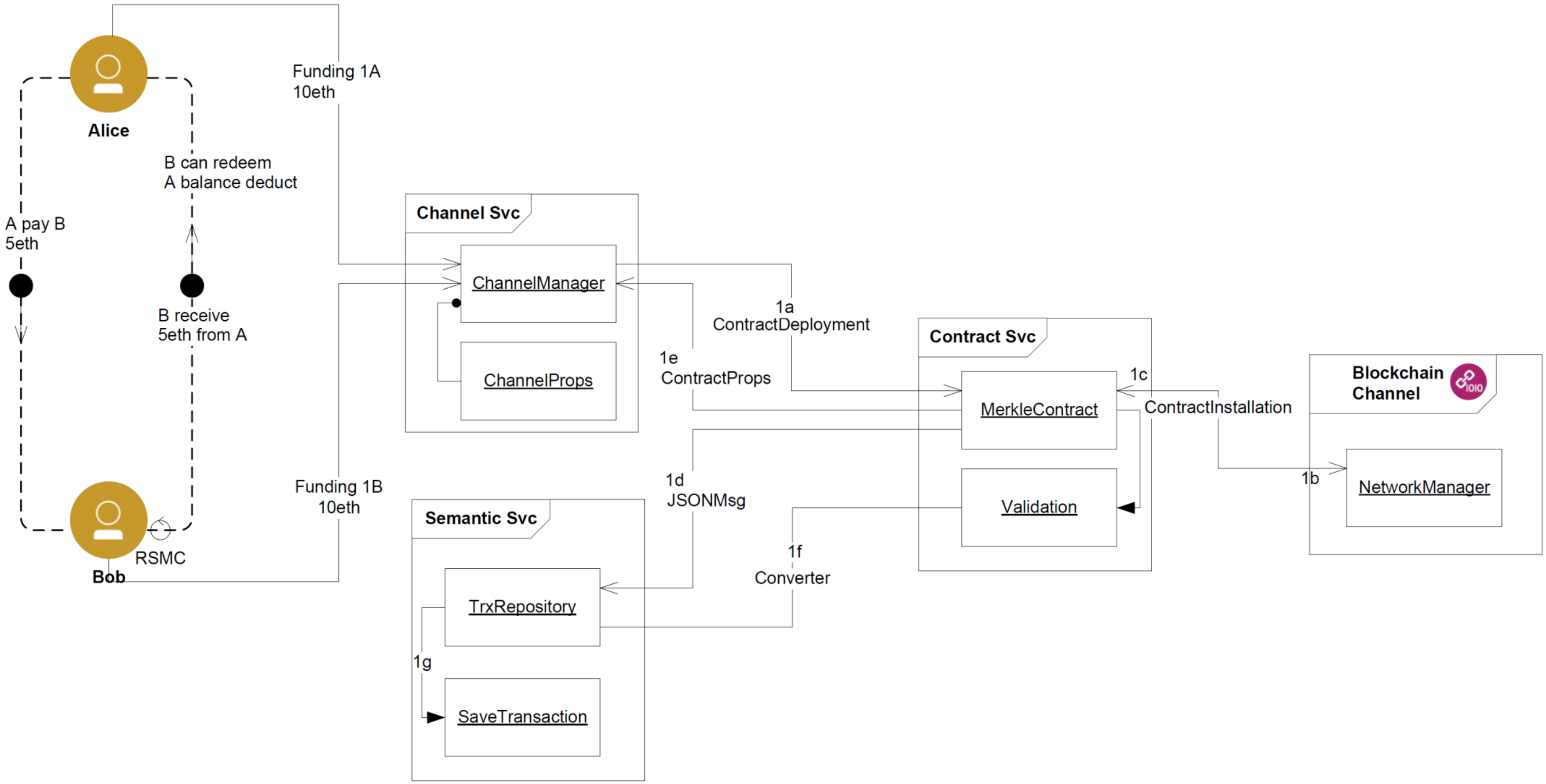
1. Channel lifecycle
    - open, establish, closed cooperatively
2. Simple payment transfer (off-chain)
    - Data transaction with local data store and Semantic2019
3. Simple payment transfer (on-chain)
    - Data transaction connected to Ethereum network test

Adds-on

S2 – DEVELOPMENT IN PROGRESS

System Design – Information Flow

Transaction Flow: simple transaction – unhappy path (on-chain)
New commitment transaction and reversal



System Design – Information Flow

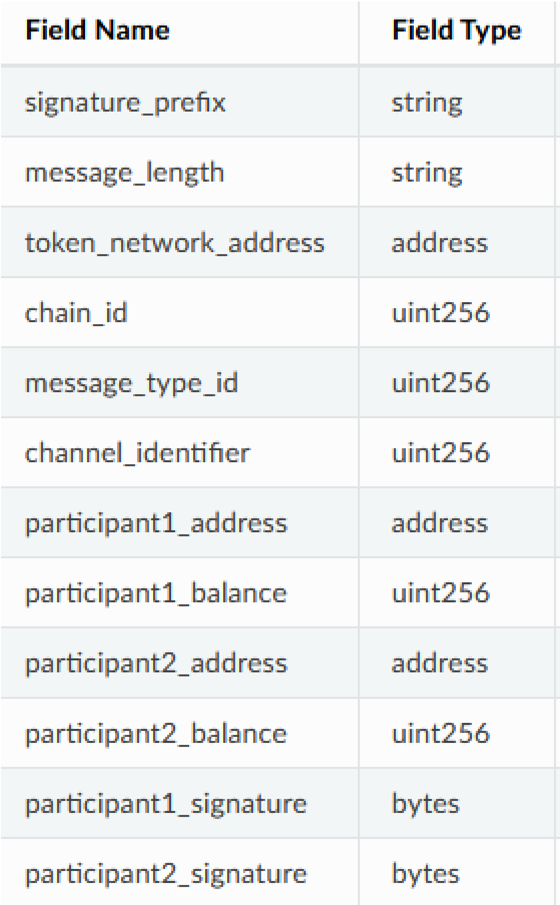
Creating a new commitment multi channel

System Design – Information Flow

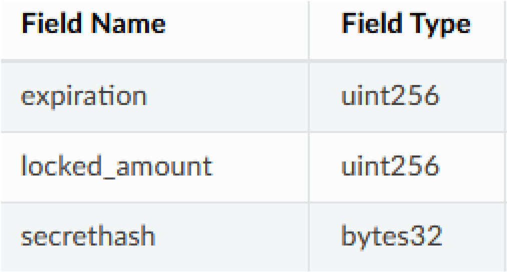
Creating a new multi channel commitment with dishonest from one participants

System Design – Data Structures

JSON Message are used to convert existing field into persistence approach.



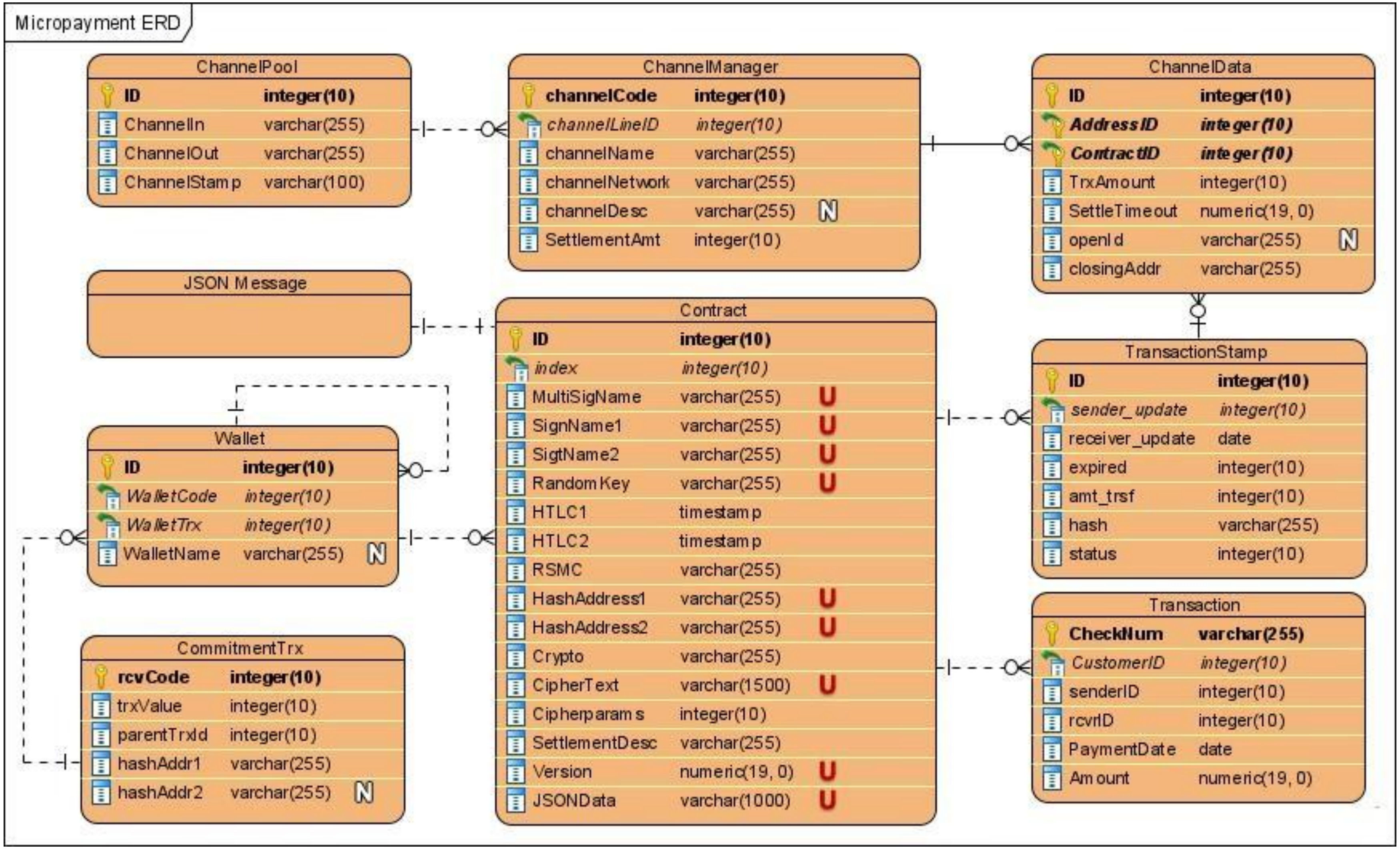
Field Name	Field Type
signature_prefix	string
message_length	string
token_network_address	address
chain_id	uint256
message_type_id	uint256
channel_identifier	uint256
participant1_address	address
participant1_balance	uint256
participant2_address	address
participant2_balance	uint256
participant1_signature	bytes
participant2_signature	bytes



Field Name	Field Type
expiration	uint256
locked_amount	uint256
secrethash	bytes32

System Design – ERD

Database – initial proposal



ChannelPool

ID	**integer(10)**	
ChannelIn	varchar(255)	
ChannelOut	varchar(255)	
ChannelStamp	varchar(100)	

ChannelManager

channelCode	**integer(10)**	
channelLineID	*integer(10)*	
channelName	varchar(255)	
channelNetwork	varchar(255)	
channelDesc	varchar(255)	N
SettlementAmt	integer(10)	

ChannelData

ID	**integer(10)**	
AddressID	***integer(10)***	
ContractID	***integer(10)***	
TrxAmount	integer(10)	
SettleTimeout	numeric(19, 0)	
openId	varchar(255)	N
closingAddr	varchar(255)	

JSON Message

Contract

ID	**integer(10)**	
index	*integer(10)*	
MultiSigName	varchar(255)	U
SignName1	varchar(255)	U
SigtName2	varchar(255)	U
RandomKey	varchar(255)	U
HTLC1	timestamp	
HTLC2	timestamp	
RSMC	varchar(255)	
HashAddress1	varchar(255)	U
HashAddress2	varchar(255)	U
Crypto	varchar(255)	
CipherText	varchar(1500)	U
Cipherparams	integer(10)	
SettlementDesc	varchar(255)	
Version	numeric(19, 0)	U
JSONData	varchar(1000)	U

Wallet

ID	**integer(10)**	
WalletCode	*integer(10)*	
WalletTrx	*integer(10)*	
WalletName	varchar(255)	N

CommitmentTrx

rcvCode	**integer(10)**	
trxValue	integer(10)	
parentTrxId	integer(10)	
hashAddr1	varchar(255)	
hashAddr2	varchar(255)	N

TransactionStamp

ID	**integer(10)**	
sender_update	*integer(10)*	
receiver_update	date	
expired	integer(10)	
amt_trsf	integer(10)	
hash	varchar(255)	
status	integer(10)	

Transaction

CheckNum	**varchar(255)**	
CustomerID	*integer(10)*	
senderID	integer(10)	
rcvrID	integer(10)	
PaymentDate	date	
Amount	numeric(19, 0)	