003 Data block Trading (DBK) Contract

Target release	
Epic	
Document status	DRAFT
Document owner	Alex Zabolotniy
Designer	Alex Zabolotniy
Developers	Alex Zabolotniy
QA	

General Economy Goals

- The goal of Solidity smart contract(s) is to provide the smart economy to Sample Protocol. more specifically to provide the below features:
    - User and data ownership registration, user permission management (Entity Management - EM Contract).
    - Data trading and client protection (Data Block Trading - DBK Contract).
    - Partnership organization, shares allocation for each data block and funds spreading (Partnership - PTR Contract).

DBK Contract Goals

- Trade data block (contract per Data block).
- Customer protection (receive and hold payments until verification or period of X days).
- Dispute mechanism (3rd party governance for dispute resolving).
- Integration to PTR contract (for funds spread between owners based on allocated shares).

~~Background and strategic fit~~

Assumptions

- DBK Contract is a template Solidity contract for deployment per request. (deployed by EM Contract).
- After a data user (registered entity in EM contract) registers a digital signature of his data block (in EM contract) then he allowed to request to deployment of a DBK contract for his data block to start trading.

Requirements

#	Title (Type)	User Story	Importance	Notes
1	**Data Types and Variables**			
2	**Request *(Data structure)***	Request struct{} use to represents the income purchase request for data block **Request.account** *(address type) - holds the account number of the data consumer.* **Request.value** *(uint256 type) - holds the amount sent with the request.* **Request.logTime** *(uint256 type) - holds the timestamp of the received request.* **Request.confirm** *(bool type) - holds the confirmation flag which raised by the consumer.* **Request.dispute** *(bool type) - holds the dispute flag*	MUST	
3	**requests *(Request[ ])***	Stores all (pending) purchase requests in an array	Must	Pending request which are received, verified and saved still not confirmed by the dat consumer

4	**consumerAddressToRequestIdx** ***(mapping(address=>uint256))***	Address to index converter	Must	
5	**EMAddress *(address)***	Variable to EM contract address	Must	
6	**EMContract** ***(contract object)***	Variable to EM contract object	Must	
7	**datablockOwner *(address)***	Account number of data-block owner(s)	Must	
8	**datablockHash *(bytes32)***	Digital signature of the data block (keccak256)	Must	
9	**datablockPrice *(uint256)***	The price for the data block	Must	
10	**daysLimit *(uint256)***	Period of days before auto confirmation	Must	Keeps the received funds fro specific data consumer for confirmation period of time dispute occurs the funds will released to data block owner
11	**CounterPurchased *(uint256)***	Counter for total data-block purchases	Must	
12	**CounterDisputed *(uint256)***	Counter for total purchase disputes	Must	
13	**idx *(uint256)***	Index pointer to **requests** array	Must	
14	**Events**			
15	**LogConfirmation()**	Log event when purchase confirmation received.	Must	Arguments: (address _sender,uint256 _idx)
16	**LogDispute()**	Log event if data consumer raised a dispute.	Must	Arguments: (address _sende
17	**LogWithdraw()**	Log when withdraw completed.	Must	Arguments: (address _consu
18	**LogBuyRequest()**	Log when a new purchase request is received.		Arguments: (address indexed _sender)
19	**LogNewDBK()**	Log when new DBK contract is deployed.		Arguments: (address indexed _owner, bytes32 _hash)

20	**Modifiers Functions**			
21	**datablockOwnerOnly()**	Limit function call to data block owner only	Must	Arguments: (address _acc)
22	**dataConsumerOnly()**	Limit function call to data consumer only (consumer is considered a data user who successfully purchased the data block)	???	Arguments: (address _acc)
23	**requestOwnerOnly()**	Limit function call to (data-block) purchase request owner Only	Must	Arguments: (address _acc)
24	**Functions**			
25	**constructor()**	Constructor function.	Must	Arguments: (address _ptr, by _hash), Modifiers: (public, payable, dataUserOnly(msg.sender)) Returns (bool success)
26	**kill()**	Destructor function	Must	
27	**()**	Payback function	Must	
28	**purchaseRequest()**	Send a function call transaction to DBK contract to purchase a data block.	Must	Arguments: (No), Modifiers: (public, payable, dataUserOnly(msg.sender)) Returns (bool success)
29	**withdrawAll()**	Withdraw all confirmed payments from the contract.	Must	Arguments: (No), Modifiers: (public, payable, datablockOwnerOnly(msg.se Returns (bool success)
30	**withdraw()**	Withdraw payment from specific purchase using request index.	Must	Arguments: (uint256 _idx), Modifiers: (public, payable, datablockOwnerOnly(msg.se Returns (bool success)
31	**disputeOpen()**	Data consumer will call this function to raise a dispute.		Arguments: (No), Modifiers: (public, payable, requestOwnerOnly(msg.send Returns (bool success)
32	**disputeCloseAndConfirm()**	Data consumer will call this function to raise a dispute.	Must	No Arguments: ()

33	**purchaseConfirm()**	Data consumer will confirm the purchase, allowing the data owner to withdraw the payment.	Must	No Arguments: ()
34	**changeOwner()**		Must	No Arguments: (address _ne Modifiers: (public, payable, datablockOwnerOnly(msg.se Returns (bool success)

User interaction and design

1. The DBK contract deployed for data owner account address (which is PTR contract account) and for specific data block signature (which must be registered in EM contract first)
2. Purchase process:
    1. Data consumer will generate function call transaction with the specified price
    2. DBK contract will verify the consumer and price, push the request to requests buffer and generate event on the blockchain.
    3. DBK will also hold the received funds till one of two confirmations occurs:
        1. A period of X days was passed
        2. Received confirmation from the consumer
3. Generate *LogConfirmation* event on the blockchain
4. Set the confirmed request in requests buffer to allow data owner to withdraw the funds.
5. Data owner withdraws the funds

Questions

Below is a list of questions to be addressed as a result of this requirements document:

Question	Outcome
What does data producer need to trade a data block?	1. To be registered as data user (in EM contract) 2. To be registered as hash owner (in EM contract) 3. Request from EM contract to deploy a DBK contract
What kind of information DBK contract holds?	• Digital signature of the traded data block • Account address number of owner's PTR contract

How DBK contract protects consumers?	DBK contract holds ALL income transaction till confirmation from the data consumer: When data consumer transfers payment to a DBK a request is added to an unconfirmed-request buffer and the payment stays in the contract account. DBK also generates an event on the blockchain notifying a new data-block purchase request. after consumer receives data block, he will confirm the request and the data producer will be able to pull the payment for the specific request. the consumer has a period of _X_ days to confirm the purchase or raise a dispute: 1. If request is confirmed by the consumer then the payment is ready to be pulled by the data owner 2. If dispute is raised then the payment is looked until dispute resolved.

Not Doing

-