001 Entity Management (EM) Contract

Target release	Sample Protocol V1.0
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).

EM Contract Goals

- User registration and permission control
- Deployment of shareholders (PTR) contract
- Deployment of data-block trading (DBK) contract

Background and strategic fit

-

EM Assumptions and notes:

- EM Contract is a singleton contract (one instance per economy)
- Entities added by a member-member policy.
- Data users (data producer/consumer) interact with contracts using function call transactions (no need to sign the data, just generate a transaction to the specific function in the contract)
- When a registered entity registers a new entity, the new address and its permission is updated but Authentication and Authorization flags aren't, when the new entity is

Authenticated the Authorization flag is set too. Keep in mind that if from some reason entity must be disabled, a higher entity will set its Authorization flag to false.

Requirements

#	Title (Type)	User Story	Importance	Notes
1	**Data Types and variables**			

2	**EntityType (enum data type)**	This variable will keep a list of predefined entity types. Entity type is included in each entity record to notify the EM contract about its permission level. --- **EntityType.UNKNOWN** (default) - This account number is unregistered - no permissions. --- **EntityType.OWNER** - This account number registered as Sample Protocol Foundation Owner - can register Foundation Administrators. --- **EntityType.ADMIN** - This account number registered as Sample Protocol Foundation Administrator - can register service providers. --- **EntityType.PROVIDER** - This account number registered as Sample Protocol Service Provider. can register Data users. --- **EntityType.USER** - account number registered as Sample Protocol Data user (Data Producer/Consumer) - can be registered as data block owner, trade and purchase data blocks. --- **EntityType.GUEST** - This account number registered as Sample Protocol Guest - no permissions	Must	• Please see the below description for each entity type

		EntityType.DBK - ... **EntityType.PTR** - ...		
3		**EntityType.ORC** - ...		
4	**Entity (data structure)**	This data structure is used to keep entity record, it has the below fields: **Entity.entityType -** (*EntityType type)* - holds the permission level of the specified entity. **Entity.parent** - (*address type)* - holds the registrar address of the entity. **Entity.authorized** - (*bool Type)* - holds the "Is Authorized" flag of an entity. informs the contract if this entity is Authorized to act in the network **Entity.authenticated** - (*bool Type)* - holds the "Is Authenticated" flag of an entity. Informs the contract if this entity is Authenticated by (defined authentication process like 3rd party KYC service provider)	Must	Entity is a data structure referenc by account addres
5		**Entity.reputation** - (*uint256 Type)* - holds the reputation value which is set by the community.		
6	**entityTable (mapping (address => Entity))**	Is a hash table which stores entity records - where the key type is address and the value type is Entity data structure	Must	keeps entity recor for each address

7	**hashOwnershipTable (mapping (bytes32=>address))**	Is a hash table which stores hash ownership records - where the key type is a 256-bit number (digital signature of a data block) and the value is address	Must	keeps address (of owner) for each digital signature
8	**Events**			
9	**LogNewContract()**	Log an event for the EM contract.	Must	Arguments: (addr indexed contractOwner, address indexed contractAddress, string contractName)
10	**LogNewFoundationOwner( )**	Log an event for registration of a new foundation owner.	Must	Arguments: (addr indexed foundationOwner)
11	**LogNewFoundationAdmin( )**	Log an event for registration of a new foundation administrator.	Must	Arguments: (addr indexed foundationAdmin address indexed ServiceProvider)
12	**LogNewServiceProvider( )**	Log an event for registration of a new service provider.	Must	Arguments: (addr indexed foundationAdmin address indexed ServiceProvider)
13	**LogNewDataUser( )**	Log an event for registration of a new data user.	Must	Arguments: (addr indexed entity, address indexed newEntity)
14	~~**LogNewDBK( )**~~	~~Log an event for registration of a new DBK contract~~ (instead, use DBK's constructor event)	DISABLED	Arguments: (addr indexed entity, bytes32 indexed hash)
15	~~**LogNewPTR( )**~~	~~Log an event for registration of a new Ptr contract~~ (instead, use PTR's constructor event)	DISABLED	Arguments: (Shareholder[ ] _owners)
16	**LogKill( )**	Log an event for killing the EM contract.	Must	Arguments: ()

17	**LogDeposit( )**	Log an event for depositing to the contract. Depositing means to transfer funds to the contract without calling any specific function.	Must	Arguments: (addr _src, uint256 _idx
18	**Modifiers Functions**			
19	**PassIfTrue()**	general modifier - pass if provided condition evaluated to true	Must	Arguments: (bool test)
20	**EntityOnly()**	pass if 'msg.sender' is an entity in the contract	Must	Arguments: (addr _acc)
21	**NotEntityOnly**	pass if 'msg.sender' is NOT an entity in the contract30	Must	Arguments: (addr _acc)
22	**AuthorizedEntityOnly()**	pass if 'msg.sender' is an authorized entity	Must	Arguments: (addr _acc)
23	**foundationOwnerOnly()**	pass if 'msg.sender' is an authorized foundation owner	Must	Arguments: (addr _acc)
24	**foundationAdminOnly()**	pass if 'msg.sender' is an authorized foundation admin	Must	Arguments: (addr _acc)
25	**serviceProviderOnly()**	pass if 'msg.sender' is an authorized service provider	Must	Arguments: (addr _acc)
26	**dataUserOnly()**	pass if 'msg.sender' is an authorized data user	Must	Arguments: (addr _acc)
27	**dataOwnerOnly()**	pass if 'msg.sender' is owner of registered _hash	Must	Arguments: (address _acc, bytes32 _hash)
28	**Functions**			
29	**constructor( )**	constructor Function	Must	
30	**kill( )**	destructor Function	Must	
31	**( )**	Payback Function	Must	
32	**checkHashOwnership( )**	Allows to verify if specific address is the owner of specific signature	Must	Arguments: (addr _address, bytes32 _hash)
33	**addFoundationOwner( )**		Must	Arguments: (addr _newAddress)

34	**addFoundationAdmin( )**	Allows foundation owner to add foundation admin. The registrar must be an authorized foundation owner. The registrar will become admin's parent.	Must	Arguments: (addr _newAddress)
35	**addServiceProvider( )**	Allows foundation admin to add service provider. The registrar must be an authorized foundation admin. The registrar will become service provider's parent.	Must	Arguments: (addr _newAddress)
36	**addDataUser( )**	Allows service provider to add data user (data consumer or producer). The registrar must be an authorized service provider. The registrar will become data user's parent.	Must	Arguments: (addr _address)
37	**locateDBK( ) ???**	Get the DBK-contract address of specific data-block (get address links digital-signatures/hash values (keccak-256) of data-block to)		Arguments: (bytes _hash) Returns: (address DBK)
38	**deployPTR( )**	The function caller asks to deploy partnership contract: 1. The partnership group representative(any shareholder from the partnership group) will call **EM.deployPTR( )** function contract, providing it a list of initial shareholders accounts and their absolute amount of shares. 2. **EM** contract will verify the shareholder list provided to **deployPTR( )** function making sure every entity in it is registered. 3. **EM** contract deploys **PTR** contract and registers its address as entity in **entityTable** with **EntityType.PTR** as its entity type. 4. Deployed **PTR** stores **EM**-contract's address for future referencing.	Must	Arguments: (Shareholder[ ] _partners)c

39	~~**registerHash( )**~~	This internal function sets **hashOwnershipTable** to provide ~~This function caller asks to declare ownership over a data-block, by mapping data-block's digital signature (keccak-256) to a **PTR** contract address by setting **hashOwnershipTable**.~~	DISABLED	Arguments: (bytes _hash)
40	**deployDBK( )**	The function caller asks to deploy a **DBK** contract: 1. The caller is a group representative (any shareholder from the partnership group) will call **EM.deployDBK( )** function contract, providing it a list of initial shareholders accounts and their absolute amount of shares.	Must	Arguments: (addr _acc, bytes32 _ha
41	**getEntity( )**	returns Entity structure of the required address.	Must	Arguments: (addr _acc) Returns: (address Entity)

User interaction and design

1. The account address which deploys the EM Contract becomes an entity with Foundation-Owner permissions. (known as the "first entity")
2. each entity, after registration must be authenticated in and authorized (authentication process must be separated from the registration process)
    1. Authentication flag is set when the specified entity successfully passed authentication process by its parent (it done once, after registration).
    2. Authorization flag will be set automatically after successful authentication process.
    3. Authorization flag is used to disable an entity. this may be done by higher level permission entity.
    4. an entity may be unauthenticated and unauthorized or,
       authenticated and authorized or,
       authenticated and not authorized
       but never, unauthenticated and authorized.
3. A data producer (entity with data-user permissions) willing to declare an ownership over a data block:

1. will send a function call transaction named: registerHash() where the EM contract will verify sender account address and the hash number and (if successfully verified) will register the hash in reference to the sender's account address.

FAQ

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

Question	Outcome
Why we included Data ownership registration mapping in EM contract?	We need a single location where we can register and link between data-block's digital signature (keccak256(data))and owners account number. hence we use a Solidity Mapping to do that and we do it in EM contract because of: 1. Up till now, EM contract is the single singleton contract. 2. To deploy a special contract just to keep a single mapping table - will result unnecessary
What is an entity and why we use this title?	An Entity is anything capable of generating transactions and function calls, for example: data users (data producers/consumers), other contracts, foundation owners, foundation administrators and service provider.
suggestions for contract/entity names:	1. Permission Management (PM) Contract 2. Entity Permission Management (EPM) Contract.
Why do we need EM contract?	EM Contract is the main contract in the network - it keeps a record per each (registered) account, a record which specify permission level and important indicators like Authorization and Authentication flags and parent account of the record. EM contract also deploys other template contracts as needed.
How EM Contract will recognize users?	EM contract stores and recognizes only address numbers, accounts permission level, account's parent account and flag indicators. In other words, the contract doesn't stores personal information.
Who keeps and where my personal information?	Personal information stored on KYC/AML servers of 3rd party service provider
What is the entity registration procedure?	???
How reputation works ?	

Questions

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

Question	Outcome
s	

Stories

Below is a list user manual of the contract:

Story	Notes

Not Doing

-