
HAIN-IN-LAW

TABLE OF CONTENTS

1.
Introduction.

Our civilization is built upon the law — a complex system of norms governing the behavior of individuals. This complex construction was built to get us a sustainable structure (namely society) consisting of unreliable components — the people.

Thousands of years since the first legal relations emerged, a similar problem was solved by building a reliable system for unreliable elements called the **Distributed Ledger Technology (DLT).**

In the real world relations of the subjects lead to the creation of legal relations governed by certain rules — the norms of law. Legal relations and their changes can be represented as structures that can be grouped as directed acyclic graphs. The passage of time sets a(n) "orientation" (vector) and "acyclicity" of relationships: since one arises, it is valid until changed or terminated by a new relationship. Recognizing the right as absent would not erase it from the history but influence the future consequences.

For the emergence of a legal relationship, it is necessary to have legally defined subjects and a legal composition — the required aggregation of legal facts, i.e. legally significant actions and events with which the rule of law or the terms of the contract link the emergence of legal relations. At the same time, the key element of a legally significant action is the will of the subject of the legal relationship. It must be unequivocally interpreted action of the subject and have the direction of the emergence (change or termination) of the relationship.

The development of cryptography and the widespread introduction of asymmetric ciphers into practice allowed us to legally identify a subject and determine the fact of his will through the use of his private key. Usage of time synchronization between all network nodes allowed to reliably determine the corresponding point in time.

> Today technologies let us build a universal cryptoplatform incorporated in the existing legal system, which allows users to enter into various legal relationships as easily as

> possible while simultaneously capturing them in the form of permanent records in a distributed registry.

To build such a platform, we only need to determine the legality of actions for a unit (node) of the network. As we tried to find the most common solution, we divided the classical consensus of the distributed registry into two levels — at the application level (user-defined transaction processing and block building) and at the network level, which ensures the stability and integrity of the network. Thus, users can independently determine the composition of the nodes, the rules for joining new nodes, and the order of processing transactions. This allowed us to create a tool for constructing legally defined procedures for servicing a wide class of business processes.

2.
Platform basics.
2.1. Network status and legality.

Key users.

When using the Platform, users join the general Basic Agreement, which defines all the basic rights and obligations of users of the system. According to it, Users have the right to create their own rules for processing transactions (hereinafter referred to as Rules) and to determine the composition of nodes (Validators) who are entitled to process a certain type of transaction.

In addition to the transaction processing rules, Users can define in the Rules other rights and obligations of Validators and Users who have selected Validator data for processing their transactions. This creates many distributed registries that handle various business processes.

When sending a transaction to the network, the user indicates the composition of the Validators authorized to process this transaction. Thus, the User explicitly expresses his will to enter into certain legal relations.

Any user can download (https://github.com/SilentNotaryEcosystem/Cil-core), install and run a full network node. But in order to become a Validator, he must join one or more Rules (join the Consilium). If this user meets the requirements defined in the Rules, he will be accepted by the network for processing the parts of transactions that correspond to these Rules.

It is necessary to understand that the Rules define not only the order of processing transactions, but can also impose a number of restrictions on the composition of the nodes, up to the indication of a closed list of addresses. Also, the adoption of the Rules may lead to the adoption of certain rights and obligations related to the processing of business transactions.

Juridical entity.

> According to the classical definition,
> **Entity is a person (physical, legal, or other entity) for which the law recognizes the ability to have, directly or through a representative of the law, legal obligations.**
>
> From the point of view of the theoretic system representation,
> **Entity is a carrier of subjectivity (an individual, a legal entity, state authority), having certain interests and opportunities (resources).**

We will analyze these constructions in detail in separate works, but currently for the functioning of the Platform we will express the following concepts related to the definition of the subject in the Platform's system objects:

- the identifying signs of the Entity (see the Entity Identification section);
- and the Entity's expression of will (see the section «Entity Will»).

Entity Identification.

Entity can be identified by a variety of (possibly infinite) features. This can be either:

- passport data, certified by third parties;
- fingerprints and other biometric data;
- state registration data (registration numbers), including:
  - data of birth;
  - place of registration;
  - and other.

Some subsets of this feature set are standard sets of personal identification (KYC requirements) and serve as a condition for the emergence of a legal relationship (the enter the process, see the section Legal Relationship below).

> A fundamental method of identification is the use of asymmetric encryption and pairs of public and private keys. It is considered that only the Subject knows his private key, and since the public key is uniquely tied to the private key (one unique key corresponds to a single unique private key), the public key is the Subject's identifier. This method has long been introduced into the business through
> the use of EDS.

Consider the concept of identification more detailed.

Introduced references:

Entity, Public key, Private key, Compromised key, Witness, Identifying attribute.

Entity is considered to have a private key set. Each private key has its own unique public key. Therefore, Entity can be put in a one-to-one correspondence with some set of public keys.

However, some of the private keys may be compromised: they may become known to third parties and the Subject must be able to exclude these keys from circulation. Thus, each key has its own period of validity: from the moment of its creation until the moment of compromise (or the deletion from the registry by the Entity).

Accounting for compromised keys is made by creating public registries, with the involvement of specially authorized subjects—witnesses certifying this compromise. Thus, the set of compromised keys is defined as the set of records of the form (Public Key; Certifying Entity; Compromise Time).

Each Entity has infinite content in the real world. To identify it, we use some easily verifiable features—identifying information. Thus, identification can be represented as a set consisting of pairs (Entity’s Public key; Identification attribute).

But, since the identification is subjective, namely, with the involvement of an identifying Entity (Certifying Witness—US), the design is complicated: a set consisting of records of the form signed with a private US key (US Public Key; Identifying Character; Entity's Public Key).

The next feature is the fact that the Entity owns his personal data, which is expressed in a number of national laws, for example, GDPR. The general approach is to determine the right of the Entity to conceal such data and the need for the explicit Entity's will to provide this data to third parties.

Although the identifying information may be encrypted, it may be the case that the set of attributes associated with the same Entity's Public key will allow its identification. Thus, it is required to bind identifying features not directly to the Entity's Public key, but to the Permitted Entity Identifier, i.e. to the secondary key generated from the original Public key. To generate such keys, various algorithms for creating temporary keys can be used.

In general, such an algorithm requires a special code to decrypt the value of the Public key. We call such a code "Consent to access to personal data" (in short—the Code of Consent). This code can be valid for a certain time, or be indefinite.

Essentially, the Allowed Identifier looks like a record of the form (Allowed Identifier Code, Encryption Algorithm Code). Entity's accidence to get access to his data can now be defined as (Permitted Identifier, Consent Code, Entity Public Code), transmitted to the receiving party via a secure communication channel, excluding third party access.

Example of implementation:
Encryption algorithm code = SHA2 (256). Allowed ID code = Hash ("Public Code" & "Consent Code"). Authorized identifier = (Hash ("Public Code" & "Consent Code"; SHA2 (256))
Now we can **define Entity's identification as a number of consisting of records of the form (Public key of the CS; Identification attribute; Allowed Identifier), signed by the keys of the corresponding CSS**.

Entity's Will.

For entry into most legal relations the Entity's Will is required to be explicitly expressed.

We use the electronic digital signature (EDS) method, which is widely used in business circulation, to record the Will. Each transaction sent by the user to the network is signed by EDS. The user can utilize any standard of the encryption standards supported by the system, which allows the use of local national EDS standards. Therefore, a transaction sent to the network and signed by the User's EDS uniquely determines his Will.

In any transaction there is a field ("witnessGroupId") defining a group of Validators (Consilium) authorized by the User to process this transaction. This group of Validators acts in accordance with the Rules, which are a public contract with the algorithmic part (user consensus).

The completed witnessGroupId field of the signed User's EDS uniquely identifies the will of the User to enter into legal relations in accordance with the Rules corresponding to this witnessGroupId value.

Legal objects.

Users can use objects defined outside the Platform, as well as define legal objects using the Platform's system objects. The basic process of identifying an object is similar to the Entity identification (see the section Entity Identification), except for the following features:

- rights for the object may be alienated or restricted;
- the object does not have subjective rights to its identifying information (personal data), but some of the information may constitute a commercial or other type of secret.

Thus, an object of law can be associated with a token issued in accordance with a special procedure (a unique cryptographic identifier), and its identifying features can be associated with a set of transactions of the type (such as Public Key of the Certifying Witness [CS] Identification Tag, Allowed Token Identifier) signed by the keys of the corresponding CA. If the information about the object is of a public nature, the Permitted Token Identifier coincides with the Token Identifier (Address).

Since any actions with the token are made only with the help of transactions signed by the EDS, it is easy to record the will of the parties and associate them with legal regulation.

Legal relations.

The main scheme is the representation of the legal relationship in the form of an abstract process with the allocation of the main roles (legal entities, objects of the legal relationship, the content of the legal relationship).

> Legal relations can be built on the basis of existing ones, forming a network of processes in the form of a directed acyclic graph. The entry of a legal relationship can be a legal (unencumbered) exit of an already completed process (a legal relations that has arisen earlier), forming a transaction network based on the UTXO model.

As we described earlier, the Legal Entity is defined through the address space with cloud bind of its identifying features, including those confirmed by third parties (one of the special cases is the KYC procedure).

Communication with a real Entity is provided through a private key, the expression of the subject's will is determined by his implicit action in the form of using his private key when signing a transaction. The object of a relationship is defined as a token of the Platform with the possible binding of a cloud of identifying features and conditions for its legal use.

When we use a distributed registry for fixing legal relations i, users receive the opportunity to choose the method of validation, i.e. the choice of the consensus used between the decision makers to include the transaction in the registry of the network nodes. For this purpose, subsets from the set of validators, are combined and united by a general agreement and a consensus algorithm (the software part of a legally binding agreement).

Let us illustrate the description of the legal relations in more detail.

As we previously postulated, the main function of the system is the creation, modification and termination of legal relations between users of the system with simultaneous fixation in the form of permanent entries in the distributed registry. Creating legal relations is possible as a result of the Entity's active actions when using the Platform.

We consider the relationship as a process of occurrence of the rights and obligations of the subjects of the relationship regarding the object of the relationship in accordance with the norms of law governing these relations.

As an abstract concept, a process has an entrance space, a state change, and an exit space.

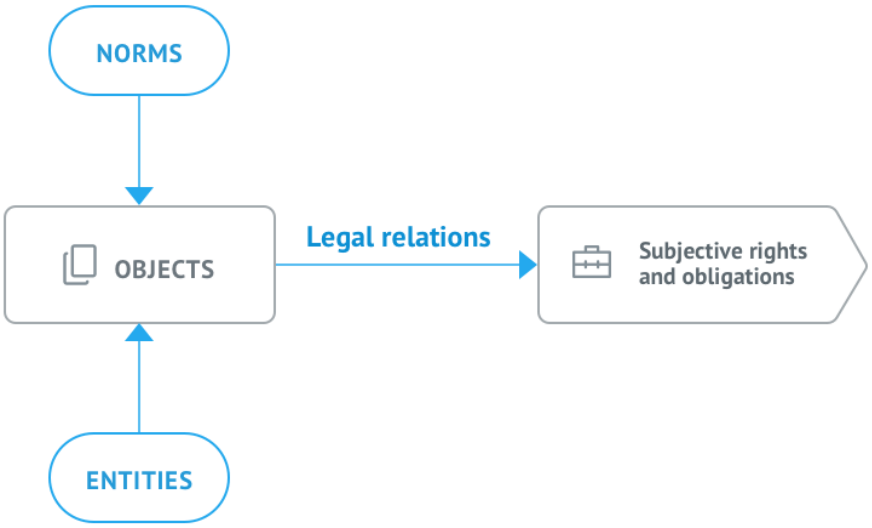


Diagram: the process of the legal relations occurrence.

The input space includes Entities (not yet entered into a legal relations), objects (not yet involved in this legal relations), legal norms regulating the order of occurrence of this legal relations, Entity's Will - parties to the legal relations, various environmental parameters (system environment).

A change (in terms of a process) is the content of a legal relations — a change in the Entity's rights and obligations in the legal relations.

The output is the Entity's subjective rights and obligations regarding the objects of law. Of course, formally, the output also includes Entities (with new rights and obligations) and objects (which became the object of a relationship, for example, those that have become the subject of a transaction).

Outputs of one process can be inputs of other processes, forming a network of processes or, in our case, a network of legal relations. Since we are dealing with completed processes (and time is irreversible), this network is directed towards past processes and is a directed acyclic graph.

Diagram: legal relations network (acyclic graph).

At the same time, if a legal relationship is built on the basis of already existing legal relations, then it uses the unspent outputs of this legal relationship allowed for use, which is very close to the UTXO model.

2.2. General description.

The exchange of User information with network units – nodes, occurs through transactions. The transaction contains a witnessGroupId field that defines the Rules to which the User (the sender of the transaction) joins and in accordance with which its transaction must be processed. When a transaction enters the network, it begins to spread across the network, passing from node to node. A number of transactions form a mempool – a temporary depository of transactions. Upon the occurrence of the conditions specified in the Rules, the group of Validators that have joined the Rules (nodes implementing the algorithmic part of the Rules, also called the Consilium), selects the transactions that they must process and form a block. After the transaction has been confirmed (included in the block), it is removed from the memory. Next, the block falls into the general network storage, built on the principle of a directed acyclic graph. The blocks are connected in accordance with the rules of Meta-consensus (the network level of consensus of the Platform distributed registry), which ensures the integrity of the network.

2.3. Platform Layers.

The platform is divided into five layers. Each of the layers can be reworked without affecting other layers.

Application layer.

At this layer applications access the network using RPC protocol.

Main applications developed so far:

SilentNotary – user interfaces (web, messenger bots, email, mobile apps, API) for hashing and cloud storage of files. Used for notary verification and item tokenization. See section "Identification").

Wallets – standard applications for payment transactions.**Explorer** – a tool for browsing graph,

blocks, transactions and smart contracts. On top of that Explorer, the Explorer is able to review node groups and their consensus rules

Node consensus layer.

Our platform implements a two-level distributed registry consensus:

- **Meta-consensus** (across the network layer) ensures the integrity of the network and counteracts some types of attacks;
- **Node group consensus** (application layer) contains the rules for processing transactions inside a certain group of nodes. A group of nodes could belong to the company or the government structure and have their business logic backed up with the group consensus rules.

Node group consensus layer. Consiliums.

Network **nodes** can form subsets that implement certain rules. Each group is called a "**concilium**" — also the name of a procedure in ancient Rome. Conciliums required a meeting of people familiar with the laws to help make a decision.

Every node of the network can join any Consilium, if it is allowed by the *Rules of the respective Consilium*. A node that joins a Consilium is called the **Validator**.

The platform supports an unlimited number of Consiliums and Validators.

The methods for Consilium creation looks like this:

```
message WitnessGroupDefinition {
        repeated bytes publicKeys = 1;
        bytes groupContractAddress = 2;
    uint32 groupId = 3;
    uint32 quorum = 4;
    uint32 minFee = 5;
    uint32 contractMultiplier = 6;
    repeated bytes delegatesPublicKeys = 7;
    }
```

where:

publicKeys – an array of keys that form a group of Validators (Consilium) ;

groupName – is the address of the Consilium contract ;

groupId – Consilium index ;

quorum – quorum requirement ;

minFee – the minimum transaction fee. Cannot be less than transaction fee across the system;
contractMultiplier – a multiplier that determines the commission for the execution of the contract;
delegatesPublicKeys – are delegate keys that can determine the decisions of the Consilium.

This method defines two sets of rules:

1. Consilium evolution rules.

How nodes can enter or exit the Consilium and how to change the rules of the Consilium.

2. Consensus rules of the Consilium.

Any consensus model could be chosen inside the Consilium – private , PoW , PoS , delegated PoS, federated, etc... can be arbitrarily determined and modified following the Consilium evolution rules.

Network consensus level. Meta-consensus.

Meta-consensus defines the general rules for adding blocks and block connectivity.

As other distributed ledgers CIL uses the Merkle tree to ensure data integrity. The tree is constructed by recording unit hash values of the previous blocks (we call it the block "parental link"). The rules determining the correct parental links as well as the procedure for resolving conflict situations (for example, double spending) constitutes the content of the logic of Meta-consensus. In general, when choosing parent hashes, we adhere to the principle of forming a partially ordered set of transactions in a chain of blocks in relation to the used transaction outputs.

> The general structure of Meta-consensus-linked blocks is a directed acyclic graph, where the nodes are the vertices, and the edges are Merkle links between theblocks. When the load on the network increases, the graph "expands", while decreasing it "shrinks". In the edge case it turns into a classic hash chain as in common blockchains.

Meta consensus rules ensure the following:

Data integrity is provided by the well tested solution: the Merkle tree connects all the blocks into one network. Since all transactions interconnect via edges, they form a partially ordered

set. The basic protocol rule or the **"Parental Choice" rule** looks like this: **processed block must refer to all the previous blocks that contain the outputs of spent transactions.**

Block requirements:
All transactions must be ordered within the block by their submission time. It is not allowed to have transactions that consume the output of a later (next in order) transaction.

Double spending prevention.
Anti-spam (overloading the volume of the node).
Resistance to network overload (DDOS-attack).

Network layer.

Developing the platform we used the modular architecture. This nets a sufficiently high flexibility and the possibility of a fairly simple network evolution: it is enough to replace one unit with another, and the system will keep working seamlessly.

Network technical solutions:

Cryptography: SHA -3 (**github link**);

Serialization: Protocol buffer (**github link**);

Transport: IPv6 (**github link**);

Storage: LevelDb;

Transaction standard: UTXO (**github link**);

Smart Contracts: JavaScript VM (**github link**);

Hardware level.

We have developed a hardware solution for the rapid deployment of network nodes. We use single-board computers with a SSD drive running optimized Linux OS. The node computers also have communication ports (LAN , WiFi and USB) for EDS. Node computers come with a pre-installed and partially synchronized CIL node client.

Below is an example of the SilentNotary Consilium node:

The cost of a designer node ranges from 200 to 500 USD.

Hardware r&d goals:

1. **Reducing the entry complexity** by transitioning to a bundled solution. This is crucial for the implementation of the System in state institutions and for business saving on non-core competencies.
2. **Increased security.** We use a specially assembled Linux kernel with the exclusion of components unnecessary for the Platform's operation, which mitigates hacking threats. Also, separating CIL node from a workstation PC significantly increases its reliability.

There are options to use specially licensed locally produced hardware to meet local technical requirements - both corporate or state.

3.
Technical description of the platform.

The platform is a distributed registry with multiple user-defined consensuses.

3.1. General architecture.

Chain in Law is a distributed registry Platform comprised of a directed acyclic graph (DAG), of blocks produced by different Consiliums (node groups) and assembled in accordance with the network rules (Meta consensus) and Consilium rules.

3.2. Transactions.

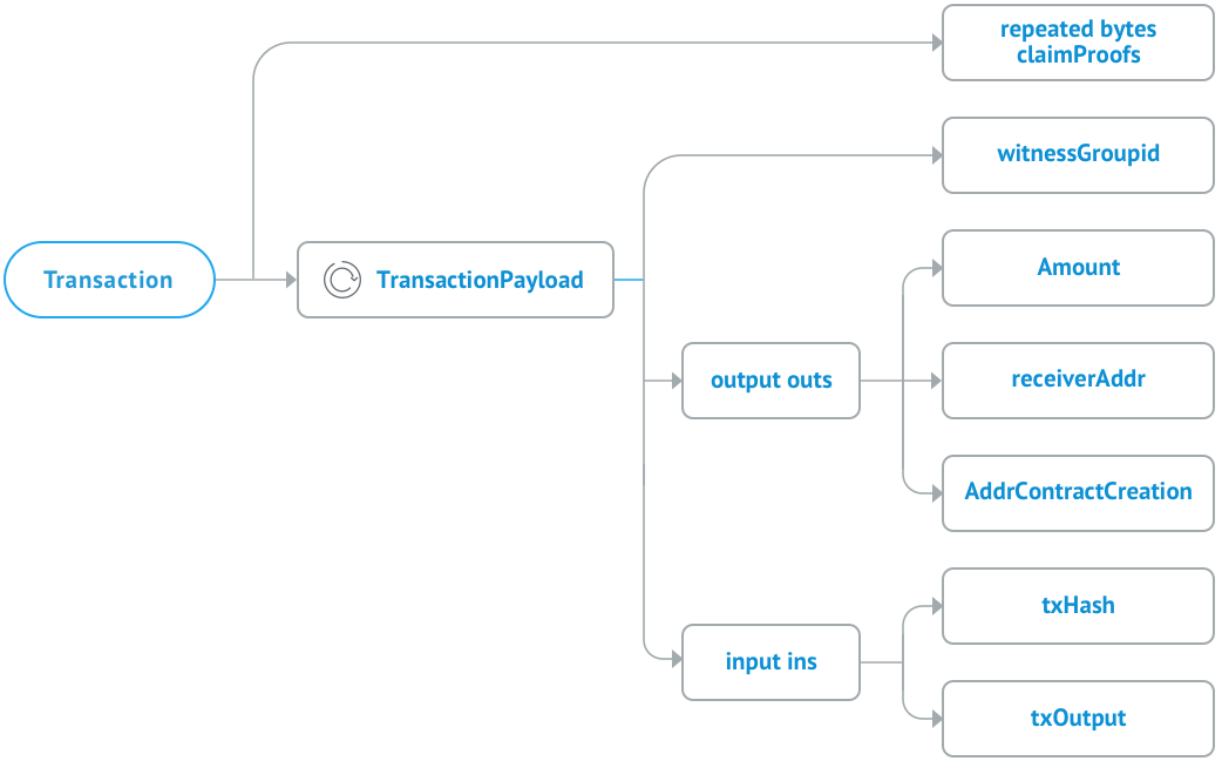
Transactions are based on the UTXO model. The main feature of the transaction in the platform is witnessGroupId, defining a group of validators (Consilium) and an authorized user (sender transaction).

An example of a transaction is shown below:

```
message input {
    bytes txHash = 1;
    uint 32 nTxOutput = 2;}
message output {
 possibly here should be uint64 or bytes (BN)
fixed64 amount = 1;
 payment receiver
    bytes receiverAddr = 2; 3
     if receiverAddr is AddrContractCreation
    bytes contractCode = 3;}
message TransactionPayload {
uint32 version = 3;
uint32 witnessGroupId = 4;
 place it fields
repeated input ins = 1;
repeated output outs = 2;}
message Transaction {
```

```
 place here non hashed fields
    TransactionPayload payload = 1;
repeated bytes claimProofs = 2;}
```

The general structure of the transaction is shown in the diagram below:



The value of the hash function from Payload is a TransactionHash which is also transaction ID in the network.

3.3. Blocks.

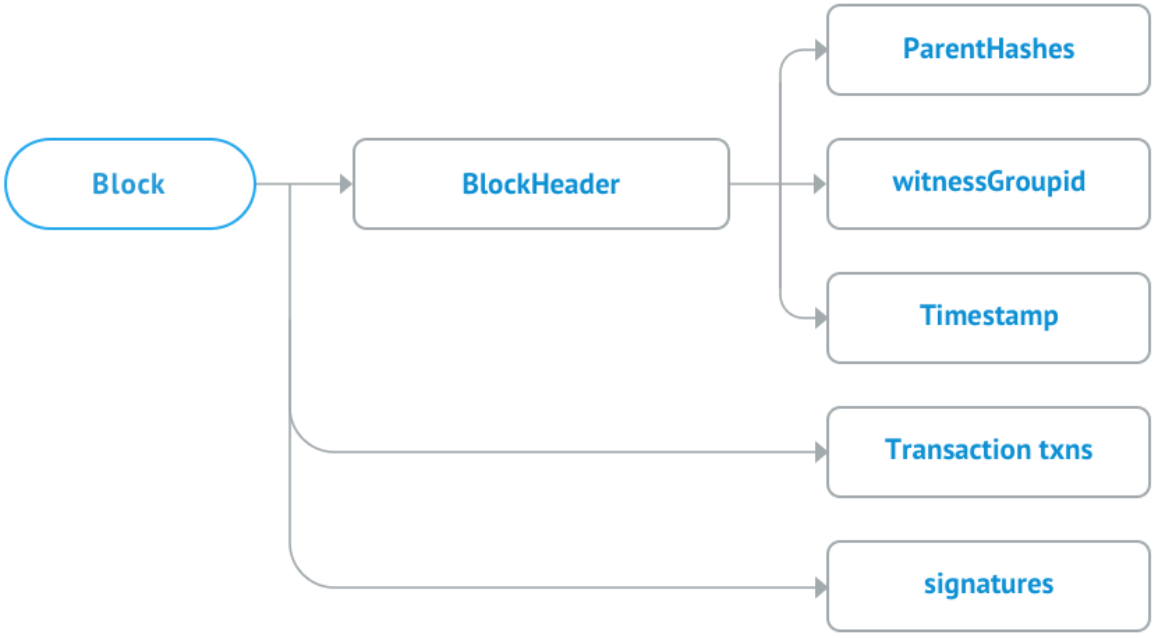
Blocks consist of transactions with the same witnessGroupId field value. Blocks are formed by Consiliums in accordance with the Rules. The block structure is generally similar to the bitcoin block structure except witnessGroupId, which determines its belonging to a certain Consilium.

Example of a block serialization:

```
message BlockHeader {
repeated bytes parentHashes = 1;
bytes merkleRoot = 2;
uint32 witnessGroupId = 3;
uint32 timestamp = 6;
uint32 version = 7;}

message Block {
    BlockHeader header = 1;
repeated Transaction txns = 2;
repeated bytes signatures = 3;}
```

The general structure of the block is shown in the diagram:



3.4. Smart Contracts.

Currently, smart contracts are implemented in JavaScript – one of the most used programming languages in the world.

The smart contract is processed inside of the Consilium in which it was uploaded. A smart contract processed in one consulium may invoke methods (and thus activate) smart contracts from other Consilium. This allows you to make your smart contract a procedural part of a legal agreement in a stand-alone code.

3.5. Gas prices, currency of payment.

Since smart contracts are created in the Turing-complete language, a restriction on the consumed computing power is required. Our platform implements the standard mechanism for charging for the computing power necessary to execute a smart contract.

The cost of the smart contract is determined as follows: defining computational complexity of this smart contract with a special gas price calculator. At the same time, the Consilium, guided by its own rules, can set a multiplying factor (contract Multiplier) for the cost of contract execution by its nodes.

> When one runs a smart contract using the standard procedure, the transaction during the execution of a smart contract method transfers the required amount of gas in the payment account as the cost of its implementation. If the smart contract exceeds the paid amount, then the smart contract execution is stopped, nothing is written to the registry, commissions are charged and distributed in accordance with the rules of the Consensus.

4.
Main modules of the platform business logic.

In addition to the development of the Platform and interfaces, we are developing a number of versatile solutions representing the implementation of the most frequently used business processes. These are lego blocks (or a library of models) from which various business solutions can be constructed. **The first pre-built modules are:**

4.1. Personal Identification Module (KYC).

Identification or authentication is a necessary part of almost any service.

The KYC module has several functions:

A) Service and identification (ID Verification Operations): login, load of identifying data, calculation of hash, record of transaction with reference to identifiable subject. It is executed by a SilentNotary interface.

B) Identity verification: search by ID hash, search by ID subject, ID verification (hash calculation and comparison with record), verification subject (decoding his true identity), search for existing black-list (records of unreliable persons), and search for compromised keys that will be implemented in Network Explorer.
For the identification procedure the User is able to determine the nodes that process these transactions. This allows geo-referencing of a node group to meet the requirements of the legislation on handling personal data (e.g. GDPR).

4.2. Tokenization.

Token is a digital ID of objects that may be involved in a relationship in the real world *(including intangible values such as IP.).*
Token is a specific construct, the meaning of which is created by users by entering into various agreements. These agreements should take into account the requirements of the Transaction Processing Rules or be fully contained in such Rules.

As an example, a token can be used to represent a vehicle. A car's token is issued by its manufacturer with the production of a real car. Consilium nodes that handle token smart

contract identify the manufacturer and include a network of authorized dealers and authorized service centers. This car token enables the tracking of the entire history of the car:a change of ownership, maintenance, repairs, insurance, etc.

Currently tokenization of various types of objects is being developed. For example: cars, objects of art and historical value, licenses for aquatic farming.
The Token creation process is facilitated as much as possible. The process was divided into two smart contracts — the registry-contract and the token contract. Thus, the user token is created using the register-contract while simultaneously classifying a token in the unified registry of tokens. During the token's life cycle, the user can assign additional features and properties, based on the changing needs.

The main purpose of the classification is to build a registry of all things. This problem is solved by constructing a polyfactor structure:

Let X be the set of all objects. Classifying means a partition of the set on any basis (i.e. – all warm-blooded animals or not, in this case the partition is on the grounds of "warm-blooded"). We will use the operation Boolean B (X) – as sets of all subsets of the original set X (i.e., the elements of B (X) are subsets). Then sets of partitions S is contained in B (x):

$$S \epsilon B(X)$$

In this case, we require the execution of two axioms from splits:

$$S_i \cap S_j = \emptyset$$

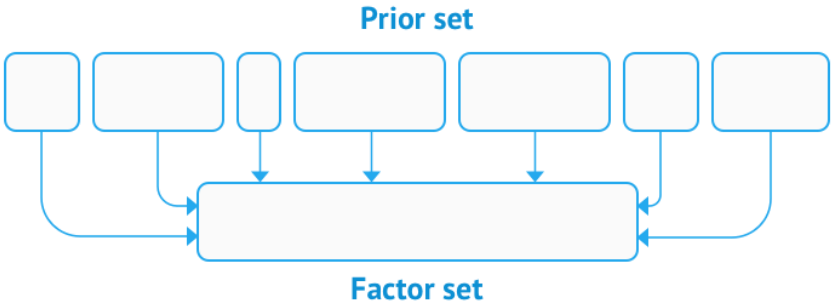
$$S_i \cup S_j = X$$

It was argued that a plurality of partitions corresponds to a plurality of attributes. Here is an example graphical representation:

The prior set X consist of several objects

These elements of the partition we will call **classes**. You can enter the concept of nested partitions into subdivisions, partitions of independent classes of the original partition.

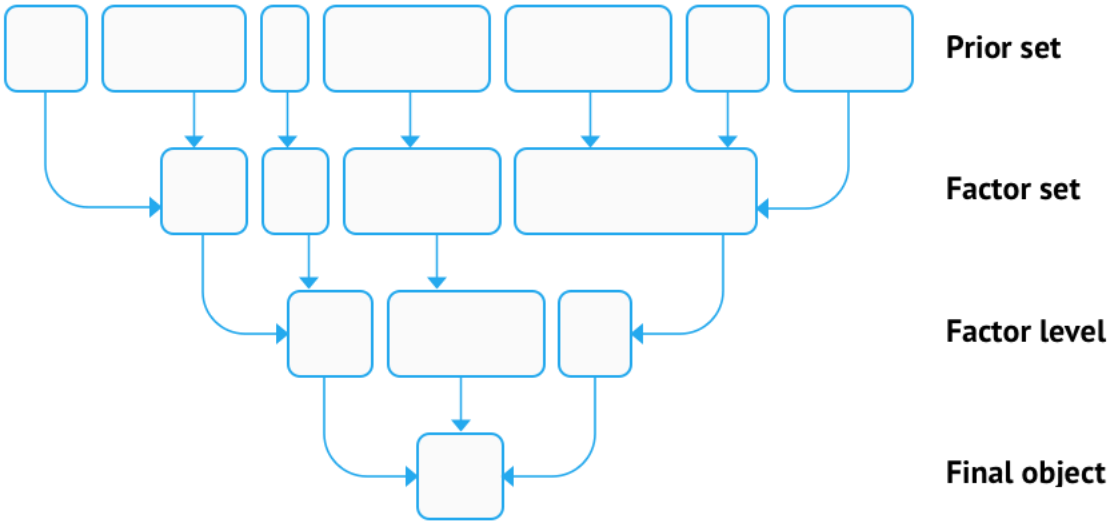
Thus, we create a set consisting of a plurality of elements of the source, the so-called **factor set**.



We often do this procedure at the household level, for example, if the initial set is a set of people, then the set factor can consist of two elements, men and women.

Further we will carry out a splitting of the Factor-Sets.

Factor partition-set is not the original object's partitioning but the partition classes of the original objects. You can repeat this procedure several times by moving to the next factor-level until one element remains:



The resulting structure is called a **factor structure**. It is possible to build a structure factor by any other indication such as the initial partition We are going to do – build another factor.

A set of these structures is called a **polyfactor structure**. We can establish relationships between the elements of these structures; when these relations can include several structures, then we can talk about polyfactorial relations. The task of creating the token is the task of building the appropriate multifactor relationships. This design is descriptive for all the possible diversity of the empirical world.

4.3. The voting module.

In the course of the implementation of business, processes often need to quickly get feedback from a large number of users, and to do so in a legally meaningful way. It may be a decision to shift, let's say, the management company of the investment fund unit holders, the meeting of creditors of the borrower, or just any elections—a representative of a group of people to delegate their interests.

5.
Conclusion.

Our vision for the development direction of distributed registry technology is to introduce network nodes into a legal space, dividing consensus into network and application levels and, as a result, creating an ecosystem of deeply integrated distributed registries located in a single network - in a common space addresses and transactions. The future of technology is not in opposition to the existing order of things, but in integration into the legal system and its organic change from within.

We thank all our followers, our friends and colleagues who, with their advice, with their kind words, and with their hard work, have helped us to create a new distributed platform that we believe will have an impact on the lives of an enormous number of peopleThe main part of our platform has already been developed and tested. The source code (Node . Js) is available on Github at: https://github.com/SilentNotaryEcosystem/Cil-core

We will be grateful for any contribution to the development of our platform and we are open to any advice and comments.

We also encourage developers interested in using our platform to contact our team for clarification. We will be happy to help you to integrate our DLT into your project.