The Kalima API

To actually understand the operating of the examples in this GitHub repository, it's important to understand a few notions about the Kalima Blockchain.

1. Notary Nodes

A Kalima blockchain is composed of Notary Nodes an ordinary Nodes also called client Nodes. The Notary Nodes are particular nodes, but they are essential because it's thanks to them that other nodes will directly communicate. The Notary Nodes are responsible of the validation of all the transaction on the Blockchain. They ensure the traceability, as well as the integrity and the immutability of all the transactions. We can install as many Notary Nodes as we want to set up a Kalima blockchain, with a minimum of 4 Notary nodes.

2. Client Nodes

The Client Nodes allow connection to the blockchain, synchronization in the cache memory data they're authorized for, interaction with that data and the creation of new transactions.

Every example of this repository corresponds to client nodes in different forms, for example in Java and for Android.

3. Cache memory/ history

In Kalima, history data which corresponds to all the transaction that have happened on the Blockchain since its creation needs to be differentiated from data in cache memory which is data that is current.

Every transaction is automatically saved in a history which is shared between all the Notary Nodes. That history can't be modified. No data can be deleted.

However, the « Cache Memory » is shared between the Notary Nodes but also with all the other nodes (in part, depending on their authorizations). At the start of a Node, its cache memory is automatically synchronized with the one of the notary nodes, which allows it to run locally without interrogating the notary nodes each time.

Data in cache memory can be modified, deleted, and new data can be added. However, every action corresponds to a new transaction, and appears in the history.

4. Cache path / keys

To create a transaction, in addition to generating useful data and identity data, you have to create a « Cache Path » and a key.

In fact, every transaction is placed in a database which can be seen as a file system. The Cache Path simply corresponds to the path on which we want to store our transaction. The Notary Nodes manage the possible paths. It is not possible to create a transaction with a Cache Path that doesn't exist.

For a same Cache Path, every transaction is identified by its key. In the Cache Memories, keys are unique. This means there can only be one message in the cache memory with the « key » key. If you create a new transaction with the same key, it will correspond to a modification of the cache

memory. If you create a transaction without useful data with that same key, it corresponds to a deletion for the cache memories.

5. Smart contracts

Kalima allows to execute smart contracts. Smart contracts are scripts written in Javascript or Python. Those smart contracts need to be hosted in a repository git. They can be executed in any Java Node (see KalimaJavaExample).