L'API Kalima

Pour bien comprendre le fonctionnement des exemples présents dans ce repository GitHub, il est important de comprendre certaines notions à propos de la Blockchain Kalima.

1. Notary Nodes

Une blockchain Kalima est composée de Notary Nodes et de Nodes ordinaires aussi appelés Nodes clients. Les Notary Nodes sont des Nodes particuliers mais essentiels car c'est avec eux que tout les autres nodes vont directement communiquer. Les Notary Nodes sont responsables de la validation de toutes les transactions dans la Blockchain, ils assurent la traçabilité ainsi que l'intégrité et l'immuabilité de toutes les transactions. On peut installer autant de Notary Nodes que l'on veut pour mettre en place une Blockchain Kalima, avec un minimum de 4 Notary.

2. Nodes clients

Les Nodes clients permettent de se connecter à la blockchain, de synchroniser dans leurs mémoires caches les données auxquels ils sont autorisés, et d'interagir avec ces données et de créer de nouvelles transactions.

Chaque exemple de ce répertoire correspond à des nœuds clients sous plusieurs formes, notamment en Java et pour Android.

3. Mémoire cache / historique

Dans Kalima il faut différencier les données historiques, qui correspondent à toutes les transactions qui ont eu lieu sur la Blockchain depuis sa création, des données en mémoire cache qui sont les données actuelles.

Chaque transaction est automatiquement sauvegardée dans un historique qui est partagé entre tous les Notary Nodes. Cet historique ne peut pas être modifié, aucune donnée ne peut y être supprimée.

En revanche la partie « Mémoire Cache » est partagée entre les Notary Nodes mais également avec tous les autres nodes (en partie, suivant leurs autorisations). Au démarrage d'un Node, sa mémoire cache est automatiquement synchronisée avec celles des notary nodes, ce qui lui permet ensuite de travailler localement, sans interroger les notary à chaque fois.

Les données en mémoire cache peuvent être modifiées, supprimées, on peut en ajouter de nouvelles. Cependant chaque action correspond à une nouvelle transaction, et apparaît donc dans l'historique.

4. Cache path / clés

Pour créer une transaction il faut, en plus des données utiles et des données d'identités, préciser un « Cache path » ainsi qu'une clé.

En fait, chaque transaction est placée dans une base de données qui peut être vue comme un système de fichiers. Le Cache path correspond tout simplement au chemin sur lequel on veut stocker notre transaction. Les Notary régissent les chemins possibles, il n'est pas possible de créer une transaction avec un Cache path qui n'existe pas.

Pour un même Cache Path, les différentes transactions sont identifiées par leurs clés. Au sein des mémoires caches, les clés sont uniques. C'est-à-dire qu'il ne peut y avoir qu'un seul message dans la

mémoire cache avec la clé « key » Si on crée une nouvelle transaction avec la même clé, ça correspond à une modification du point de vue des mémoires caches. Si on crée une transaction sans données utiles avec cette même clé, cela correspond à une suppression du point de vue des mémoires caches.

5. Smarts contracts

Kalima permet d'exécuter des smarts contracts. Les smarts contrats sont des scripts écris en Javascripts ou Python. Ces smarts contracts doivent être hébergés sur un repository git. Ils peuvent être exécutés sur n'importe quel Node Java (cf KalimaJavaExample).