Les Smart Contracts

1. Prérequis

Pour créer un Smart Contract il est nécessaire d'avoir un éditeur de code et/ou un éditeur de texte :

- Exemple d'un éditeur de code : Visual Studio Code, lien de téléchargement : https://code.visualstudio.com/
- Exemple d'un éditeur de code : notepad++, Gedit (Ubuntu).

2. Définition d'un Smart Contrat

Un smart contract, ou contrat intelligent, est un programme informatique qui automatise l'exécution d'un ensemble d'instructions pré-définies lorsque des conditions préalables sont réunies.

Durant son exécution, toutes les étapes de validation sont enregistrées au niveau de la blockchain. Ces enregistrements assurent la sécurisation de l'ensemble des données en empêchant leur modification ou leur suppression.

3. Utilisation des Smart Contracts dans la blockchain Kalima

La Blockchain Kalima utilise des Smart Contracts développés par des outils standards et ouverts comme JavaScript et Python, ces outils ouverts permettent sa simple interconnexion avec d'autres blockchains, d'autres bases de données et d'autres applications.

Dans ces Smart Contrats on peut faire de l'inférence de l'IA grâce à des modèles crées éventuellement à partir des données collectées de préférence par Kalima, car ce sont des données immuables, sûres et complètes.

Dans notre cas les smart contrat sont stockés sur Git mais validés par la blockchain Kalima. La gestion de ces Smart Contracts est intégrée dans l'API Kalima. Pour pouvoir « exécuter des Smart Contracts depuis notre Node, il suffit de fournir les informations de connexion (identifiant et mot de passe) d'un compte autorisé sur le répertoire git où sont stockés les Smart Contracts.

4. Création d'un Smart Contract JavaScript

Remarques :

- Cette partie vous permettra de créer vos propres Smart Contracts et les tester au niveau de la blockchain Kalima.
- Pour utiliser les Smart Contracts que vous allez créer au niveau de la blockchain Kalima, il faut les partager sur Git dans un répertoire propre à vous sur lequel vous serez autorisé.

Un Smart Contract JavaScript est formé principalement des parties suivantes :

Instruction obligatoire :

Pour garantir le fonctionnement de votre Smart Contract JavaScript, il est nécessaire d'ajouter la ligne suivante au début de votre programme :

```
load("nashorn:mozilla_compat.js");
```

Importation des packages :

Cette partie vous permet l'importation des packages Java dont vous aurez besoin de les utiliser au niveau des votre Smart Contract JavaScript.

Pour importer un package Java vous pouvez utiliser la fonction « importPackage() » comme il est montré dans l'exemple suivant :

```
importPackage(Packages.java.util);
importPackage(org.kalima.kalimamq.nodelib);
```

Création des objets Java :

Pour mettre suite à l'importation des packages Java, vous aurez besoin de déclarer les objets Java à utiliser dans votre Smart Contract de la même façon que l'exemple suivant :

```
var JString = Java.type("java.lang.String");
var KMsg = Java.type("org.kalima.cache.lib.KMsg");
```

Les fonctions :

Le but d'utilisation des fonctions est de simplifier un programme pour le rendre plus lisible, pour ne pas avoir à retaper les mêmes lignes de code plusieurs fois dans un même programme, pour éviter les erreurs.

Dans un Smart Contract il est possible de définir une ou plusieurs fonctions dans le but de le découper en petits éléments réutilisable. Ces fonctions permettent aussi de passer des variables au Smart Contrat lors de son exécution.

L'exemple ci-dessous montre l'exemple de la fonction main d'un Smart Contract :

```
function main(logger, kMsg, clone, node) {

	var body = new JString(kMsg.getBody());
	var value = parseInt(body, 10);
	if(value == null)
		return "NOK";

	if(value >= 100) {
		var kMsg1 = new KMsg(0);
		node.sendToNotaryNodes(kMsg1.getMessage(node.getDevID(),
KMessage.PUB, "/alarms/fire", kMsg.getKey(), kMsg.getBody() , new
KProps("-1")));
	}

	return "OK";
}
```

5. Exemples des Smart Contracts

Remarque :

- Pour organiser les données dans la blockchain Kalima, nous utilisons des cache paths. Cela peut être vue comme un système de fichiers avec des dossiers et des fichiers qui vont contenir les données.

Il existe des exemples des Smart Contracts partagé sur git. Ces Smart Contrats sont appelés depuis l'exemple Java du client de la blockchain Kalima et ils permettent d'exécuter des tâches précises à chaque fois que les conditions sont remplies.

Le Smart Contract « sensor.js » :

Le fonctionnement de ce Smart Contract repose sur la partie suivante :

```
if(value == null)
            return "NOK";

      if(value >= 100) {
            var kMsg1 = new KMsg(0);
            node.sendToNotaryNodes(kMsg1.getMessage(node.getDevID(),
KMessage.PUB, "/alarms/fire", kMsg.getKey(), kMsg.getBody() , new
KProps("-1")));
      }
```

Ce Smart Contract effectue un test sur les valeurs de nouvelles données ajoutées dans le cache path « /sensor ». A chaque fois qu'une valeur dépasse le seuil fixé (par exemple un seuil de 100), il écrit une alarme dans le cache path « /alarms/fire ».

Le Smart Contract « fire.js » :

Le fonctionnement de ce Smart Contract repose sur la partie ci-dessous :

```
for(i=body.length ; i>=0; i--) {
            reverseString += body.charAt(i);
      }
```

Le Smart Contract « fire.js » est exécuté à chaque fois qu'il y a une nouvelle donnée à ajouter dans le cache path « /alarms/fire ». Il permet d'inverser la valeur de cette nouvelle donnée.

6. Sécurisation des Smart Contracts

Il faut noter que l'exécution des Smart Contracts est sécurisée par la blockchain Kalima : uniquement les Smart Contracts validés seront lancés par la blockchain Kalima.

7. Exécution des Smart Contracts

L'exécution des Smart Contracts au niveau de la blockchain Kalima s'effectue depuis un nœud Java. Pour savoir comment créer le nœud Java et/ou comment exécuter ces Smart Contracts il nécessaire de consulter la documentation « JavaExemple.pdf » partagée dans le chemin d'accès suivant : https://github.com/Kalima-Systems/Kalima-Tuto/blob/master/etc/doc/fr/JavaExample.pdf