smartDirectory (mission smart2)

Document de liaison équipe en date du 8 nov. 2024

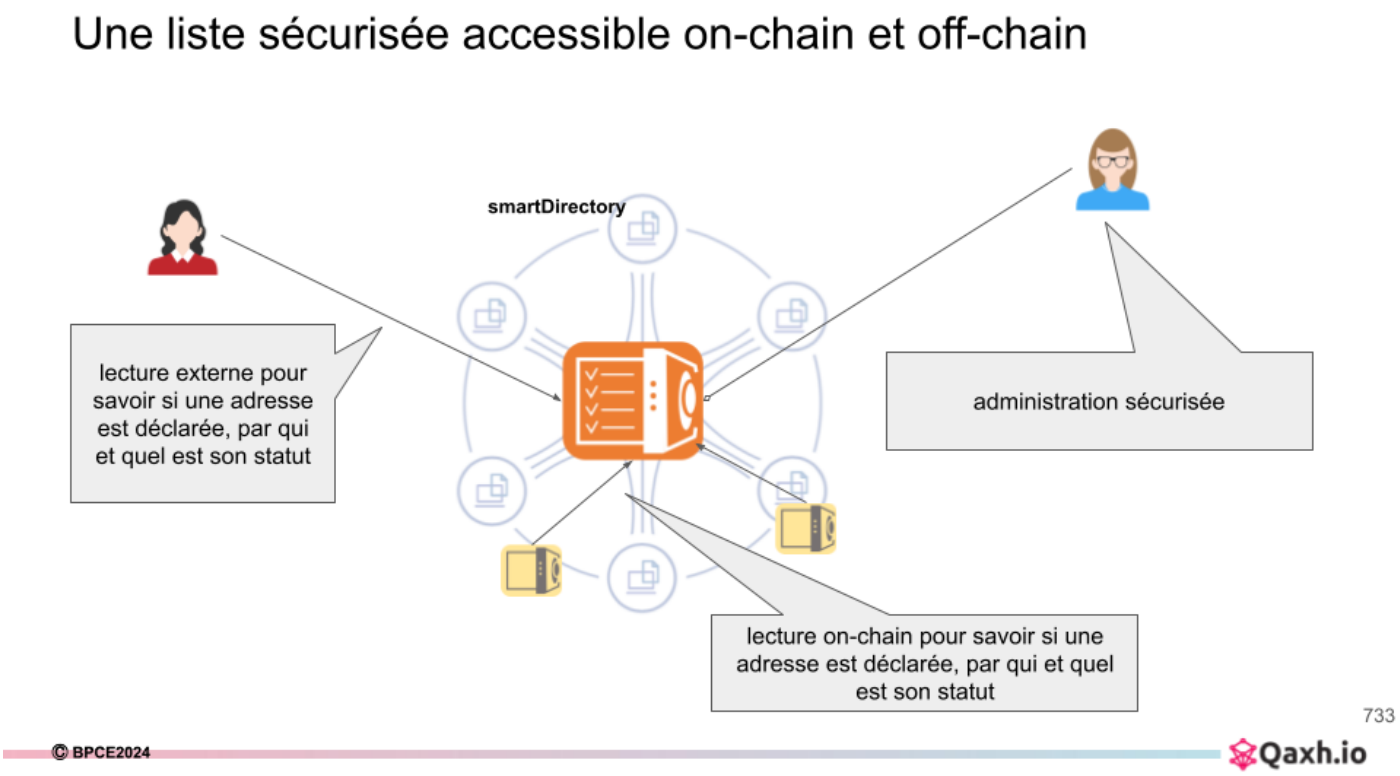
Livrables envisageables

Juillet 2024	Proposition du smartDirectory
	Spécifications générales du smartDirectory
	Encapsulation des fonctions blockchain pour l'app smartphone
	Mise à disposition des API sur GitLab
	Spécification de la démo
	Proposition de l'interface smartphone
	smartDirectory avec test python
	fonction spécifique smartDirectory pour App smartphone
	Application smartphone

Rappel des principes initiaux

Voir le détail complet de la proposition

Avec l'explosion anticipée du nombre de smartContracts due à la tokenisation de l'économie et l'account abstraction, nous pensons que la première étape doit être la facilité de création de listes de références pour des accès privatifs ou publics, hors de la blockchain ou en interne de la blockchain.



En synthèse, nous identifions les besoins génériques suivants :

- une facilité de déploiement de nouveaux smartContracts,
- une capacité de contrôle de la validité de ces smartContracts par un utilisateur externe à la blockchain,
- une capacité de contrôle de la validité de ces smartContracts par un autre smartContract,
- une nécessité d'identifier des écosystèmes de smartContracts pour en faire une analyse,
- une capacité de maintenir à jour ces écosystèmes.

Cas d’usages

Les cas d’usages du smartDirectory

- usage privatif (ou de supervision) :
  - l’administrateur d’une entité (le déclarant) liste des smartContracts en y affectant un projectID
  - ceci permet à une application pour des utilisateurs d’avoir toujours la liste des smartContracts à jour en ne connaissant que la seule adresse du smartDirectory et le projectID
  - Ceci peut se faire sur des smartContracts déployés ou non par le déclarant
- usage déclaratif par un déployeur
  - une entité, le déclarant, qui émet des smartContracts les référence lors du déploiement
  - l’entité peut aussi mettre à jour l’URI présente dans la structure de liste des déclarants
- usage de régulation
  - un régulateur peut imposer à une entité régulée de déclarer tous les smartContracts qu’elle émet (comme dans le cas de l’usage déclaratif)
  - Au préalable le régulateur enregistre l’adresse de l’entité
  - de plus il peut imposer que l’entité expose une API permettant l’accès à plus d’information. Il fait inscrire par le déclarant l’accès à cette API

738

Qaxh.io

Présentation du smartDirectory

Le smartDirectory est une évolution dérivée du schemeDirectory reprenant ses principales fonctions mais dans une optique sans hatchSafe.
L’objet du smartDirectory est de pouvoir bénéficier de tous les apports du schemeDirectory sans pour autant rendre obligatoire la création d’un hatchSafe.

Définitions et nommage

- **smartDirectoryAddress** : adresse du smart contract “smartDirectory”
- **déclarant (“registrant”)** : acteur externe identifié par son adresse et à même d’enregistrer des ”references”
- **Reference:** Adresse de smart contract déclarée, c’est adresse du smartContract qui est conservé dans la liste des references du smartDirectory
- code projet (“project_id”)
- adresse du déclarant (registrant_address)

Les structures du smartDirectory

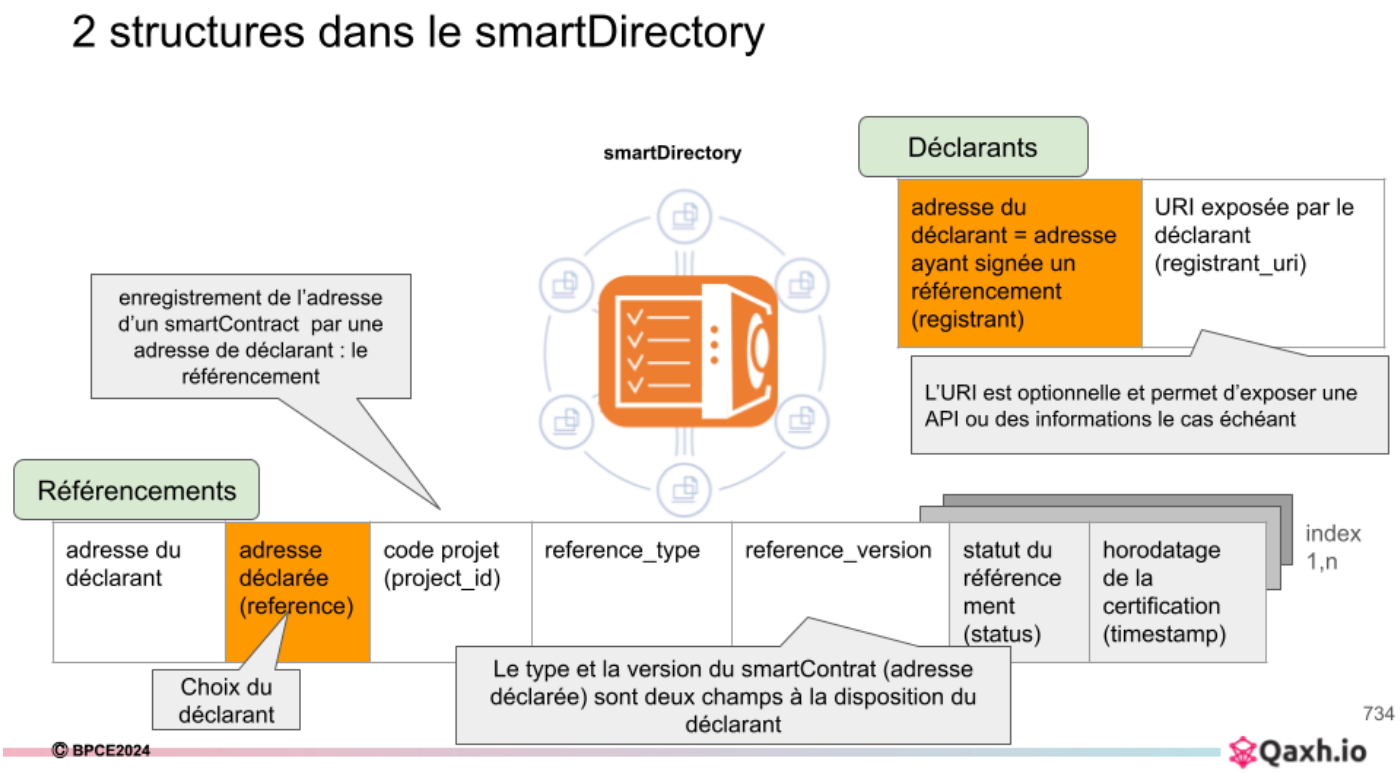
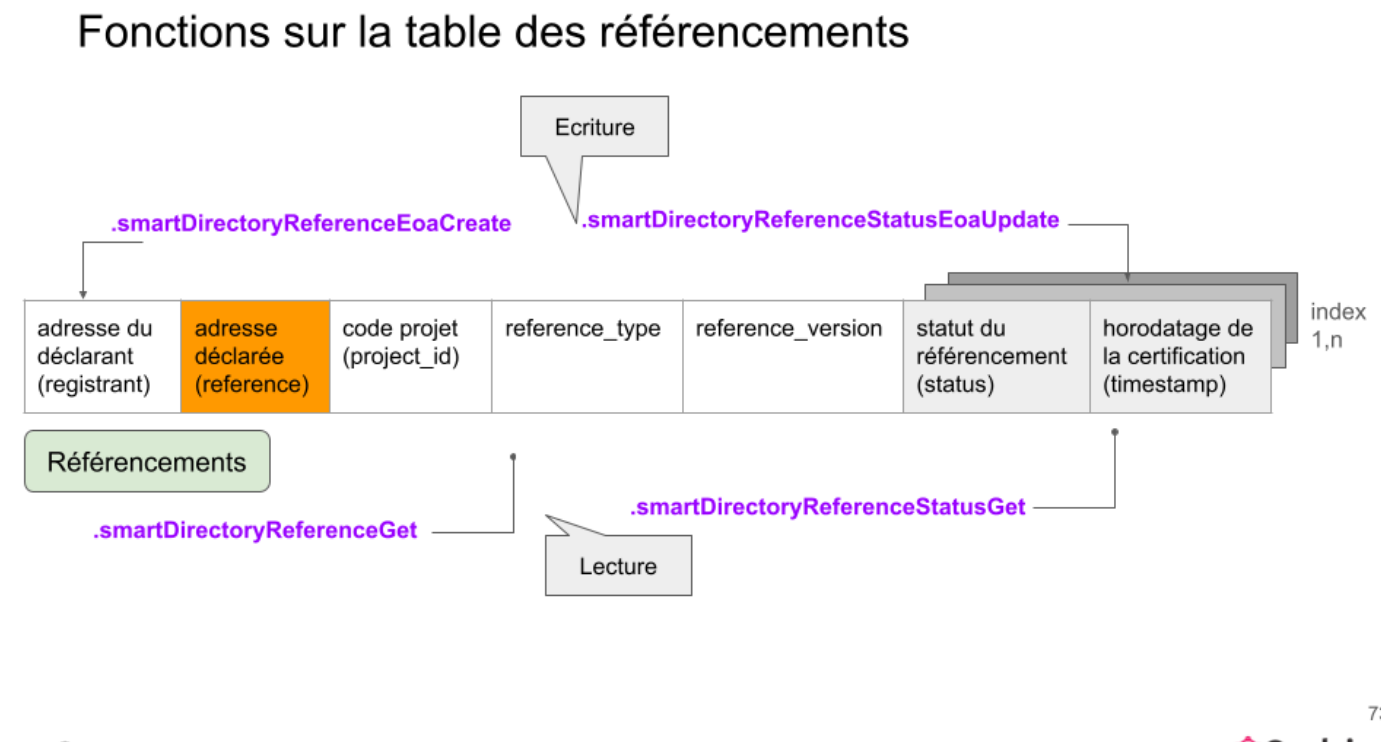


Table des référencements

Cette table contient tous les référencements sur la base du “record” suivant :

- adresse du déclarant (EOA ou smartContract), utilisation possible comme index de lecture de la table.
- adresse du smartContract déclaré (référence). Cette adresse peut aussi être une EOA. index de lecture
- project_id, qui est une chaîne de caractères représentant le code projet.
- Variables complémentaires associées à ce record écrites une fois et non modifiables :
  - contract_type : chaîne de caractères
  - contract_version : chaîne de caractères
- sous-table des statuts :
  - statut du référencement (status) : champ string, modifiable uniquement par le déclarant. La sémantique de ce statut est à la main du déclarant, une signification explicite est conseillée (par ex: en production, en suspens, abandonné, etc…) ou une URI d’explication
  - horodatage (timestamp) de l’écriture lors de la la création ou du changement de statut,
  - Les statuts sont une liste qui peut être parcourue (premier index: 1)



.smartDirectoryReferenceEoaCreate (smartDirectoryAddress, reference, project_id, contract_version, contract_type, status)

Cette fonction permet la création d'un nouvel enregistrement dans la table des référencements.

- L'adresse du déclarant n'est pas explicitement dans les paramètres car c'est l'adresse qui signe la transaction de création
- l'horodatage (timestamp) est mis automatiquement lors de la création du record
- le statut (status) est une chaîne de caractère libre qui peut être sous forme d'un "URI"
- Valeur de retour: retour un tx_hash pour vérifier le minage côté client.

S'il existe déjà un record de l'adresse déclarée (référence) par ce déclarant alors faire un "revert".

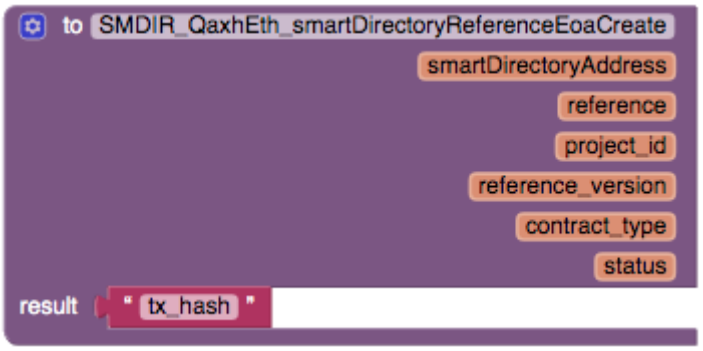
2 Stratégie d'utilisation suivant le paramétrage du smartDirectory au moment de sa création :
Enregistrement préalable des déclarants[43] (mint_code = 0):

- Si le déclarant n'est pas présent -> rejet
- Si le déclarant est bien présent dans la table des déclarants -> ajout de la reference

Enregistrement au simultané du déclarant (mint_code = 1)

- Si le déclarant n'est pas présent -> ajout dans la table des déclarants et ajout de la référence
- Si le déclarant est présent -> ajout de la reference

[43] La déclaration est réservée aux adresses Parents




```
 @SimpleFunction(description = "create a new smartContract reference in the SmartDirectory")
   public String smartDirectoryReferenceEoaCreate (String smartDirectoryAddress, String
referenceAddress, String projectId,
                                        String referenceType, String referenceVersion, int status) {

      SmartDirectory folderContract = SmartDirectory.load(smartDirectoryAddress, web3,
transactionManager,
            new DefaultGasProvider());

      String tx_hash;
      try {
         tx_hash = doTransaction(
               folderContract.addReference(referenceAddress, projectId, referenceType, referenceVersion,
                     new BigInteger(status)),
               "addrReference");
      } catch (Exception e) {
         String message = "Error smartDirectoryReferenceEoaCreate: " + e.getMessage();
         android.util.Log.d(LOG_TAG, message);
         tx_hash = message;
      }
      return tx_hash;
   }
```

.smartDirectoryReferenceGet (smartDirectoryAddress, reference)

Le principe de gestion de confiance d'un smartDirectory peut constituer à inscrire dans une APP l'adresse du déclarant et l'adresse du smartDIrectory à qui l'on peut faire confiance et de tester si le smartContract est bien créé par ce déclarant.

Cette fonction permet la lecture d'un "record" en ne connaissant uniquement l'adresse du smartContract référencé. C'est la fonction principale d'utilisation du smartDirectory.

- en retour idéalement dans un dictionnaire :
  - adresse du déclarant
  - project_id
  - reference_type
  - reference_version
  - dernier status
  - dernier timestamp
  - dernier index de statut

- En cas d'inexistence la fonction retourne un dictionnaire vide

La fonction appelle les fonctions “getReference” et “getReferenceLastStatusIndex” du smartContract “SmartDirectory.sol” :

```
@SimpleFunction(description = "get details of a smartContract reference")
public Object smartDirectoryReferenceGet(String smartDirectoryAddress, String reference) {

    SmartDirectory folderContract = SmartDirectory.load(smartDirectoryAddress, web3,
transactionManager,
            new DefaultGasProvider());

    SmartDirectory.Reference result_raw;
    List<Map<String, String>> results = new ArrayList<>();

    try {
        result_raw = folderContract.getReference(referenceAddress).send() &
folderContract.getReferenceLastStatusIndex(referenceAddress).send();

        Map<String, String> result_dict = new HashMap<>();

        result_dict.put("registrantAddress", result_raw.registrantAddress.toString());
        result_dict.put("referenceAddress", result_raw.referenceAddress.toString());
        result_dict.put("projectId", result_raw.projectId.toString());
        result_dict.put("referenceType", result_raw.referenceType.toString());
        result_dict.put("referenceVersion", result_raw.referenceVersion.toString());
        result_dict.put("lastStatus", result_raw.status.toString());
        result_dict.put("lastTimestamp", result_raw.timestamp.toString());
        result_dict.put("lastStatusIndex", result_raw.lastStatusIndex.toString());

        results.add(result_dict);
    } catch (Exception e) {
        Map<String, String> result_dict = new HashMap<>();
        result_dict.put("Error smartDirectoryReferenceGet", e.getMessage());
        results.add(result_dict);
    }
    return results;
}
```

.smartDirectoryReferenceStatusEoaUpdate (smartDirectoryAddress, reference, project_id, status)

Cette fonction permet de changer le statut du record identifié par le triplet (smartDirectory, project_id, smartContract)

- Si le record n'existe pas faire un "revert".
- Si 1) le record existe 2) la transaction d'update est signée par l'adresse du déclarant présente dans le record (vérification à faire en solidity) alors une nouvel enregistrement est faite dans la sous-table des statuts avec le nouveau statut et l'horodatage associé

```
@SimpleFunction(description = "update status of a reference")
   public int smartDirectoryReferenceStatusEoaUpdate (String smartDirectoryAddress, String
referenceAddress, String projectId,
                                        String status) {

      SmartDirectory folderContract = SmartDirectory.load(smartDirectoryAddress, web3,
transactionManager,
            new DefaultGasProvider());

      String tx_hash;
      try {
         tx_hash = doTransaction(
               folderContract.updateReferenceStatus(referenceAddress, projectId, new BigInteger(status)),
               "updateReferenceStatus");
      } catch (Exception e) {
         String message = "Error smartDirectoryReferenceStatusEoaUpdate: " + e.getMessage();
         android.util.Log.d(LOG_TAG, message);
         tx_hash = message;
      }
      return tx_hash;

   }
```

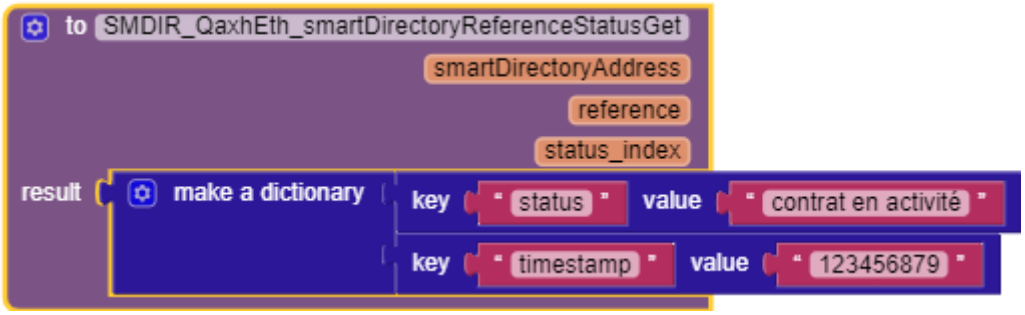
.smartDirectoryReferenceStatusGet (smartDirectoryAddress, reference, status_index)

Cette fonction permet la lecture d'un index des statuts. En effet, la fonction .smartDirectoryReferenceGet renvoie le triplet (dernier statut, dernier timestamp, dernier index) ce qui devrait subvenir à la plupart des besoins.

Cependant, en ayant le dernier index, il devient facile de faire une boucle dans la requête d'interrogation pour obtenir si besoin l'historique des statuts.

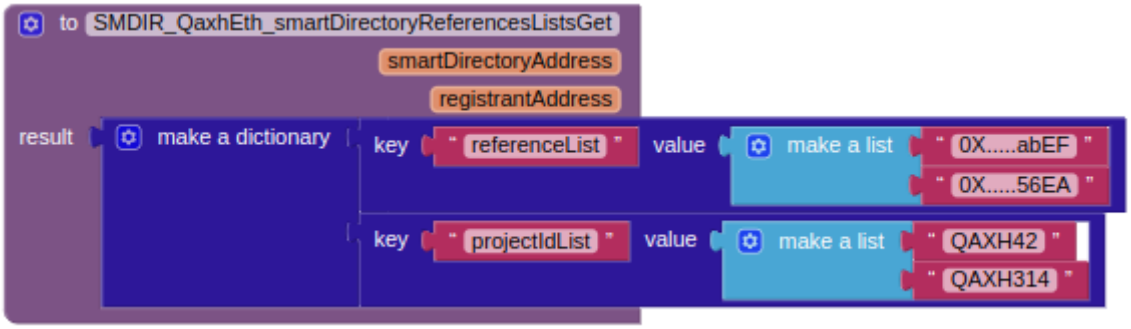
en retour cette fonction donne :

- le statut à l'index de la requête
- le timestamp à l'index de la requête



.smartDirectoryReferencesListsGet (smartDirectoryAddress, registrantAddress)

Cette fonction donne les deux listes (liste des adresses déclarées, liste des codes projets) pour une adresse de déclarant donnée.



⚠ Comme à chaque adresse déclarée (reference) correspond un code projet (project_id), ces deux listes ont la même taille et sont ordonnées pour que l'index de la liste des adresses déclarées corresponde à l'index du code projet.

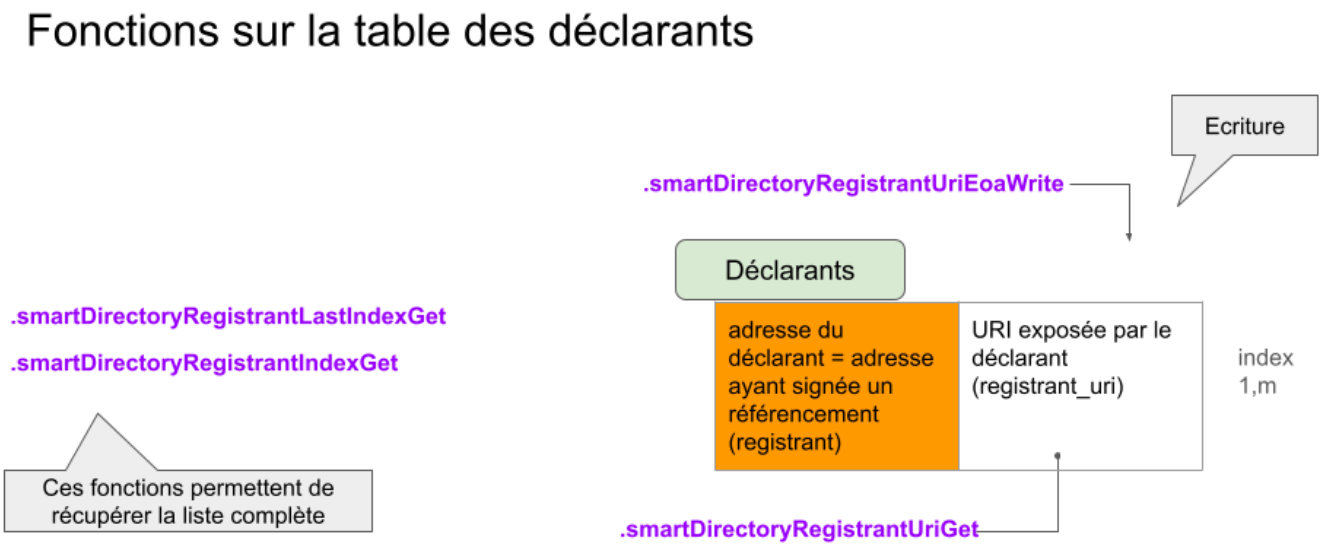
```
@SimpleFunction(description = "return a list of reference/projectId for a given registrant address")
    public Object smartDirectoryReferencesListsGet(String smartDirectoryAddress, String
registrantAddress) {

        SmartDirectory folderContract = SmartDirectory.load(smartDirectoryAddress, web3,
transactionManager,
                new DefaultGasProvider());

        SmartDirectory result_raw;
        List<Map<String, String>> results = new ArrayList<>();

        try {
            result_raw = folderContract.getReferencesLists(registrantAddress).send();
            Map<String, String> result_dict = new HashMap<>();

            result_dict.put("referenceAddresses", result_raw.referenceAddresses.toString());
```

```
            result_dict.put("projectIDs", result_raw.projectIds.toString());

            results.add(result_dict);
        } catch (Exception e) {
            Map<String, String> result_dict = new HashMap<>();
            result_dict.put("error smartDirectoryReferencesListsGet", e.getMessage());
            results.add(result_dict);
        }
        return results;
    }
```

.smartDirectoryReferencesCountGet (smartDirectoryAddress, registrantAddress)

Cette fonction donne la taille de la liste précédente. Cela peut être utile pour modifier l'UX de l'utilisateur en cas de taille importante.

Table des déclarants (registrants Table)



Une table des déclarants est créée et mise à jour soit explicitement pour chaque nouveau declarant soit le cas échéant à chaque nouvelle création de record dans la table des référencement, ceci en fonction de la strategie d'utilisation (voir la fonction **.smartDirectoryReferenceEoaCreate)**

Le paramétrage est donné par le "mint_code" du smartDirctory :

- "mint_code" = 0 : les adresses des déclarants doivent être au préalables enregistrées par un des parents

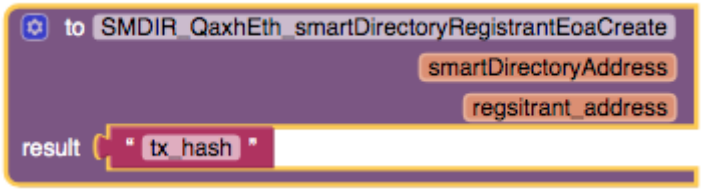
- en particulier, si les parents veulent être déclarants, ils doivent s'auto-déclarés dans la table
- "mint_code" = 1 : toutes les adresses se déclare automatiquement lors de la première déclaration

Cette table est constitué de 2 éléments :

1. l'adresse d'un déclarant
2. une chaîne de caractère à la disposition du déclarant afin d'y déposer une URI d'information.. Cette chaîne de caractère est mise à jour dans un deuxième temps par le déclarant.

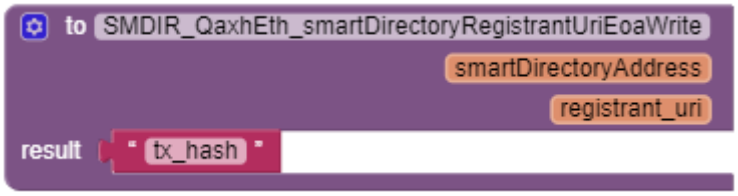
.smartDirectoryRegistrantEoaCreate (smartDirectoryAddress, registrant_address)

Cette fonction permet la création d'un nouveau déclarant par une adresse parent. Cette fonction n'est utilisable que pour le mint_code =0. En effet, dans le mint_code = 1, c'est l'auto-déclaration qui est privilégiée.



.smartDirectoryRegistrantUriEoaWrite (smartDirectoryAddress, registrant_uri)

Cette fonction permet la mise à jour de la chaîne de caractère de la table des déclarants. La transaction doit être signée par le déclarant (donc l'adresse du déclarant n'est pas dans les paramètres).
Sinon, faire un revert.
Si l'adresse de la transaction n'est pas dans la table, faire un revert.




```
@SimpleFunction(description = "update uri associated to a registrant")
  public String smartDirectoryRegistrantUriEoaWrite(String smartDirectoryAddress, String registrantUri) {

      SmartDirectory folderContract = SmartDirectory.load(smartDirectoryAddress, web3,
transactionManager,
            new DefaultGasProvider());

      String tx_hash;
      try {
         tx_hash = doTransaction(
```

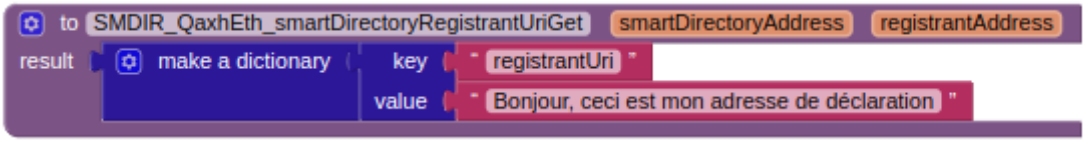
```
                folderContract.updateRegistrantUri(registrantUri),
                "updateRegistrantUri");
        } catch (Exception e) {
            String message = "Error smartDirectoryRegistrantUriEoaWrite: " + e.getMessage();
            android.util.Log.d(LOG_TAG, message);
            tx_hash = message;
        }
        return tx_hash;

    }
```

.smartDirectoryRegistrantUriGet (smartDirectoryAddress, registrantAddress)

Cette fonction renvoie l'URI de la table des déclarants pour un déclarant donné.

Dans la réponse, idéalement un dictionnaire :

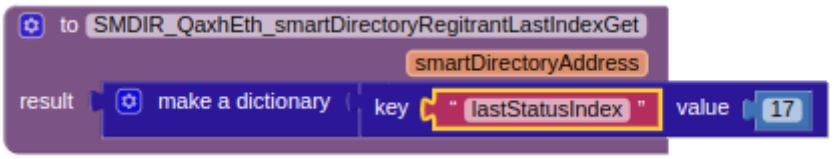
- dictionnaire vide si l'adresse demandée n'existe pas dans la table
- {"registrant_uri" : "<string>"} si l'adresse est dans la table




```

```

.smartDirectoryRegitrantLastIndexGet (smartDirectoryAddress)

Cette fonction permet de connaître le dernier index de la liste des déclarants.




```
@SimpleFunction(description = "")
    public int smartDirectoryRegistrantLastIndexGet(String smartDirectoryAddress) {
        SmartDirectory folderContract = SmartDirectory.load(smartDirectoryAddress, web3,
transactionManager, new DefaultGasProvider());

        BigInteger result;
        int resultInt;

        try {
            result = folderContract.getRegistrantLastIndex().send();
            resultInt = result.intValue();
```