nomadic labs

Les Tickets sur Tezos

Sommaire

Introduction

Introduction

L'amendement Edo, qui a été voté par la communautés en février 2021 contient une amélioration du langage Michelson sous la forme d'un nouveau type de données : les **tickets**.

Les tickets ont la particularité de contenir de façon immuable l'adresse du contrat qui les a créés et de ne pas être duplicables, ce qui offre la garantie à un contrat qu'il n'y aura pas plus de tickets portant son adresse que ceux qu'il a lui même créés.

Un ticket est composé de trois éléments :

1. **une origine** : l'adresse du contrat qui l'a créé
2. **un identifiant** : un élément de n'importe quel type comparable. Celui-ci peut simplement servir à gérer plusieurs types de jetons ou à contenir une information.
3. **une valeur** : cette valeur est équivalente à une quantité qui caractérise des jetons fongibles ou non-fongibles. Dans le cas de jetons fongibles, cette valeur est supérieure à 1 mais divisible en unité. Dans le cas de jetons non fongibles, cette valeur est égale à 1.

Les cas d'usages sont nombreux, en particulier les tickets permettent de simplifier les contrats de tokenisation en offrant une matérialisation des jetons.
Ils permettent également de garantir simplement des droits d'accès qui peuvent être cédés à l'extérieur du contrat, c'est à dire sans l'appeler et donc sans changer son état.

Instructions

Instructions

Il existe 4 instructions Michelson permettant d'interagir avec des tickets :

- TICKET : permet de créer un ticket à partir d'un identifiant et d'une valeur, l'adresse est celle du contrat.
- READ_TICKET : indique les éléments du ticket (adresse, identifiant et valeur) sans le détruire [1].
- JOIN_TICKETS : assemble deux tickets de même origine et de même identifiant en un nouveau ticket, sa valeur sera la somme des valeurs des tickets initiaux.
- SPLIT_TICKET : inverse de l'opération JOIN_TICKETS, découpe un ticket en deux tickets de même identifiant et tel que la somme de leurs valeurs est égale à la valeur du ticket initial.

1. La plupart des fonctions Michelson détruisent les éléments qu'elles utilisent. Une fois la transaction appliquée, les arguments de cette fonction ne sont plus sur la pile de sorte que si on veut les réutiliser, il faut préalablement les dupliquer (avec l'instruction DUP).
READ_TICKET fait exception à cette règle car il est impossible de dupliquer un ticket.

Identifiant

Identifiant

Le champ que nous avons appelé identifiant ne sert pas uniquement à identifier un ticket, il peut également permettre de stocker des informations qui pourront être utilisées par les contrats qui liront le ticket.

On pourra par exemple :

- Gérer un nombre arbitraire de tickets différents sur un même contrat en faisant varier l'identifiant,
- faire vérifier une donnée par un contrat à la manière d'une signature,
- donner une date limite d'utilisation, ou au contraire une date d'activation au moyen d'un timestamp.

Grâce à l'extension des types comparables [1], il est possible de cumuler ou combiner ces usages. Il est important de considérer que le type dépend de l'usage qui sera fait du ticket et que deux tickets de même type n'auront pas forcément les mêmes usages selon les contrats dans lesquels ils seront utilisé.

1. Depuis le protocole Edo, les paires de deux types comparables, des types optionnels comparables et les unions de deux types comparables sont des types comparables

Cas d'usage

Tokenisation

Les tickets permettent de matérialiser les jetons d'un contrat de tokenisation. Ainsi, ils peuvent être manipulés en dehors du contrat d'origine, ce qui permet de les échanger contre des tickets provenant de différents contrats.

Dans certains cas, les jetons peuvent être intégralement stockés sous forme de tickets et circuler en dehors du contrat. Ceci fonctionne pour des actifs fongibles ou non fongibles.
C'est notamment le cas lorsque les jetons servent à faire des ``delivery versus payment'', et dans lesquels il n'est pas nécessaire de savoir qui détient des jetons jusqu'à ce qu'ils soient renvoyés au contrat pour les détruire et faire valoir ce que de droit.

D'autres contrats seront hybrides. C'est le cas des contrats représentant des items de jeux vidéo : étant donné qu'il est nécessaire d'être capable de savoir à tout moment si un joueur détient un item, il faut que le contrat garde l'information dans ses propres données.
Il est cependant possible de permettre à un joueur de retirer temporairement un de ses items, par exemple pour les échanger, ou pour prouver à un autre contrat qu'il le possède.
Il est cependant possible de permettre à un joueur de retirer temporairement un de ses items, par exemple pour l'échanger, ou pour prouver à un autre contrat qu'il le possède.

Gestion de privilèges

Les tickets permettent également d'accorder un privilège d'utilisation à un utilisateur. On peut ainsi garantir que cet utilisateur est le seul à pouvoir utiliser certaines propriétés. Excepté si celui-ci décide de céder son jeton, on peut alors considérer qu'en donnant (ou en vendant) le ticket, il vend également le contrat et les privilèges associés. Ceci peut faciliter les cas d'échange ou de vente de contrats.

La gestion de privilèges par ticket permet aussi à l'utilisateur de choisir la façon dont il conserve son ticket. Il peut par exemple utiliser un contrat de multi-signatures pour que le contrat soit en réalité contrôlé par plusieurs utilisateurs.

On peut également envisager que ce contrat possède les tickets d'accès de plusieurs contrats.

Il est par conséquent possible de céder ou d'acquérir les droit d'accès d'un contrat simplement, et tel que le nouvel utilisateur puisse choisir la façon dont il souhaite conserver son ticket.

Votes

Les tickets peuvent également permettre de réaliser des votes pour prendre des décisions ou pour valider des transactions.

Un contrat pourrait assurer que des transferts se font suivant les votes d'un comité : à chaque fois qu'une transaction est proposée, les membres votent avec leurs tickets.

Il devient donc possible de considérer des cas de votes proportionnels dans lesquels des participants n'ont pas le même poids (en fonction du nombre de tickets possédés).

Les membres peuvent alors vendre tout ou une partie de leurs droits, ou au contraire acheter plus de parts.

Il est également possible dans ce contexte d'organiser des votes avec une finalité extérieure à la blockchain, comme des élections ou des décisions d'entreprises.

Authentification

L'authentification par tickets consiste à permettre à chaque utilisateur de choisir sa méthode de protection, il lui suffit d'enregistrer l'adresse du contrat qui lui sert à générer les tickets authentifications et de valider les transactions le concernant à condition qu'elles soient accompagnées d'un ticket de ce contrat comme si celui-ci était une signature.

Le principal avantage de ce système, est que le contrat pourra avoir sa propre gestion de l'émission de tickets et pourra être par exemple : Un multi-sig, un contrat temporisé ou un contrat manager. De plus un même contrat pourra être utilisé pour générer les tickets d'authentifications de plusieurs autres contrats. Il sera également possible de faire un contrat adapté qui n'associe pas la même sécurité aux différentes fonctionnalités.

Enfin, le principal avantage de cette méthode réside dans sa non-réplicabilité, en effet, si on réclame un ticket de valeur 1 et que ce ticket est consommé lors de l'application, ce ticket ne pourra pas être réutilisé. Ceci permet d'émettre plusieurs tickets sans se soucier de l'ordre dans lequel ils seront exécutés.

Il est également possible pour le contrat authentificateur de créer un ticket de valeur supérieure à 1, cette valeur fixera le nombre maximum d'utilisation de cette authentification.

nomadic labs

Continuons cet échange

https://tezos.com
https://developers.tezos.com