nomadic labs

Tickets on Tezos

nomadic labs

Contents

nomadic labs

Introduction

Introduction

The Edo Amendment, which was voted by the community in February 2021 contains an improvement to the Michelson language in the form of a new data type: **tickets**.

The specific feature of tickets is that they contain the address of the contract that created them, which is fixed and cannot be duplicated; this ensures that a contract will not have more tickets bearing its address than those that it has itself created.

A ticket comprises three elements:

1. **an source**: the address of the contract that created it
2. **an identifier**: an element of any comparable type. This can simply be used to manage several types of tokens or to contain information
3. **a value**: this value is equivalent to a quantity that distinguishes fungible or non-fungible tokens. In the case of fungible tokens, this value is greater than 1 but divisible into units. In the case of non-fungible tokens, this value is equal to 1.

There are many use cases; in particular, tickets help to simplify tokenization contracts by providing token materialization.
They also ensure that access rights can be transferred simply outside the contract, i.e., without calling it and so without changing its status.

nomadic labs

Instructions

Instructions

There are four Michelson instructions that enable interaction with tickets:

- TICKET: allows a ticket to be created from an identifier and a value; the address is that of the contract.
- READ_TICKET: indicates the elements of the ticket (address, identifier, and value) without destroying it.[1]
- JOIN_TICKETS: combines two tickets with the same source and identifier into a new ticket; its value will be the sum of the values of the original tickets.
- SPLIT_TICKET: reverse of the JOIN_TICKETS operation; splits one ticket into two tickets with the same identifier such that the sum of their values is equal to the value of the original ticket.

[1] The majority of Michelson functions destroy the elements that they use. Once the transaction is applied, the arguments of this function are only still in the stack if you want to reuse them; you must first duplicate them (with the DUP instruction). READ_TICKET is the exception to this rule as it is not possible to duplicate a ticket.

nomadic labs

Identifiers

Identifiers

The field that we have called identifier is not only used to identify a ticket; it can also store information that may be used by the contracts that will read the ticket.

For example, we can :

- manage an arbitrary number of different tickets on the same contract by changing the identifier;
- have a piece of data verified by a contract in the same way as a signature;
- give a piece of data a use-by date or, alternatively, an activation date by means of a timestamp.

By extending comparable types,[1] it is possible to aggregate or combine these uses.
It is important to consider that the type depends on the use that will be made of the ticket and that two tickets of the same type will not necessarily have the same uses depending on the contracts in which they will be used.

[1] From the Edo protocol, comparable types are pairs of two comparable types, optional comparable types and unions of two comparable types.

nomadic labs

Use cases

Tokenisation

Tickets enable the tokens of a tokenization contract to be materialized. In this way, they can be manipulated outside the original contract; this allows them to be exchanged against tickets coming from different contracts.

In some cases, tokens may be stored entirely in the form of tickets and circulate outside the contract. This works for fungible or non-fungible assets. This is particularly the case when tokens are used for delivery versus payment, and where it is not necessary to know who holds the tokens until they are sent back to the contract to be destroyed, and for all legal intents and purposes.

Other contracts will be hybrid. This is the case for contracts representing video game items: Given that it is necessary to be able to know at any time whether a player holds an item, the contract must keep the information within its own data. **However, it is possible to allow a player to withdraw one of their items temporarily, for example to exchange them or prove to another contract that they have it.**

Privilege management

Tickets also enable a usage privilege to be granted to a user. We can thus ensure that this user is the only one able to use certain properties. Unless this user decides to transfer their token, we can therefore consider that by giving them (or selling them) the ticket, they are also selling the contract and associated privileges. This can facilitate cases where contracts are exchanged or sold.

Privilege management by ticket also enables the user to choose the way in which they retain their ticket. For example, they can use a multi-signature contract such that the contract is in effect controlled by several users.

We can also consider that this contract has tickets to access several contracts.

It is therefore possible to transfer or acquire rights of access to a contract simply and such that the new user can choose the way in which they wish to retain their ticket.

Votes

Tickets can also allow voting to take decisions or to validate transactions.

A contract could ensure that transfers are made according to the votes of a committee: Each time a transaction is proposed, members vote with their tickets.

It therefore becomes possible to consider cases of proportional voting in which participants do not have the same weighting (according to the number of tickets held).

Members can then sell all or some of their rights or, on the other hand, buy more units.

It is also possible in this context to organize voting for a purpose external to the blockchain, such as elections or business decisions.

Authentication

Authentication by tickets entails allowing each user to choose their protection method. They just need to register the contract address, which they use to generate authentication tickets and validate transactions that concern them, provided that they are accompanied by a ticket from this contract as if this were a signature.

The main advantage of this system is that the contract may have its own ticket-issue management and may, for example, be a multi-signature wallet, a time-based contract or a manager contract. Furthermore, the same contract may be used to generate the authentication tickets of several other contracts. It will also be possible to make a suitable contract that does not associate the same security with different functionalities.

Lastly, the main advantage of this method lies in its non-replicability. Indeed, if we request a ticket with a value of 1 and this ticket is used at the time of application, this ticket cannot be reused. This enables several tickets to be issued without having to worry about the order in which they will be executed.

It is also possible for the authenticating contract to create a ticket with a value of greater than 1. This value will set the maximum number of uses of this authentication.

nomadic labs

Let's keep the conversation going

https://tezos.com
https://developers.tezos.com
https://tezos-baking.slack.com