Tokenization Without Tokens

How to implement asset ownership in a Blockchain using access control.

Alberto Cuesta Cañada

Follow

HackerNoon.com

--

Listen

Share

The only constant in the technology industry is change. — Marc Benioff

Introduction

There is the general idea that to represent assets in a blockchain you should use the ERC721 token. As Chief Architect for TechHQ I've designed a number of solutions where physical assets are represented in a blockchain, and I've found that in a supply chain scenario the ERC721 is not great.

A Directed Acyclic Graph allows to allows to track the asset lifecycle more naturally, and in a previous article we described in detail how to implement it.

In a nutshell each asset is represented by a series of states with each state recording an event in the life of the asset. States can point to past states and states can't be modified once created.

As a result of ditching the ERC721 token we have to come up with a way to represent asset ownership and transfer.

If you think about it, the transfer

and transferFrom

methods from any ERC token are a form of access control. A transfer

call is just an update on the data structures that record token ownership, which only a specific user can execute. A transferFrom

call is a delegation of permissions to do the same.

We will have to implement something equivalent to transfer

, but for nodes in a graph. We will have to code access controls for appending data to our graph structure.

To represent asset ownership and transfer without a token we are deconstructing and then rebuilding one of the core features of the ethereum ecosystem. Please follow me into the rabbit hole.

Conceptual Solution

In our supply chain as a graph each asset is represented as a series of states linked between them. As that asset goes through its lifecycle we append more states, signifying events such as transformation, composition, destruction, and so on.

Only the owner of an asset should be able to append new states to its lifecycle. It should be possible the owner to give this right to others for as long as he wishes. He should also be able to lose the right of appending states to an asset in favour of someone else.

This restriction on appending states represents ownership of each asset. Ownership of an asset is controlling access to it.

A real world scenario can help to visualise our use case more clearly.

- Let's say I own a manufacturing plant that receives materials, transforms them into gadgets, and sells them to a distributor.
- As the materials arrive into my plant I am given ownership of them which I delegate into my plant operators.

- My operators transform the materials into gadgets recording the steps in the supply chain graph.
- I will then hand over the gadgets to the distributor along with the permission to append states to them.

We are going now to implement these features using theRole Based Access Control library which we published recently and was featured in the front page of this medium.

Implementation

In the original implementation of a supply chain as a graph we recorded three data items for each state.

- The account that created the state;
- the asset that the state refers to
- and the precedent states.

For the implementation of the ownership concept I'm going to add two variables in each state:

- The owner role who can create new assets and hand them over to other users
- and an operator role who can add new states to an asset without changes in ownership.

Below you can see the basic contract implementation, including events, structs, contract variables and helper methods.

I'm now going to implement three new functions that add states to the supply chain graph in our three new use cases:

- Creating a new asset by adding a state with no precedents;
- appending new states to an existing asset, with respect for ownership
- and handing over the control of an asset to a different account.

The insiġ repository has a fully featured version of this code. For the sake of simplicity I've removed in this article the capability of having multiple precedents for each state.

Conclusion

In this article we have shown how to implement asset ownership for a supply chain. We have deconstructed the purpose of the ERC721 methods to show that in reality asset ownership is a form of access control.

In the implementation we represent asset ownership by controlling who can append new states to a specific asset. To represent asset transfer the owner changes the permissions in the last state of an asset.

This article and implementation is part of a new solution for the manufacturing supply chain. We are disrupting what we know about what we consume and unlocking a massive amount of investment capital currently tied to physical assets.

There is more to follow. In future articles we will show how to represent the merging and splitting of assets and how to tokenize assets in the described supply chain.