Integrated ERC20 <> ERC721

Background

An ERC 20 token representing an asset is connected to a ERC721 contract. The 721 token represents an off-chain clause such as terms and conditions - which can be a PDF document stored on a private IPFS instance, run by the ERC 20 token issuer.

ERC20 token issuer have the ability to initiate updates to the bound documentation (ERC721) of their ERC20 token. However, before these updates can be implemented, multi-sig (or equivalent) sign-off is required from the ERC 20 token holders. This ensures that the updates are valid and approved by all relevant parties before being incorporated into the contract.

For ERC 20 token holder who declines the change in terms and conditions, their token changes state and can only move from their wallet to issuers wallet.

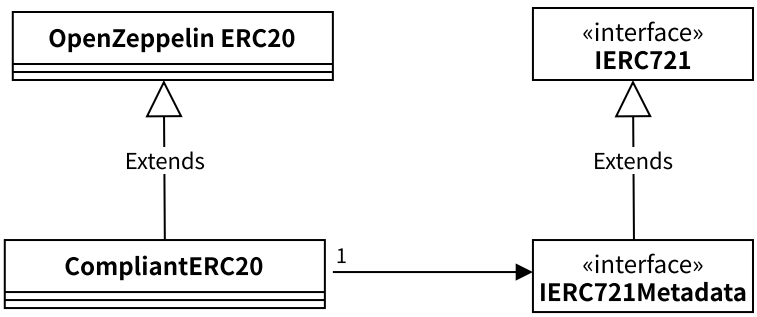
Objective

Implement the above by writing the smart contract for ERC 20 token. For purpose of implementation you can create the ERC 721 token on public IPFS or just use any NFT contract that already exists.

Technical Solution

Since OpenZeppelin has matured ERC20 implementation, the required implementation will be based on OpenZeppelin ERC20. Since I don't want to have dependence on any dedicated ERC721 implementation, the interface IERC721Metadata from OpenZeppelin, IERC721Metadata is used instead of IERC721 because we can read document URI with this interface.

So the code structure is as follows:



There are mainly two roles in the requirement: Token Issuer and Token Holder, and Token Issuer can also be one Token Holder.

For Token Issuer, two functions should be provided:

function issueTokens(address to, uint256 amount) external onlyTokenIssuer

Issue new tokens to specific token holders.

function updateCompliance(uint256 newComplianceId) external onlyTokenIssuer

Initiate updates to the bound documentation

For Token Holder, we need to provide extra functions to signoff the bound documentation change. So the below two functions are provided:

function acceptComplianceUpdate() public

Accept the change in terms and conditions

function declineComplianceUpdate() external

Decline change in terms and conditions

The read-only function is also provided for frontend to read the documentation without the need to interact with ERC721

function complianceURI() external view returns (string memory currentComplianceURI, string memory pendingComplianceURI)

Return both URIs of in-use documentation and to be signed off documentation

Below 4 state variables are used to manage the documentation change cycle:

uint256 public currentComplianceId; // ERC721 token id represents in-use documentation

uint256 public pendingComplianceId; // ERC721 token id represents new documentation

uint256 public tokenHoldersCount; // Count of all token holders

uint256 public pendingComplianceSignOffCount; // Count of token holders already signed off new documentation

The value of "pendingComplianceId" and "pendingComplianceSignOffCount" are both zero in normal state, but when the new documentation is initiated by token issuer, "pendingComplianceId" and "pendingComplianceSignOffCount" will become non-zero. The value of "pendingComplianceSignOffCount" is increased when any token holders accept or decline the new documentation. When the value of "pendingComplianceSignOffCount" equals

the value of "tokenHoldersCount", that means relevant parties have signed off the new doc, "pendingComplianceId" and "pendingComplianceSignOffCount" will both be reset to zero.