MINDZ KONNECTED

1) ERC-20

ERC-20 is a widely adopted standard for creating and managing fungible tokens on the Ethereum blockchain. These tokens can be used for representing digital assets, facilitating ICOs and creating decentralized applications.

Here is the list of some of the functions/features that ERC20 offers :

- ***name()*** : returns the name of the ERC-20 token, usually set during contract deployment.
- ***symbol()*** : returns the symbol used to represent the token, set during contract deployment.
- ***decimal()*** : function to define the number of decimal places that can be used to represent the smallest unit.
- ***totalSupply()*** : The total supply of an ERC-20 token is fixed when the contract is deployed. It makes sure that the token's value is not subject to inflation.
- ***balanceOf(****address _owner)*: returns the token balance of the entered address.
- ***transfer(****address _to, uint value)* ***:*** function to transfer a specific amount of tokens to the desired address.
- ***transferFrom(****address _from, address _to, uint value)* : similar to the transfer() function defined above but we can add a specific address this time from which the tokens are being transferred.
- ***approve(***address _spender, uint value*)* : It is kind of a security function for the token owners. They can set the amount of tokens a spender is allowed to transfer on behalf of them.
- ***allowance(****address _owner, address _spender)* : It is a function to check how many tokens is the spender (the person who is transferring the token on behalf of the token owner) allowed to transfer which was previously set by the token owner in the *approve()* function.

2) ERC-721

ERC-721 defines a standard for creating and managing non-fungible tokens (NFTs) . Unlike ERC-20 tokens which are fungible and interchangeable, ERC-721 tokens are unique and indivisible , meaning a single kind of token can only be owned by one owner. Since each token is now unique, therefore a token id is specified with each token .ERC-721 thus facilitated the creation of digital assets that represent ownership of unique items in the form of digital art, virtual real estate, in-game assets etc.

name(), symbol(),balanceOf(), transfer(), transferFrom(), approve() functions are pretty much the same as ERC-20 with one additional parameter in i.e., token_Id. Apart from that, here are some of the other functions/features that ERC-721 offers :

- ***mint(****address _owner, token_Id, token_URI)* : This function is used to mint i.e., create a token with a specific id and Uri(metadata regarding the nft) and associate it to a specific address who becomes the owner of that token. We can use pinata to generate the token URI and we can add our nfts on platforms like openSea where we can sell and buy NFTs.

- ***ownerOf(****uint token_Id)* : As the name suggests, this function returns the owner of the token on passing the token_Id.
- ***setApprovalForAll(****address _operator, bool _approved)* : This function is a bit similar to the approve() function, the difference being instead of approving permission for each token, it grants/revokes the permission for all the tokens to the operator who is responsible for transferring the tokens on behalf of the token owner.
  There is a function called ***isApprovedForAll****(address _owner, address _operator)* which simply returns a true/false value according to the permission granted by the token owner.
- ***getApproved(****uint token_Id)* : This is a counterpart of the ***allowance()*** function in ERC-20. It returns the address of the person who is allowed to transfer the token on behalf of the token owner.

Apart from these functions there are also functions like ***SafeTransferFrom()*** and ***SafeMint()*** which is quite similar to the mint() and transferFrom() functions but it also allows the user to add in some extra data as parameter which adds an extra layer of security.

3) ERC-1155

ERC-1155 is a multi-token standard that allows the creation of both fungible and non-fungible tokens within a single smart contract. ERC-1155 is much more gas efficient than ERC-721 as it allows batch operations i.e., multiple token transfers and approvals can be performed in a single transaction. Moreover, one does not need to deploy separate contracts for each token type (as with ERC-721 and ERC-20).

name(), symbol(), mint(), balanceOf(), transfer(), transferFrom(), approve() are pretty much the same as in ERC-721.

Here are some functions exclusive to ERC-1155 :

- ***balanceOfBatch(****address[] _accounts, uint _tokenIds)* : Returns the balance of the mentioned token_Id for each address in the accounts array.
- ***safeBatchTransferFrom(****address _from, address _to, uint[] tokenIds, uint[] amount)* : A function to help transfer different amounts of different tokens to a specified address.

4) ERC-6551

ERC-6551 is a relatively new standard meant for non-fungible token bound accounts. It allows every ERC-721 token to have a smart contract wallet associated with it which can be used to own on-chain assets. This means that ERC-6551 can hold all kinds of tokens and other NFTs just like a regular smart contract wallet can.

There are two essential components to consider when talking about ERC-6551 contracts. These are :

i) **Registry Contract :** It serves as an entry point for users trying to make token bound accounts. The Registry Contract has two exclusive functions :

   a. ***createAccount****(address implementation, uint chainid, address tokenContract, uint tokenId, uint salt)* : This function is used to create a token bound account. It requires the address of the *Implementation Contract*, the chainid i.e. , a unique

identifier specific to the blockchain network, the address of the tokenContract which was used to mint the ERC-721 token and the id of that specific token.

b. ***account(****address implementation, uint chainid, address tokenContract, uint tokenId, uint salt)* : This is a deterministic function i.e., this function returns the address of the tba without deploying it, meaning it tells the user that the tba if deployed, will be deployed on this address.

ii) **Implementation Account** : This is the contract which will carry all the functions to work with the TBA. Here are some of the typical functions :

a. ***executeCall(****address _to, uint value, bytes data****)*** : As discussed before, a TBA will hold all the different kinds of assets and tokens. We need a means to be able to transfer the tokens between the contracts, hence this function can help transfer eth and tokens to different addresses.

b. ***token()*** : This function is used to give information about the token : chainid, tokenContract address, tokenId etc. It is basically used by other contracts to verify the ownership of an NFT.

c. ***owner()*** : This function returns the address of the owner of the token associated with the contract. So basically, the token() function is used to retrieve the ERC-721 token associated with the NFT and the owner() function is used to retrieve the address of the owner of the token. So one can say that both these functions are used to verify the ownership of an NFT.

Now, creating a token bound account is a simple three step process :

i) Mint an ERC-721 token and note the token contract address
ii) Deploy the Implementation Contract and note the contract address
iii) Deploy the Registry Contract and call the createAccount() function passing all the necessary parameters.
iv) Redeploy the Implementation Contract on the TBA's address that you got after calling the createAccount() function in the Registry Contract.
v) Now you can call the different functions in the Implementation Contract to interact with your TBA.