ACHouse Functions

create1155MarketItem(address nftContract, uint256 tokenId, uint256 price, uint256 amount) => uint256

nftContract – pass the address of the contract that made the token. If token was minted in our system, then pass ACHouseToken1155 address.

TokenId- pass tokenId of token, you want to enter in market.

Price – set price of token

amount – 1 = NFT, >1 = fungible 1155 token

Returns itemId

create721MarketItem(address nftContract, uint256 tokenId, uint256 price ) => usint256

nftContract – pass the address of the contract that made the token. If token was minted in our system, then pass ACHouseToken1155 address.

TokenId- pass tokenId of token, you want to enter in market.

Price – set price of token

Returns itemId

removeMarketPlaceItem( uint256 itemId)

addMarketPlaceItem( uint256 itemId)

itemId – during create1155MarketItem or create721MarketItem a itemId is created to for all marketplace items.

createMarketSale(address nftContract, uint256 itemId)

nftContract – pass the address of the contract that made the token. If token was minted in our system, then pass ACHouseToken1155 address.

itemId – during create1155MarketItem or create721MarketItem a itemId is created to for all marketplace items.

This function is to be called when buyer had bought the NFT and pays the required eth. This function also transfers ownership of token to buyer.

MarketItem {

uint itemId;

address nftContract;

uint256 tokenId;

address payable seller;

address payable owner;

uint256 price;

uint256 amount;

bool sold;

bool isMultiToken;

bool isRemoved;

}

FetchUnSoldMarketItems() => MarketItem[]

function returns all marketplace items created that are sold yet.

Returns array of objects of type MarketPlace

fetchMyNFTs() => MarketItem[]

Returns all items that user has purchased. User address automatically passed via the tranaction via msg.sender.

FetchItemsCreated() => MarketItem[]

Returns only items created by User. Retuns a array of MarketItem.

ERC1155 functionality-

setURI1155(string memory \_uri)

URI – metadata for the creation of NFT. JSON object that contains info like { Name, desc, imageUrl }

its a good idea to set a proper URI before minting. Waiting on sendmeat to integrate this.

getTokenURI(uint256 \_tokenId) => string

tokenId – id set for a token during minting.

returns the URI set for a specific Token

createNFT1155(uint256 \_id, uint256 \_amount)

id – set a id for the token to be minted.

Amount – set value 1 or more. 1 = NFT, >1 = fungible Token (for fractionalization)

getTokenCount() => uint256

returns the total number of token in the contract.

GetTokenIds() => uint256[]

returns a array of tokenId ever created by this contract

getTokenSupply(uint256 \_tokenId) => uint256

return total supply of token for a tokenId.

Since 1155 can be used for NFT or fungible, if tokenId is NFT then returns 1 else more than 1.

ERC721 Functionality:

createNFT721(uint256 \_id, string memory uri)

\_id – id for token to be created

URI – metadata URI for token

get721TokenCount() => uint256

returns total number of 721 token in contract

get721TokenIds() => uint256[]

returns array of ids of all the token ids generated by contract

get721TokenURI(uint256 \_id) => string

returns the URI set for token

\_id – id of the token

Fractionalization Function:

two functions for 1155 and 721 NFTs to fractionalize.

fractionalize1155NFT(address nftContract, uint256 tokenId, uint256 shardId, uint256 priceOfShard, uint256 supplyToCreate, string memory uri)=> uint256

nftContract – address of the contract holding the token.

TokenId – id of the NFT token to fractionalize

shardId – id of the shard ( fractional token ) to be set

priceofShard – price of each shard to be set.

SupplyToCreate – amount of token to be created.

URI – metadata uri for the token.

Returns fracId – Id of the fractional token.

fractionalize721NFT(address nftContract, uint256 tokenId, uint256 shardId, uint256 priceOfShard, uint256 supplyToCreate, string memory uri) => uint256

nftContract – address of the contract holding the token.

TokenId – id of the NFT token to fractionalize

shardId – id of the shard ( fractional token ) to be set

priceofShard – price of each shard to be set.

SupplyToCreate – amount of token to be created.

URI – metadata uri for the token.

Returns fracId – Id of the fractional token.