# 255.https://stackoverflow.com/questions/71988522/can-i-create-a-token-that-would-be-having-an-array-of-multiple-tokens

**T:**can I create a token that would be having an array of multiple tokens?

**Q:**I am developing a blockchain based food supply chain project in which I have to create raw products as well as a finished product (nugget in my case). I have created an ERC 721 contract for tracking the process but now I have to add that one nugget token contains several raw material tokens. Is it possible? Like a token consisting of an array of other tokens?  
  
// SPDX-License-Identifier: GPL-3.0pragma solidity >=0.4.20;import '@openzeppelin/contracts/token/ERC721/ERC721.sol';// import '@openzeppelin/contracts/token/ERC721/extensions/IERC721Metadata.sol';contract token is ERC721 { address contractOwner; bytes32[] array\_proof; bytes32 hash; constructor() ERC721("supplychain", "SCP") public { contractOwner=msg.sender; } bytes32 tokenID;enum State{ producedBySupplier, forSaleBySupplier, purchasedByManufacturer, shippedBySupplier, receivedByManufacturer, packagedByManufacturer, forSaleByManufacturer, purchasedByRetailer, shippedByManufacturer, receivedByRetailer} struct Item { uint product\_code; address ownerID; State itemState; uint price; } struct Nugget{ uint product\_code; address ownerID; State NuggetState; } //hashtable---------------------------------- mapping(bytes32 => Item ) itemInfo; // mapping(uint => Item) itemforSale; mapping(bytes32 => Nugget) nuggetInfo; mapping(uint => uint) nuggetMap; //events-------------------------------- event lognewItem(bytes32 tokenID,uint createdAt); //create new item by supplierevent lognewNugget(bytes32 tokenID, uint createdAt);event \_purchasedByManufacturer(bytes32 tokenID, uint createdAt); //modiifer --------------------modifier arrayproof(address receiver, uint createdAt){ hash=sha256(abi.encodePacked(msg.sender,receiver,createdAt)); array\_proof.push(hash); \_; } function itemBySupplier(uint weight, uint flavor, uint qty, uint productType)public arrayproof(msg.sender, block.timestamp) returns(uint ) { // uint i; tokenID= sha256(abi.encodePacked(weight, flavor, qty, productType)); Item memory newItem = Item(uint(tokenID),msg.sender,State.producedBySupplier,uint(0)); \_mint(msg.sender,uint(tokenID)); emit lognewItem(tokenID,block.timestamp); return(uint(itemInfo[tokenID].itemState)); } function itemForSale(bytes32 \_tokenId, uint256 \_price) public { require( ownerOf(uint(\_tokenId)) == msg.sender, "You can't sale the item you don't owned" ); itemInfo[(\_tokenId)].price = \_price; //assigning price to that item } function purchasedByManufacturer(address from , address to, bytes32 \_tokenId) public arrayproof(msg.sender, block.timestamp) {(itemInfo[\_tokenId].price > 0, "The item should be up for sale"); safeTransferFrom(from,to,uint(\_tokenId)); emit \_purchasedByManufacturer(\_tokenId,block.timestamp); } function shippedBySupplier(bytes32 \_tokenId) public { //to be seen again based on states itemInfo[(\_tokenId)].itemState=State.shippedBySupplier; } function packagedByManufacturer(uint weight, uint flavor, uint qty, uint productType) public returns (uint){ tokenID= sha256(abi.encodePacked(weight, flavor, qty, productType)); Nugget memory newNugget = Nugget(uint(tokenID),msg.sender,State.packagedByManufacturer); \_mint(msg.sender,uint(tokenID)); emit lognewNugget(tokenID,block.timestamp); return(uint(nuggetInfo[tokenID].NuggetState)); } function validate (address sender, address receiver,uint date, uint txn) public view returns(bool valid){ bytes32 hash\_new= (sha256(abi.encodePacked(sender, receiver, date))); if(array\_proof[txn]==hash\_new){ return(true); } // forsale by supplier will be used for sale by manufacturer }}

0 **Answer**