PerpetualLicense Contract Documentation

Overview

The PerpetualLicense contract is a smart contract for managing perpetual software licenses using the ERC721 token standard. It enables users to purchase and receive licenses, as well as minting new tokens by the owner. This contract also implements royalty for the creators of the license. The metadata is stored as a JSON object encoded in Base64 in the tokenURI. The contract collaborates with a separate LicenseActivation contract to handle the activation and deactivation of licenses.

ERC721 Extension

This contract extends the ERC721 token standard by implementing the royalty and access control extensions of the OpenZeppelin ERC721Royalty and Ownable contracts respectively.

LicenseActivation Contract

The LicenseActivation contract is responsible for managing the activation and deactivation of licenses. It works together with the PerpetualLicense contract to ensure that only the license owner can activate or deactivate their license.

Constructor

Function signature:**constructor(string memory companyName, string memory licenseName, string memory licenseAgreementUrl, uint256 licensePrice, uint96 royaltyPercentage, address licenseActivationContractAddress)**

The constructor function sets the initial state of the contract. It takes six parameters:

1. **companyName**: a string representing the name of the company offering the license.
2. **licenseName**: a string representing the name of the license being offered.
3. **licenseAgreementUrl**: a string representing the URL of the license agreement.Uploaded to distributed file system like IPFS
4. **licensePrice**: a uint256 representing the price of the license in wei.
5. **royaltyPercentage**: a uint96 representing the percentage of royalty that the creator of the license will receive on each sale.

6. **licenseActivationContractAddress**: the address of the LicenseActivation contract to be used with this PerpetualLicense contract.

Functions

PerpetualLicense Contract Functions

buyToken()

- Function signature: function buyToken() public payable

The buyToken function enables users to purchase a perpetual license. It takes no parameters, but requires the user to send the correct amount of ether to cover the license price. If the user sends less ether than the license price, the transaction will revert.

mintToken(address customer)

- Function signature: function mintToken(address customer) public onlyOwner

The mintToken function enables the owner to mint new perpetual license tokens. It takes one parameter:

customer: the address of the user receiving the new license.

tokenURI(uint256 tokenId)

- Function signature: function tokenURI(uint256 tokenId) public view override returns (string memory)

The tokenURI function returns the metadata associated with a particular token. It takes one parameter:

tokenId: the ID of the token for which to return metadata.

getTokenCounter()

- Function signature: function getTokenCounter() public view returns (uint256)

The getTokenCounter function returns the current value of the s_tokenCounter state variable.

getLicensePrice()

- Function signature: function getLicensePrice() public view returns (uint256)

The getLicensePrice function returns the current value of the s_licensePrice state variable.

withdraw()

- Function signature: function withdraw() public onlyOwner

The withdraw function enables the owner to withdraw ether from the contract. It takes no parameters.

updateLicensePrice(uint256 newPrice)

- Function signature: function updateLicensePrice(uint256 newPrice) public onlyOwner

The updateLicensePrice function enables the owner to update the price of the perpetual license. It takes one parameter:

newPrice: the new price of the license in wei.

LicenseActivation Contract Functions

initialize(address perpetualLicenseContractAddress)

- Function signature: function initialize(address perpetualLicenseContractAddress) external onlyOwner

The initialize function sets the address of the PerpetualLicense contract. It takes one parameter:

perpetualLicenseContractAddress: the address of the PerpetualLicense contract.

activateLicense(uint256 tokenId, bytes32 hash, bytes calldata signature)

- Function signature: function activateLicense(uint256 tokenId, bytes32 hash, bytes calldata signature) external

The activateLicense function enables the license owner to activate their license. It takes three parameters:
tokenId: the ID of the token to be activated.
hash: the hash of the license agreement.

signature: the signed message by the license owner.

deactivateLicense(uint256 tokenId)

- Function signature: function deactivateLicense(uint256 tokenId) external

The deactivateLicense function enables the license owner to deactivate their license. It takes one parameter:

tokenId: the ID of the token to be deactivated.

isLicenseActivated(uint256 tokenId)

- Function signature: function isLicenseActivated(uint256 tokenId) public view returns (bool)

The isLicenseActivated function returns the activation status of a particular license. It takes one parameter:

tokenId: the ID of the token to check the activation status.

Inherited Functions from ERC721 Token Standard

The contract inherits the following functions from the ERC721 token standard:

- **balanceOf(address _owner)**: Returns the number of tokens owned by a particular address.
- **ownerOf(uint256 _tokenId)**: Returns the owner of a particular token.
- **approve(address _approved, uint256 _tokenId)**: Approves another address to transfer the given token ID.
- **getApproved(uint256 _tokenId)**: Returns the approved address for a token ID, or zero if no address is approved.
- **setApprovalForAll(address _operator, bool _approved)**: Enables or disables approval for a third party ("operator") to manage all of the caller's tokens.
- **isApprovedForAll(address _owner, address _operator)**: Returns true if the given operator is approved to manage all of the caller's tokens.
- **transferFrom(address _from, address _to, uint256 _tokenId)**: Transfers the ownership of a given token ID to another address.
- **safeTransferFrom(address _from, address _to, uint256 _tokenId, bytes memory _data)**: Safely transfers the ownership of a given token ID to another address, checking first that the recipient is a contract and that it supports ERC721Receiver or ERC721ReceiverUpgradeable interfaces.

- **safeTransferFrom(address _from, address _to, uint256 _tokenId)**: Safely transfers the ownership of a given token ID to another address, checking first that the recipient is a contract and that it supports ERC721Receiver or ERC721ReceiverUpgradeable interfaces with extra data.
- **royaltyInfo(uint256 _tokenId, uint256 _value)**: Returns the royalty fee information for a given token ID, including the fee recipient and fee amount.

These functions are inherited from the ERC721 standard, which is extended by ERC721Royalty. For additional functionalities, please refer to the official documentation of the ERC721 token standard available here. In addition to these functions, ERC721Royalty adds a royalty fee mechanism for each token.

The activation process works as follows:

1. The license owner calls the **activateLicense** function on the LicenseActivation contract with the tokenId, hash, and their signature.
2. The contract checks if the license is already activated. If it is, the transaction reverts.
3. The contract recovers the signer address from the given signature and compares it to the license owner's address. If they don't match, the transaction reverts.
4. If everything is valid, the contract updates the activation status of the license and emits an Activation event.

The deactivation process is similar:

1. The license owner calls the **deactivateLicense** function on the LicenseActivation contract with the tokenId.
2. The contract checks if the license is activated. If it is not, the transaction reverts.
3. The contract checks if the sender is the owner of the license. If not, the transaction reverts.
4. If everything is valid, the contract updates the activation status of the license and emits a Deactivation event.