**NFT Marketplace Smart Contract - README**

**Overview**

This NFT Marketplace Smart Contract is designed to facilitate the locking and unlocking of NFTs (Non-Fungible Tokens) in a secure and decentralized manner on the Cardano blockchain. The smart contract allows sellers to lock their NFTs in the contract and unlock them upon receiving the required payment from buyers, including a marketplace fee for the marketplace owner.

**Key Components**

**Data Types**

- **NFTIdentifier**: Represents the unique identifier of an NFT (TokenName).
- **Seller**: Represents the public key hash (PubKeyHash) of the seller.
- **Price**: Represents the price of the NFT in Lovelaces (1 ADA = 1,000,000 Lovelaces).
- **MarketplaceOwner**: Represents the public key hash of the marketplace owner.
- **Buyer**: Represents the public key hash of the buyer.
- **Lovelace**: Represents ADA values in Lovelaces.

**NFTSale**

The NFTSale data type holds information about the NFT sale, including:

- nftId: The unique identifier of the NFT.
- nftImage: The image of the NFT in BuiltinByteString format.
- nftSymbol: The symbol of the NFT in BuiltinByteString format.
- seller: The public key hash of the seller.
- price: The price of the NFT in Lovelaces.

**MarketplaceDatum**

The MarketplaceDatum data type represents the state of an NFT in the marketplace contract. It can be either:

- NFTLocked NFTSale: Indicates an NFT is locked in the contract with the corresponding sale information.
- NoNFTLocked: Indicates no NFT is currently locked in the contract.

**MarketplaceAction**

The MarketplaceAction data type defines the possible actions that can be performed on the contract:

- Lock: Action to lock an NFT in the contract.
- Unlock: Action to unlock an NFT from the contract.

**Marketplace Constants**

- **marketplaceOwner**: Placeholder for the marketplace owner's public key hash.
- **marketplaceFee**: Calculates the marketplace fee as 1/20th of the NFT price.

**Functions**

**Payment Verification**

- **paymentMadeTo**: Verifies if a payment of a certain amount has been made to a specific address.
- **isPaidTo**: Checks if a transaction output is paid to a given public key hash.
- **allPaymentsCorrect**: Checks if all required payments (to seller and marketplace owner) are correctly made in the transaction.

**NFT Verification**

- **inputsContainNFT**: Checks if the transaction inputs contain the specified NFT.
- **feePaidToMarketplace**: Verifies if the marketplace fee is paid to the marketplace owner.
- **containsAsset**: Checks if a transaction output contains the specified amount of a given asset.
- **findNftIdToLock**: Identifies the NFT to be locked in the contract based on certain criteria.

**Ownership Transfer**

- **nftTransferred**: Checks if the NFT has been transferred to the buyer in the transaction.

**Utility Functions**

- **isNftUnique**: Placeholder function, always returns True to check uniqueness of the NFT.
- **isNftTokenName**: Placeholder function, always returns True to check if the token name follows NFT naming conventions.

**Validator Logic**

The mkValidator function is the core logic of the smart contract, validating transactions based on the provided datum and redeemer. It ensures:

- NFTs are not locked again if already locked.
- NFTs are unlocked only if they are locked.
- Payments are correctly made to the seller and marketplace owner during unlocking.
- NFTs are correctly transferred during unlocking and locking.

**Conversion to BuiltinData**

- **untypedLockUnlockValidator**: Converts the typed validator to an untyped validator.

**Pre-compilation**

- **LockUnlockValidator**: Defines the type for the validator contract.
- **compileLockUnlockValidator**: Pre-compiles the validator for deployment.

**Sample Data**

- **sampleSeller**: Represents a sample seller.
- **sampleBuyer**: Represents a sample buyer.
- **sampleMarketplaceOwner**: Represents a sample marketplace owner.

**Datum Construction**

- **mkLockUnlockDatum**: Constructs the datum for exporting with sample data.

**Jambhala Exports**

The lockUnlockValidatorExports defines the exports for Jambhala CLI, including:

- Sample data exports.
- Emulator test setup.

**Validator Endpoints**

The validator endpoints define the parameters and implementations for locking and unlocking NFTs:

- **GiveParam**: Represents the parameters required to lock an NFT.
- **GrabParam**: Represents the parameters required to unlock an NFT.
- **give**: Implements the function to lock an NFT.
- **grab**: Implements the function to unlock an NFT.

**Emulator Test**

The test function initializes the emulator with two wallets and tests the unlocking of an NFT from wallet 2.

**Usage**

1. **Locking an NFT**:
    - Call the give function with LockNFT parameter to lock an NFT in the contract.
2. **Unlocking an NFT**:
    - Call the grab function with UnlockNFT parameter to unlock an NFT from the contract.
3. **Emulator Testing**:
    - Use the test function to initialize the emulator and run the test scenarios.

**Conclusion**

This NFT Marketplace Smart Contract provides a secure and decentralized mechanism for locking and unlocking NFTs on the Cardano blockchain. By following the outlined functions and usage instructions, users can effectively interact with the contract to manage their NFT transactions.