

Apartment Staking Smart Contract

Introduction

The Ethereum Worlds team are introducing an ERC721 Staking Contract, that will enable holders of their NFTs to stake into a custom smart contract and be rewarded in the native ERC20 tokens. The NFT holder will be able to stake and unstake their NFT via a Web portal which will also handle the number of ERC20 tokens a specific holder will receive. This reward logic is handled entirely off-chain.

Smart Contracts URLs

Apartment Staking Smart Contract	Test	https://ropsten.etherscan.io/address/0xB3abBDFbC89DA3aEdFa638888F41E092d72a78Ac#code
Apartment Staking Smart Contract	Prod	-
721 (Apartment) Emission Smart Contract	Prod	https://etherscan.io/address/0x72b3b26848a30d3d76e5b67959a365b6ad388c49#code
ERC20 Emission Smart Contract	Test	https://ropsten.etherscan.io/address/0x1944937643d8317D23bc49766412385F8ec6452F#code


Roles

1. User - a person, who interacts with the system before the identity verification.
2. NFT Owner - a user, who connected the wallet and has NFT (ERC721 tokens) of an apartment in the Ethereum Worlds on their balance or on Staking SC balance (staked).
3. System - a set of the units connected with each other (staking smart contract, backend, frontend)
4. Admin - a user, who manages system settings in Admin Panel (off-chain).
5. Smart Contract Owner - a Smart Contract key holder, who owns the Contract and can manage it.

General information

Staking

NFT staking refers to locking up non-fungible tokens on a platform in exchange for staking rewards and other benefits. Staking NFTs allows holders to earn an income from their collection while maintaining ownership.

 Apartment owners cannot sell or transfer their NFT tokens until they are unstaked.

Reward logic

ERC20 tokens (\$TWR) will be used as currency for the reward for NFT staking.

The amount of the reward accumulates each hour (it means it increases every hour by the hourly reward amount). And as the mechanism of staking is flexible, the accumulated earned reward might be claimed whenever.

The user claims the amount that was accumulated for the rounded down hours of staking (if staking has lasted for 389 hours and 57 minutes, the user will get the reward for 389 hours).

The remaining 57 minutes will be claimed later with the subsequent minutes and hours if the NFT is remained staked. If the claim is done while unstaking, the user will not be able to claim the reward for these 57 minutes.

<u>Hourly</u> Staking Reward Formula for Standard Apartments and Penthouse Apartments	$= (\text{Base Reward} \times \text{Standard Multiplier}) / 24$
<u>Hourly</u> Staking Reward Formula for Luxury Apartments	$= ((\text{Base Reward} \times \text{Standard Multiplier}) + (\text{Base Reward} \times \text{Luxury Multiplier})) / 24$

All the parameter values are set in the Admin Panel by the admin.

As the parameters can be edited, there is a mechanism that regulates the calculation with the updated info: Updated Base Reward (BR) will not affect the sum of the reward that was accumulated before the BR value was changed, but all next time periods that are after the change will be calculated according to the new BR value.



All the reward calculations and regulating mechanisms will be set on the backend.

Environment Details

The solution is mix-implemented: part of it set on the backend (BE), and another part on-chain.

NFT staking logic that is set in the Staking Smart Contract. It includes:

- NFT transfers (from user's wallet to the Staking Smart Contract and vice versa)
- Storing the data of the stakes (timestamps, staker wallet, staked token)
- Checks on total staked token amount
- Transfer ERC20 tokens (\$TWR) as a reward and saving the data that reward was claimed
- Option to get all the staked tokens back in case of an emergency.

Other features are covered by the backend (BE):

- Wallet connection and verification the NFT (original NFT from the [Apartment Emission SC](#))
- Defining the user role and their permissions
- Calculation the hourly reward based on the multipliers and base reward value (both are set in the Admin Panel).

NFT staking UI will interact with the smart contract via backend. The UI allows users to do the following actions:

- Connect wallet
- Create a Profile (needed for another part of the Web Application)
- Stake/Unstake NFT
- See the amount of the accumulated reward
- Claim the reward

Admin panel features will be added to the existing Admin Application.

User roles and available features

Role	Business Use Cases
NFT owner	<ul style="list-style-type: none">• Connect wallet• Stake/Unstake NFT• Check the accumulated amount of the reward• Claim the reward
System	<ul style="list-style-type: none">• Check NFT on the balance• Stake/unstake logic• Transfer NFT to/from contract• Save the details of all transactions• Calculate staking reward• Transfer the reward
Admin	<ul style="list-style-type: none">• Connect wallet• Set the base reward and multipliers• Monitor and export the wallet addresses whose NFTs are staked longer than some periods of time (1 month etc)• Export the list of staked NFTs• Edit the number of NFTs that can be staked at the same time• Transfer some \$TWR to the staking SC

Staking Smart Contract

Staking Smart Contract allows NFT Owners to use their Apartment NFTs (ERC721) to earn ERC20 (\$TWR) tokens while NFTs are not used.

The contract is set as:

- IERC721Receiver (An interface for any contract that wants to support safeTransfers from ERC721 asset contracts)
- EIP712 (A standard for hashing and signing of typed structured data)
- Ownable (An extension that allows transferring Ownership)
- Pausable (A common emergency response mechanism that can pause functionality while a remediation is pending)
- ReentrancyGuard (A contract module that helps prevent reentrant calls to a function)

The Contract covers 3 main functions:

- Stake
- Unstake
- Claim

And additional secondary functions.

Main functions can be called only [when SC is not paused] via Ethereum Towers Backend as Smart Contract can recognize only requests that have cryptographic digital signature.

User calls function on the Frontend (FE) BE adds a cryptography digital signature to the data user signs this transaction with data in MetaMask transaction is sent to the SC.

There are some restrictions that are set in the Contract:

- the maximum number of tokens that can be staked at the same time = 4388 by default (max number of Apartments) [can be changed by Admin]
- maximum number of tokens to be unstaked in case of emergency at the same time = 20, to avoid out of gas issues (can't be changed)

Stake Function

There are a few checks that must be done when the function is called:

- There must be at least 1 official NFT to stake
- The maximum number of tokens that are staked at the same time must not be reached
- Signature must be valid (signed by backend service address)
- The user who calls the function and voucher owner must be the same person (to ensure that the data wasn't intercepted by imposters)
- The user who calls the function must be the real owner of the NFT
- The same voucher can't be used twice

Once all the checks are performed successfully stake function is called:

- NFT is transferred to the Contract's balance
- Stake Information is saved (timestamp when Staking starts, stake owner)
- Total number of tokens staked in the Contract is increased by the staked number of NFTs
- Event that NFT is staked is emitted
- Mark Signature as used (so that it couldn't be used again)
- RewardClaimTimestamp is set (to allow users to claim at any moment)

Unstake Function

Smart contract must do the following checks when the function is called:

- The Smart Contract must have at least 1 NFT on the balance that was staked by the user who called the function
- Signature must be valid (signed by backend service address)
- The user who calls the function and voucher owner must be the same person (to ensure that the data wasn't intercepted by imposters)
- The user who calls the function must be the real owner of the NFT
- The same voucher can't be used twice

If all the checks are done successfully Unstake function is called :

- NFT is transferred back to the owner's wallet or any other wallet that owner entered (to save the gas fee)
- **Optional step** - transfer rewards if it was requested by user (claimAmount > 0 in voucher) otherwise do nothing on this step
- Total number of tokens staked in the Contract is decreased by the unstaked number of NFTs
- Event that NFT is unstaked is emitted
- Mark Signature as used (so that it couldn't be used again)

Claim Function

When User calls "Claim" function on the FE, the BE sends the reward amount that user's earned by the moment to the SC.

Smart contract must do the following checks when the Claim function is called:

- The Smart Contract must have enough \$TWR on the balance to pay the Reward to the user
- The user who calls the function must be actual token stake owner
- Signature must be valid (signed by backend service address)
- The user who calls the function and voucher owner must be the same person (to ensure that the data wasn't intercepted by imposters)
- Reward is > \$TWR 0

If all the checks are done successfully Claim function is called :

- Reward sum (in \$TWR) must be sent from the SC balance to the user's balance
- RewardClaimTimestamp for requested token must be updated with current block timestamp (to correctly count rewards)
- Reward is marked as claimed (so that user couldn't get it twice)
- Event that reward is claimed is emitted (includes claimed reward amount)
- Mark Signature as used (so that it couldn't be used again)

Claim all Function

When User calls "ClaimAll" function on the FE, the BE sends the reward amount that user's earned by the moment to the SC. This function is used to claim all rewards for all staked tokens.

Smart contract must do the following checks when the ClaimAll function is called:

- The Smart Contract must have enough \$TWR on the balance to pay the Reward to the user
- The user must have at least one token staked
- Signature must be valid (signed by backend service address)
- The user who calls the function and voucher owner must be the same person (to ensure that the data wasn't intercepted by imposters)
- Reward is > \$TWR 0

If all the checks are done successfully ClaimAll function is called :

- Reward sum (in \$TWR) must be sent from the SC balance to the user's balance
- rewardClaimTimestamp for all staked tokens updated with current block timestamp (to correctly count rewards)
- Reward is marked as claimed (so that user couldn't get it twice)
- Event that reward claimed is emitted (includes claimed reward amount)
- Mark Signature as used (so that it couldn't be used again)

Secondary functions

Emergency Unstake

Might be called by NFT Owners only if Contract Owner switched "Shut Down" Mode on.

All the actions within this function are the same as in Unstake , the only difference is that user can unstake many tokens at once.

There is a restriction for users, who have more than 20 NFTs staked:

- the maximum number of tokens per 1 Unstake = 20 (to make sure that the transaction will be successful. If the transaction includes more than 20 tokens it will be expensive and might not fit into 1 block)

Only Contract Owner functions

There are some functions that can be called only by Contract Owner in the specific use cases:

- Transfer ownership (Ownable extension)
- Transfer ERC20 to the SC balance (as staking reward vault)
- Turn on "Shut down" mode (to make Emergency Unstake enable)
- Pause transactions within the contract (Pausable extension)
- Edit parameters (max number of tokens that are staked at the same time)
- Update backend service signer address (used for vouchers signing)