

Enumeration

In the previous tutorials, you looked at ways to integrate the minting functionality into a skeleton smart contract. In order to get your NFTs to show in the wallet, you also had to deploy a patch fix that implemented one of the enumeration methods. In this tutorial, you'll expand on and finish the rest of the enumeration methods as per the [standard](#). Now you'll extend the NFT smart contract and add a couple of enumeration methods that can be used to return the contract's state.

caution The JS-SDK is currently in [Alpha](#).

Introduction

As mentioned in the [Upgrade a Contract](#) tutorial, you can deploy patches and fixes to smart contracts. This time, you'll use that knowledge to implement `thenft_total_supply`, `nft_tokens` and `nft_supply_for_owner` enumeration functions.

To get started, either switch to the `2.minting` branch from our [GitHub repository](#), or continue your work from the previous tutorials. If you haven't cloned it yet, refer to the [Contract Architecture](#) to check out the repository.

git checkout 2.minting tip If you wish to see the finished code for this Enumeration tutorial, you can find it on the `3.enumeration` branch.

Modifications to the contract

Let's start by opening the `src/enumeration.ts` file and locating the empty `internalNftTotalSupply` function.

NFT Total Supply

This function should return the total number of NFTs stored on the contract. You can easily achieve this functionality by simply returning the length of the `thenftMetadataById` data structure.

`src/nft-contract/enumeration.ts` loading ... [See full example on GitHub](#)

NFT Tokens

This function should return a paginated list of `JsonTokens` that are stored on the contract regardless of their owners. If the user provides a `from_index` parameter, you should use that as the starting point for which to start iterating through tokens; otherwise it should start from the beginning. Likewise, if the user provides a `limit` parameter, the function shall stop after reaching either the limit or the end of the list.

`src/nft-contract/enumeration.ts` loading ... [See full example on GitHub](#)

NFT Supply For Owner

This function should look for all the non-fungible tokens for a user-defined owner, and return the length of the resulting set. If there isn't a set of tokens for the provided Account ID, then the function shall return 0.

`src/nft-contract/enumeration.ts` loading ... [See full example on GitHub](#) Next, you can use the CLI to query these new methods and validate that they work correctly.

Redeploying the contract

Now that you've implemented the necessary logic for `nft_tokens_for_owner`, it's time to build and re-deploy the contract to your account. Using the build script, deploy the contract as you did in the previous tutorials:

`yarn build && near deploy --wasmFile build/nft.wasm --accountId NFT_CONTRACT_ID` This should output a warning saying that the account has a deployed contract and will ask if you'd like to proceed. Simply type `y` and hit enter.

This account already has a deployed contract [`AKJK7sCysrWrFZ976YVBnm6yzmJuKLzdAyssfzK9yLsa`]. Do you want to proceed? (y/n)

Enumerating tokens

Once the updated contract has been redeployed, you can test and see if these new functions work as expected.

NFT tokens

Let's query for a list of non-fungible tokens on the contract. Use the following command to query for the information of up to

50 NFTs starting from the 10th item:

`near view NFT_CONTRACT_ID nft_tokens '{"from_index": "10", "limit": 50}'` This command should return an output similar to the following:

Example response: []

Tokens by owner

To get the total supply of NFTs owned by `thegoteam.testnet` account, call `thenft_supply_for_owner` function and set `theaccount_id` parameter:

`near view NFT_CONTRACT_ID nft_supply_for_owner '{"account_id": "goteam.testnet"}'` This should return an output similar to the following:

Example response: 0

Conclusion

In this tutorial, you have added two [new enumeration functions](#), and now you have a basic NFT smart contract with minting and enumeration methods in place. After implementing these modifications, you redeployed the smart contract and tested the functions using the CLI.

In the [next tutorial](#), you'll implement the core functions needed to allow users to transfer the minted tokens.

Remember If you want to see the finished code from this tutorial, you can checkout the `3.enumeration` branch. Versioning for this article At the time of this writing, this example works with the following versions:

- near-cli:3.0.0
- NFT standard:[NEP171](#)
- , version1.0.0
- Enumeration standard:[NEP181](#)
- , version1.0.0 [Edit this page](#) Last updated on Jan 19, 2024 by Damián Parrino Was this page helpful? Yes No

[Previous Upgrade a Contract](#) [Next Core](#)