



MINTING NFTS ON THE **SIMBA ENTERPRISE PLATFORM**



MINTING NON-FUNGIBLE TOKENS (NFTs)

This walk through demonstrates how easy it is to mint and interact with an ERC721 Non-Fungible Token (NFT) on the SIMBA Enterprise Platform. Open Zeppelin Solidity libraries were used as a starting point to build our example NFT smart contract. The following piece of media will be used as our NFT:



ERC721 EXAMPLE CONTRACT - DEPLOYING THE CONTRACT AND MINTING A SIMBA NFT:

Example Solidity smart contract code:

```
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.3;

import "@openzeppelin/4.0.0/contracts/token/ERC721/ERC721.sol";
import "@openzeppelin/4.0.0/contracts/utils/Counters.sol";
import "@openzeppelin/4.0.0/contracts/token/ERC721/extensions/ERC721URIStorage.sol";

contract SimbaNFTEExample is ERC721URIStorage {
    using Counters for Counters.Counter;
    Counters.Counter private _tokenIds;

    constructor() ERC721("SimbaNFTEExample", "SIMBA") {}

    function mint(address to, string memory tokenURI) public returns (uint256)
    {
        _tokenIds.increment();

        uint256 tokenId = _tokenIds.current();
        _mint(to, tokenId);
        _setTokenURI(tokenId, tokenURI);

        return tokenId;    }
}
```

All NFTs minted by an NFT contract need a unique tokenId. The tokenIds for the NFTs minted by our example smart contract (SimbaNFTExample) will use one up numbers starting with 1. The ERC721URIStorage extension is used to allow for adding a tokenURI to the NFT. This is typically a URL used to reference a JSON metadata file containing information about the NFT. This JSON metadata file will contain a reference to the location of the NFT media file (the artwork, video, song, etc.).

```

1 // SPDX-License-Identifier: MIT
2 pragma solidity ^0.8.3;
3
4 import "@openzeppelin/4.0.0/contracts/token/ERC721/ERC721.sol";
5 import "@openzeppelin/4.0.0/contracts/utils/Counters.sol";
6 import "@openzeppelin/4.0.0/contracts/token/ERC721/extensions/ERC721URIStorage.sol";
7
8 contract SimbaNFTExample is ERC721URIStorage {
9     using Counters for Counters.Counter;
10     Counters.Counter private _tokenIds;
11
12     constructor() ERC721("SimbaNFTExample", "SIMBA") {}
13
14     function mint(address to, string memory tokenURI) public returns (uint256)
15     {
16         _tokenIds.increment();
17
18         uint256 tokenId = _tokenIds.current();
19         _mint(to, tokenId);
20         _setTokenURI(tokenId, tokenURI);
21
22         return tokenId;
23     }
24 }

```

Console
Success

Expert Mode

To get started using the SIMBA Enterprise Platform we first take the SimbaNFTExample smart contract and load it into expert mode of the Smart Contract Designer. We then check that it compiles and save the contract as SimbaNFTExample. Next we add it to an application called SimbaNFTExample and deploy the contract to a Quorum blockchain network:

SimbaNFTExample
9/15/2021, 12:36:41 PM
simbachain
RPI
<https://api.sep.dev.simbachain.com/v2/apps/SimbaNFTExample/>

Contracts

- SimbaNFTExample 9/15/2021, 12:37:21 PM
- SimbaNFTExample COMPLETED

Blockchain: ethereum
Network: Quorum
Address: 0xCBe9D409aA41362efcA92B500a93C2aF80A09A45
Storage: no_storage

Search: Filter
Showing 1 of 1, 1 completed, 0 pending, 0 failed

Transactions - All

0xe18af005d8e63db4628a9866b0ba96dd0bc78a8dca22f4ebd1f583a6ae3191f **completed**
9/15/2021, 12:37:21 PM

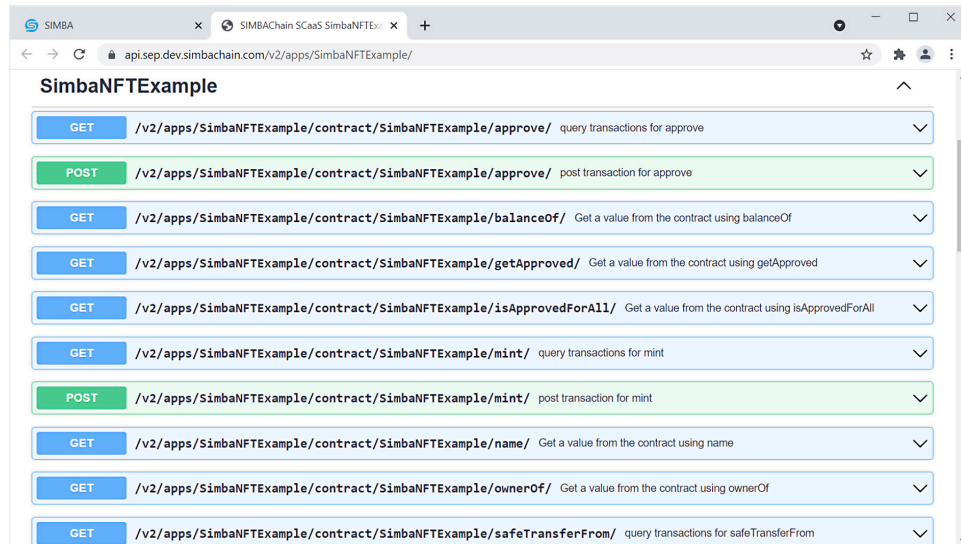
From: 0x303c9478d744e00d7EC1d6c62976204FbAb76786
Contract: SimbaNFTExample
Inputs:
Raw Transaction:
Receipt:

Showing latest 1 of 1, 1 completed, 0 pending, 0 failed



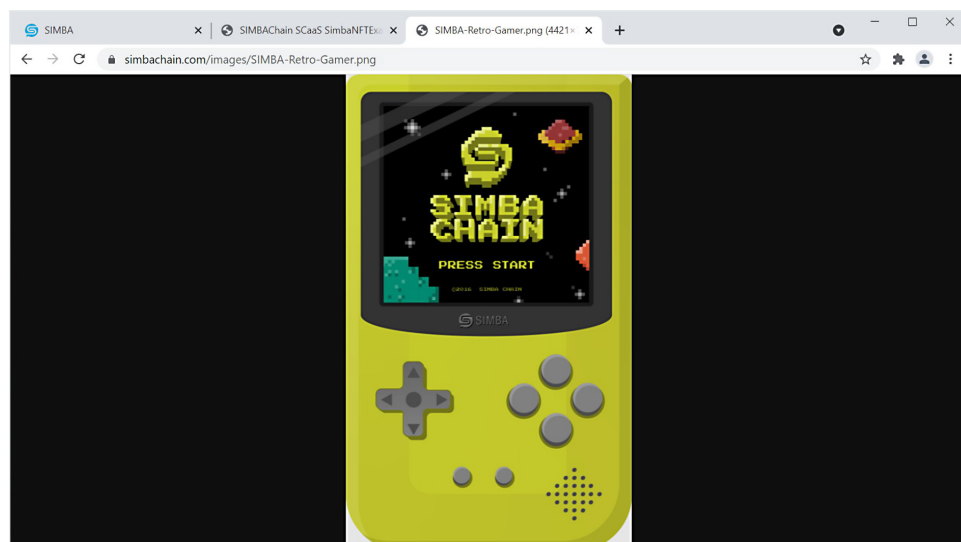
Note: This Solidity smart contract can also be deployed through the SIMBA Enterprise Platform to public networks such as Ethereum and Avalanche.

After deployment, the generated REST API can now be used to interact with the SimbaNFTEExample contract functionality:



To mint an NFT using the smart contract we just deployed we would first call the mint function. This function takes two arguments, the to address and the tokenURI. The to address will be the owner of the NFT once minted. The tokenURI is a URL that will point to a JSON file that contains the NFT metadata. This file describes the NFT itself, and most importantly contains the location of the actual NTF media. Before we can call the mint function, we first need to have our NFT media and metadata files available.

For this example we first uploaded our media content for the NFT to:
<https://simbachain.com/images/SIMBA-Retro-Gamer.png>



SIMBA Chain's approach to minting NFTs involves incorporating the metadata and content based identifiers into the NFT on chain. However, for simplicity for this demo we will be storing our NFT related media content on the SIMBA Chain website. Consequently, the next step is to take the URL for the media and populate a JSON metadata file for the NFT we are about to mint. In this JSON we also set the id field to 1 as this will be the tokenId generated by the contract for the first token we mint, and then populate the name and description fields for the NFT.

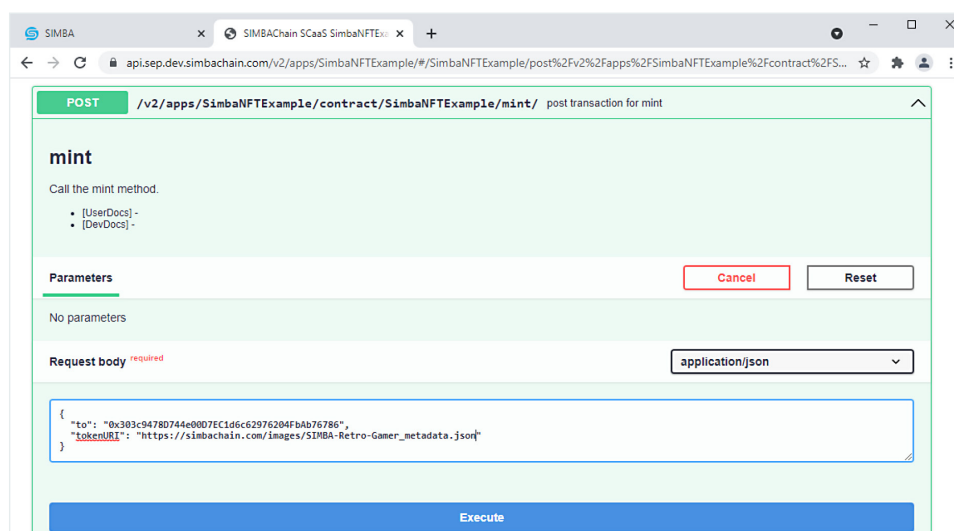
```
{
  "id": "1",
  "name": "Retro Gamer",
  "description": "SIMBA Retro Gamer NFT",
  "image": "https://simbachain.com/images/SIMBA-Retro-Gamer.png"
}
```

Again, for simplicity we will be storing our NFT metadata file on the SIMBA Chain website for this example, but storage in a decentralized platform such as IPFS is recommended. Our metadata file can be found at https://simbachain.com/images/SIMBA-Retro-Gamer_metadata.json.



We can now mint the NFT by calling the mint function by providing the address we would like to own the token as the to argument, and the URL for our metadata file as the tokenURI argument. The request body arguments should look like the following:

```
{
  "to": "0x303c9478D744e00D7EC1d6c62976204FbAb76786",
  "tokenURI": "https://simbachain.com/images/SIMBA-Retro-Gamer_metadata.json"
}
```



Reloading the application page we can now see that a second transaction completed. This transaction called the mint method with the request body we provided above.

The screenshot shows the SimbaChain application interface. On the left, the 'SimbaNFTEExample' app is listed with a status of 'COMPLETED'. Below it, the 'Contracts' section shows the 'SimbaNFTEExample' contract with details: Blockchain: ethereum, Network: Quorum, Address: 0x0Be9D409aA41362efcA92B500a93C2aF80A09A45, and Storage: no_storage. On the right, the 'Transactions' section shows two completed transactions. The first transaction is for the 'mint' method with a tokenURI pointing to a metadata file. The second transaction is also for the 'mint' method with an empty input object. Both transactions show the raw transaction, receipt, and error details.

At this point we have successfully minted our first NFT. If we click the tokenURI link it takes us to the metadata file we set up for our NFT before minting:

The screenshot shows the metadata file for the first NFT. The file is located at https://simbachain.com/images/SIMBA-Retro-Gamer_metadata.json. The metadata is a JSON object with the following structure:

```
{ "id": "1", "name": "Retro Gamer", "description": "SIMBA Retro Gamer NFT", "image": "https://simbachain.com/images/SIMBA-Retro-Gamer.png" }
```



To verify that our NFT was minted on the blockchain we can now call the smart contract's tokenURI function, with the tokenId set to 1:

The screenshot shows a web browser window with a REST client interface. The URL bar shows the endpoint: `api.sep.dev.simbachain.com/v2/apps/SimbaNFTEExample/#/SimbaNFTEExample/get%2Fv2%2Fapps%2FSimbaNFTEExample%2Fcontract%2FSim...`. The method is set to **GET** and the path is `/v2/apps/SimbaNFTEExample/contract/SimbaNFTEExample/tokenURI/`. The description says "Get a value from the contract using tokenURI".

Parameters

Name	Description
tokenId * required	number
number (query)	<input type="text" value="1"/>

Execute **Clear**

Responses

Curl

```
curl -X 'GET' \
  'https://api.sep.dev.simbachain.com/v2/apps/SimbaNFTEExample/contract/SimbaNFTEExample/tokenURI/?tokenId=1' \
  -H 'accept: application/json'
```

Request URL

```
https://api.sep.dev.simbachain.com/v2/apps/SimbaNFTEExample/contract/SimbaNFTEExample/tokenURI/?tokenId=1
```

Server response

Code	Details
200	<div><p>Response body</p><pre>{ "request_id": "9778bffa-1ed4-4e87-98c6-f172c3baa8c", "value": "https://simbachain.com/images/SIMBA-Retro-Gamer_metadata.json", "state": "COMPLETED" }</pre><p>Download</p></div>

Response headers

This returns the metadata URL we used when minting the NFT:
https://simbachain.com/images/SIMBA-Retro-Gamer_metadata.json

We can also call the `ownerOf` function with the `tokenId` set to 1 to verify that the address we provided during minting (`0x303c9478D744e00D7EC1d6c62976204FbAb76786`) is the owner of the NFT. We can see in the response data that owner address does match the address we provided at minting.

The screenshot shows a web browser window with the URL `api.sep.dev.simbachain.com/v2/apps/SimbaNFTExample/#/SimbaNFTExample/get%2Fv2%2Fapps%2FSimbaNFTExample%2Fcontract%2FSim...`. The interface is for the `ownerOf` function, which is described as "Get a value from the contract using ownerOf". It includes a "Parameters" section with a `tokenId` field (required, number, query) set to `1`. Below the parameters are "Execute" and "Clear" buttons. The "Responses" section shows the "Server response" with a status code of `200` and a "Response body" containing the following JSON:

```
{
  "request_id": "b28055ed-1829-4e10-8524-c615b084ce4d",
  "value": "0x303c9478d744e00d7ec1d6c62976204fbab76786",
  "state": "COMPLETED"
}
```

The interface also includes a "Curl" section with the following command:

```
curl -X 'GET' \
  'https://api.sep.dev.simbachain.com/v2/apps/SimbaNFTExample/contract/SimbaNFTExample/ownerOf/?tokenId=1' \
  -H 'accept: application/json'
```

And a "Request URL" section with the following URL:

```
https://api.sep.dev.simbachain.com/v2/apps/SimbaNFTExample/contract/SimbaNFTExample/ownerOf/?tokenId=1
```



To transfer ownership of the NFT to a new owner with address 0xca843569e3427144cead5e4d5999a3d0ccf92b8e, we can call the transferFrom function. To call transferFrom, we set the from field to the current owner's address (0x303c9478D744e00D7EC1d6c62976204FbAb76786), then set the to field to the new owner's address (0xca843569e3427144cead5e4d5999a3d0ccf92b8e), and set the tokenId field to 1:

POST /v2/apps/SimbaNFTExample/contract/SimbaNFTExample/transferFrom/ post transaction for transferFrom

transferFrom

Call the transferFrom method.

- [UserDocs] -
- [DevDocs] - [See [IERC721-transferFrom].]

Parameters Cancel Reset

No parameters

Request body required application/json

```
{
  "from": "0x303c9478D744e00D7EC1d6c62976204FbAb76786",
  "to": "0xca843569e3427144cead5e4d5999a3d0ccf92b8e",
  "tokenId": 1
}
```

Execute Clear

Responses

Curl

```
curl -X 'POST' \
  'https://api.sep.dev.simbachain.com/v2/apps/SimbaNFTExample/contract/SimbaNFTExample/transferFrom/' \
  -H 'accept: application/json' \
  -H 'content-type: application/json' \
  -d '{
    "from": "0x303c9478D744e00D7EC1d6c62976204FbAb76786",
    "to": "0xca843569e3427144cead5e4d5999a3d0ccf92b8e",
    "tokenId": 1
  }'
```

Back on the application page we can see that this results in a third transaction being submitted to the blockchain for the transferFrom method with the inputs we provided in the previous step:

The screenshot displays the Simba NFT application interface in a web browser. The browser tabs show 'SIMBA' and 'SIMBACHain SCaaS SimbaNFTExample'. The address bar shows 'sep.dev.simbachain.com/simbachain/app/SimbaNFTExample'. The application has a top navigation bar with 'APP', 'CONTRACT', and 'SUBSCRIPTION' tabs, and a 'Simbachain' user profile icon.

The main content area is divided into two panels. The left panel, titled 'SimbaNFTExample', shows the contract details: '9/15/2021, 12:36:41 PM', 'simbachain', and a URL. Below this is a 'Contracts' section with a list of contracts, including 'SimbaNFTExample' which is marked 'COMPLETED'. It also shows the blockchain (ethereum), network (Quorum), address, and storage details.

The right panel, titled 'Transactions - All', shows a list of transactions. The first transaction is highlighted, showing its hash, completion status, and details: 'From: 0x303c9478D744e00D7EC1d6c62976204FbAb76786', 'Contract: SimbaNFTExample', 'Method: transferFrom', and 'Inputs:'. It also shows the 'Raw Transaction', 'Receipt', and 'Error Details'.

At the bottom of the interface, there are status bars: 'Showing 1 of 1, 1 completed, 0 pending, 0 failed' for the contracts and 'Showing latest 3 of 3, 3 completed, 0 pending, 0 failed' for the transactions.



Trying to verify the owner of the NFT using the ownerOf function now gives us back the address of the new owner (0xca843569e3427144cead5e4d5999a3d0ccf92b8e), indicating that the transfer of ownership was successful:

The screenshot shows the SIMBA API interface for the `ownerOf` function. The function is described as "Get a value from the contract using ownerOf". The parameters section shows a required `tokenId` of type `number` with a value of `1`. The `Execute` button has been clicked, and the response is displayed below.

Parameters

Name	Description
<code>tokenId</code>	number

`tokenId` is marked as required. The value `1` is entered in the input field.

Responses

Curl

```
curl -X 'GET' \
  'https://api.sep.dev.simbachain.com/v2/apps/SimbaNFTExample/contract/SimbaNFTExample/ownerOf/?tokenId=1' \
  -H 'accept: application/json'
```

Request URL

```
https://api.sep.dev.simbachain.com/v2/apps/SimbaNFTExample/contract/SimbaNFTExample/ownerOf/?tokenId=1
```

Server response

Code	Details
200	<p>Response body</p> <pre>{ "request_id": "02861d9e-97f6-4d0c-a27d-f1c1b7f111f6", "value": "0xca843569e3427144cead5e4d5999a3d0ccf92b8e", "state": "COMPLETED" }</pre>

We have successfully deployed an NFT contract, minted an NFT, and transferred its ownership using the SIMBA Enterprise Platform.




This walk through only highlights a small subset of the functionality provided by the ERC721 NFT standard and its extensions. More information is available on the ERC721 standard at: <https://eips.ethereum.org/EIPS/eip-721>

FILES

The NFT media and metadata file used in this walk-through can be found at:

- <https://simbachain.com/images/SIMBA-Retro-Gamer.png>
- https://simbachain.com/images/SIMBA-Retro-Gamer_metadata.json



 simbachain.com
 574-914-4446
 info@simbachain.com