Erc1155Claimer (claim Erc1155 tokens)

Contract overview

This Erc1155Claimer contract facilitates the claiming of NFTs stored within it by authorized addresses. The claiming process involves two types of events: SimpleClaimEvent and RandomClaimEvent. Each claim event is associated with a specific ERC1155 contract, and users can only claim NFTs from the same contract within the same claim event. Whitelisted addresses are required to claim NFTs, and a predetermined number of NFTs must be assigned for claiming in a specified event.

Key features of the contract include:

- Create a simple claim event where entitled users can claim a specific ERC1155 token.
- Create a random claim event where entitled users can claim a random ERC1155 token from a pool of tokens that belongs to the same smart contract.
- Allow the contract manager to add and remove the whitelist to a specific wallet to be able to claim in a specific event.
- Disable a claim event to prevent users from claiming tokens through it.

1. Functional requirements

1.1. Roles

CollectionMinter contract has 3 roles:

1. **Pause Role (PAUSER_ROLE)**: Enable and disable all the active claim events.
2. **Team Minter Role (MANAGER_ROLE)**: Can add/remove the permission for a specific wallet address to participate in a claim event anytime.
3. **Operator Role (NFTS_OPERATOR_ROLE)**: Can send Erc1155 tokens to the smart contract in order to allow users to claim them through active claim events.

1.2. Features

Erc1155Claimer has the following features:

1. Allow a wallet with the NFTS_OPERATOR_ROLE to send Erc1155 tokens to the contract that later will be able to be claimed.
2. Create simple claim events where users can claim a specific Erc1155 token.
3. Create random claim events where users can claim one or more random Erc1155 NFTs that belong to the same smart contract collection, but randomly (the random distribution is not uniform).
4. Add and remove the permission to one or more wallets to claim through a specific claim event.
5. Stop a claim event.
6. Pause and resume of active claim events.
7. Check simple and random active claim events.
8. Allow a user to check if he can claim through a specific claim event.

1.3. Use cases

1. The MANAGER_ROLE creates a new simple claim event and then the NFTS_OPERATOR_ROLE sends 10 NFTs that will be available for the claim to the smart contract. Later the MANAGER_ROLE adds the permission for 10 wallets to claim one token each.
2. There is a claim event active and the PAUSER_ROLE stops all the claim events preventing users from claiming other available NFTs.
3. For mistake a random claim event has been created. The MANAGER_ROLE disables the event in order to prevent new claims.
4. Some NFTs have not been claimed. The MANAGER_ROLE creates a simple claim event for himself in order to withdraw the NFTs stored in the contract.

2. Technical requirements

This project has been developed with **Solidity** language, using Hardhat as the development environment. **Javascript** has been the language used for testing and scripting.

In addition, **OpenZeppelin**'s libraries have been used in the project to handle features like Access Control and Reentrancy Guard issues. All the documentation related to the used libraries is available in their Github repository.

Related to this contract the involved files in the repository are the following ones in bold red:

```
├── README.md
├── contracts
│   ├── CollectionMinter.sol
│   ├── Create and sell an ERC721 token collection.pdf
│   ├── Erc1155Claimer.sol
│   ├── Erc721Collection.sol
│   ├── SimpleErc1155.sol
│   ├── SimpleErc721.sol
│   ├── SnowMarketplace.sol
│   ├── SnowTracker.sol
│   ├── gasReport.txt
│   └── linesCounter.js
├── coverage.json
├── gasReport.txt
├── hardhat.config.js
├── package-lock.json
├── package.json
├── scripts
└── test
    ├── collectionMintTest.js
    ├── erc1155ClaimerTest.js
    └── test.js
```

Inside the **./contracts folder**, Erc1155Claimer.sol contains the smart contract with the claim functionalities explained in *section 1.2* of this document. This contract has been tested through the file **erc1155ClaimerTest.js** in order to try the different features and an instance of the SimpleErc1155 contract has been instantiated in

order to provide the tokens needed for the claims. The SimpleErc1155 contract is a general contract that has been used only for testing the features of the claimer contract and it has to be considered safe (it's only an instance of an OpenZeppelin base contracts and so the safety of the contract itself is already ensured by the OpenZeppelin team).

Automated tests have been created to test this contract and can be found in the file located at **./test/erc1155ClaimerTest.js**

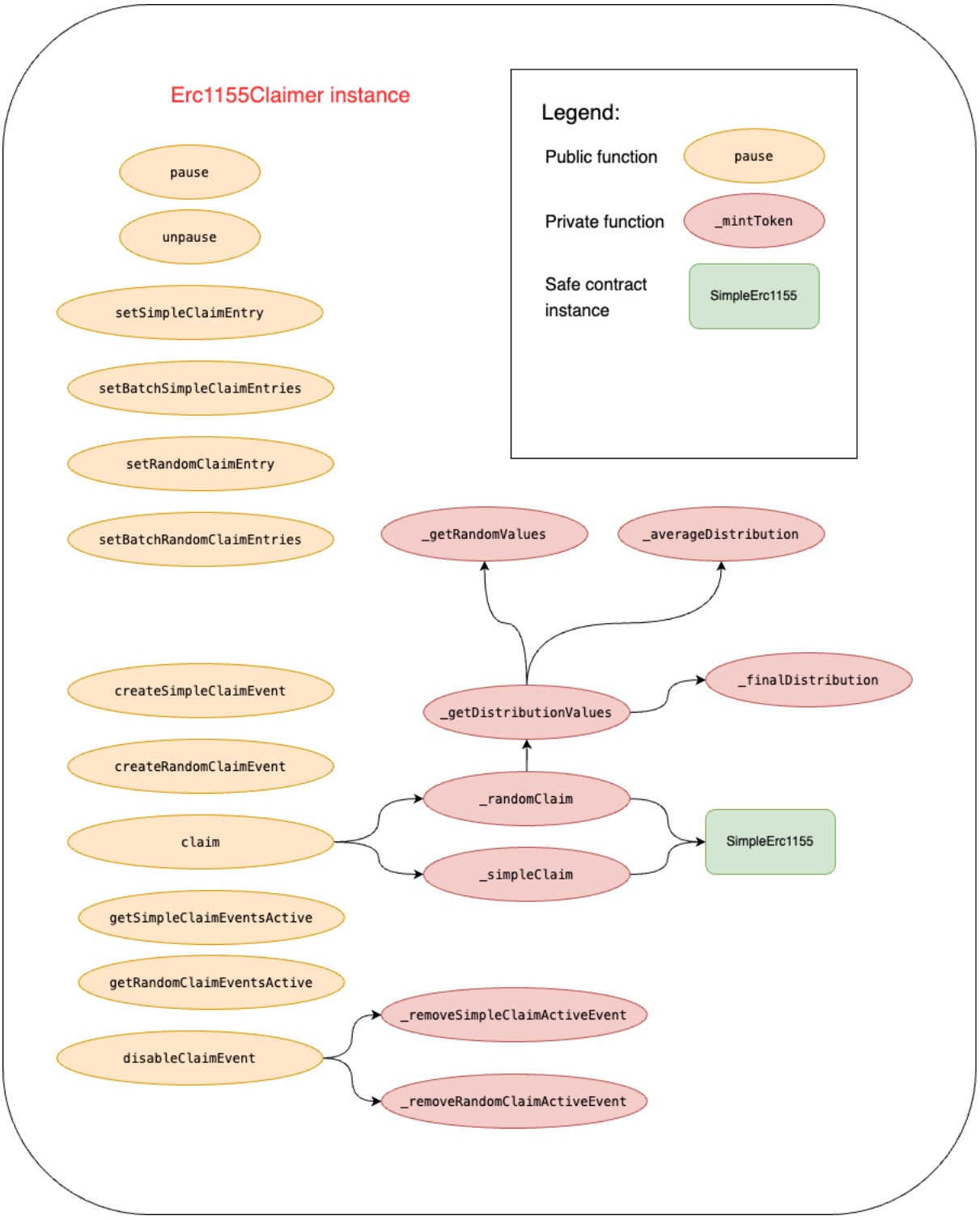
2.1. Deployment instructions

The main contract, Erc1155Claimer, can be deployed by cloning the repository using the Remix IDE and following the steps below:

1. **Connect the wallet** to the desidered chain.
2. Compile the contract with a Solidity version that is **at least 0.8.20** (this one is the suggested target version to use in order to avoid compiling issues).
3. **Deploy an instance** of the Erc1155Claimer contract (no parameters are needed to deploy an instance of this contract).

2.2. Architecture overview

The following chart provides a **general view of the Erc1155Claimer** contract structure and interactions between different functions (internal and external) and the SimpleErc1155 contract.



2.3. Contract information

This section contains useful information about the contracts used for this logic block.

2.3.1. Erc1155Claimer.sol

The Erc1155Claimer contract enables authorized addresses to claim NFTs through SimpleClaimEvent and RandomClaimEvent. Each event is linked to a specific ERC1155 contract, restricting claims to NFTs from the same contract. Whitelisted addresses need assignment of a specific number of NFTs for claiming in a given event.

This contract has no particular issues or attention that need to be taken care about and for this reason its functions behavior (parameters, result and exceptions) have been reported in NatSpec format directly in the smart contract code.

2.3.2. SimpleErc1155.sol

This contract is just an instance ready to be deployed that has been used to test the Erc1155Claimer contract. As an instance of a ready to use OpenZeppelin's contract this has been assumed as safe.

2.3.3. Emitted events

2.3.3.1. SimpleErc1155.sol events

No custom events have been defined for this contract.

2.3.3.2. CollectionMinter.sol events

The custom events triggered by this contract are the following:

- **UPDATED_SIMPLE_CLAIM_ENTRY**(_eventId, _claimableAmount): this event is triggered when is granted to claim in a simple claim event to one or more wallets. The new available amount of NFTs that can be claimed is reported into

the _claimableAmount parameter and an event is emitted for each wallet update.

- **UPDATED_RANDOM_CLAIM_ENTRY**(_eventId, _claimableAmount): this event is triggered when is granted to claim in a random claim event to one or more wallets. The new available amount of NFTs that can be claimed is reported into the _claimableAmount parameter and an event is emitted for each wallet update.
- **SIMPLE_NFTS_CLAIM**(_wallet, _eventId, _claimableAmount): this event is triggered when a user claims a _claimableAmount of NFTs in a simple claim event with _eventId ID.
- **RANDOM_NFTS_CLAIM**(_wallet, _eventId, _claimableAmount): this event is triggered when a user claims a _claimableAmount of NFTs in a random claim event with _eventId ID.
- **CLAIM_EVENT_CREATED**(_eventId, _eventType, _creator): this event is triggered when a MANAGER_ROLE creates a new claim event of type _eventType.

2.3.4. CollectionMinter.sol structs

SimpleClaimEvent { uint256 id; bool isActive; address contractAddress; uint256 tokenId; } This struct identifies a simple claim event and provide information like:

- The event ID (id)
- A boolean value that identify if the event is active and usable to claim through it (isActive)
- The Erc1155 smart contract address to which belong the type of token that can be claimed through this event (contractAddress)
- The token ID of the Erc1155 token that can be claimed through this event (tokenId)

RandomClaimEvent { uint256 id; bool isActive; address contractAddress; uint256[] tokenIds; } This struct identifies a simple claim event and provide information like:

- The event ID (id)
- A boolean value that identify if the event is active and usable to claim through it (isActive)

- The Erc1155 smart contract address to which belong the type of token that can be claimed through this event (contractAddress)
- The token IDs of the Erc1155 tokens that can be claimed through this event (tokenIds)

2.3.4.1. Smart contract functions

All functions within the smart contract are meticulously documented in the code. For a comprehensive understanding of the contract's intricacies, readers are encouraged to refer directly to the code.

Each function includes a detailed explanation of its behavior, with both technical and abstract descriptions when necessary. Additionally, the documentation outlines the inputs, expected results, and potential exceptions for each function, providing a comprehensive guide to the contract's functionality.