Hedgey Batch Streamer Documentation
Smart Contract(s): BatchStreamer.sol;

1.0 Overview & Purpose:

Background: Hedgey Finance is a group of builders that develops token tools for treasury managers of crypto companies and decentralized organizations (DAOs). DAOs in this context have a token, ERC20 standard, which represents governance, ownership or utility over the DAO and its ecosystem. Before and or after a token is launched by a DAO, they will issue tokens to their employees and investors, and most often those tokens that are granted will contain some legal lockup arrangement, where the tokens cannot be sold or claimed until after the end of this lockup period. Currently many DAOs keep these token vesting and lockup plans on legal documents and excel sheets, but are not enforceable in any way on-chain. Hedgey's StreamingNFT (and the two contracts inheriting; StreamingHedgeys and StreamingBoundHedgeys) smart contract aim to provide decentralized tools and infrastructure to support the token vesting and unlock plans for DAOs, bringing the off-chain agreements on-chain, in a transparent, trusted and decentralized tool.

As with any plan, we imagine that a DAO has a group of people, or even the entire DAO itself, that have predefined and determined the token vesting and unlock schedules for those beneficiaries, and now just need a simple and elegant tool to generate those plans on-chain and wish for a way to view them as necessary.

The BatchStreamer is a contract that acts as a simple tool in addition to the StreamingHedgeys and SteamingBoundHedgeys contracts to mint several streamingNFTs in a single transaction. This is highly useful for a team that is initiating their vesting plan for their contributors or employees and can do so in a bulk batch fashion.

Functional Requirements

The purpose is to help automate token unlocks and token vesting for DAOs. This contract helps create those unlocking and vesting plans in a single batch transaction. This contract can interface with either the StreamingHedgeys or StreamingBoundHedgeys contract, depending whether the streamingNFTs should be transferable or not.

1.1 Roles:

a. **Vesting Plan creators**

Generally the plan creator is the DAO or executive, finance, or HR team responsible for employee compensation or investor vesting. The team will establish a plan for the investor or employees, and then use the function (with the Hedgey Application) to create the on-chain vesting schedules for each beneficiary. The creators are the holders of the DAO tokens, and when setting up the on-chain plans via smart contract interaction, will provide the details of each plan to the smart contract - whereupon tokens are pulled from their wallet or DAO address, and delivered to the underlying StreamingHedgeys or StreamingBoundHedgeys contract for protocol escrow.

1.2 Functions and Features

1. **createBatch**
   The only function this contract has is the createBatch function. This function takes the address of the streamer contract (either the StreamingHedgeys or StreamingBoundHedgeys), and then the details of the recipients, token, and each recipient's vesting plan data: start date, cliff date, rate and amount.

1.3 Tokenomics & Fees

Hedgey does not have a governance token as of this product development, and so there are no tokenomics or fees associated with the tool. It is a free to use open tool built for the betterment of the crypto and web3 ecosystem.

Technical Requirements

2.0 Smart Contract Architecture Overview

The BatchStreamer.sol contract is a simple tool that allows a vesting plan creator to create multiple vesting plans for their beneficiaries in a single transaction. It leverages the power of the StreamingHedgeys or StreamingBoundHedgeys smart contracts, interfacing with them to call the createNFT function. The contract loops over the array of the inputs of the vesting plan data for each recipient, and then calls the createNFT function at the address of the StreamingHedgeys or StreamingBoundHedgeys contract.

2.1 File Imports and Dependencies

```
import './interfaces/IStreamNFT.sol';
import './libraries/TransferHelper.sol';
```

- The smart contract imports the interface necessary to interact with the StreamingHedgeys and StreamingBoundHedgeys, being the IStreamNFT.sol interface.
- The contract imports a TransferHelper.sol library that assists the contract transferring tokens from the msg.sender, and then increasing the allowance of the StreamingHedgeys contract so that it can pull the necessary tokens into the contract and mint each recipient a streaming NFT.

2.2 Global Variables

There are no global variables.

2.3 Write Functions

A streaming / vesting NFT is defined by the following parameters:

1. Recipient
2. Token address
3. The total amount of tokens locked inside the NFT
4. The start date when the tokens begin to vest or unlock
5. An optional cliff date, before which tokens cannot be redeemed
6. The rate per second which the tokens linearly vest / unlock

Thus the BatchStreamer inputs an array of these values (except token) to batch create several of these vesting NFTs at the same time. Each index in the array, as detailed below, represents a single vesting NFT, so it is important to ensure that the index of each item within each array matches perfectly together, ie the inputs at index 3 in each of the arrays correspond to a single vesting NFT, and are thus critical to keep aligned. The arrays also must be of the same length or the function will fail.

There are two versions of the external createBatch function:

createBatch (version a)

```
function createBatch(
    address streamer,
    address[] memory recipients,
    address token,
    uint256[] memory amounts,
    uint256[] memory starts,
    uint256[] memory cliffs,
    uint256[] memory rates
  ) external {
```

1. **Address streamer**: the address of the StreamingNFT contract; either the StreamingHedgeys or StreamingBoundHedgeys
2. **Address[] memory recipients**: the array of addresses of each recipient who will receive a streaming NFT
3. **Address token**: the address of the ERC20 token that will be locked and vesting in the NFTs
4. **Uint256[] memory amounts**: the array of uint256 amount that represents the total amount of tokens to be locked for each recipient.
5. **Uint256[] memory starts**: the array of unix time stamp dates that record the start date when the vesting stream begins
6. **Uint256[] memory cliffs**: the array of unix time stamp dates that record the cliff date for each vesting stream
7. **Uint256[] memory rates**: the array of rates of how the vesting tokens unlock. Each rate defines a per second rate in which tokens

Type 2 createBatch

```
function createBatch(
    address streamer,
    address[] memory recipients,
    address token,
    uint256[] memory amounts,
    uint256[] memory starts,
    uint256[] memory cliffs,
    uint256[] memory rates,
    uint256 mintType
  ) external {
```

The only difference between this function and the first, is the additional **uint256 mintType** field. This is an internal field that is called when using the Hedgey front end application. The field is picked up in an event and stored in the Hedgey database, and used for certain special visualizations on the front end. It is simply an identifier that Hedgey maintains internally for this extra usage, and has no technical or mechanical impact to the functionality of the smart contract itself.