Vesting Escrow Wallet

- **Introduced in:** 2.1.0
- **Contract name:** VestingEscrowWallet.sol
- **Type:** Wallet Module
- **Associated LucidChart:** https://www.lucidchart.com/documents/edit/ecf2a5c9-52f1-4f20-a1f2-ae6c1742490f/0?shared=true&

How it works

This module will allow approved staff to create token (ST) vesting schedule for employees and/or affiliates so that tokens get delivered to their wallets as contractually defined.
Admin can send tokens to and then select the address that would be able to withdraw them according to their specific vesting schedule.

Key functionalities (as defined in the Smart Contract)

Initialization

This module should be configured by the treasury wallet address. You will need to call the `configure` function of this contract during creation. Note that the treasury wallet can be changed later by the owner.

```
/**
 * @notice Used to initialize the treasury wallet address
 * @param _treasuryWallet Address of the treasury wallet
 */
 function configure(address _treasuryWallet) public onlyFactor
y {
```

Managing schedules

These functions allows for the admin/issuer to add a new schedule, modify or revoke existing schedule at any time using the following functions:

Add Schedule

Summary: This function is used for creating vesting schedules for each beneficiary.

```
/**
 * @notice Adds vesting schedules for each of the beneficiar
y's address
 * @param _beneficiary Address of the beneficiary for whom it
is scheduled
 * @param _templateName Name of the template that will be crea
ted
 * @param _numberOfTokens Total number of tokens for created s
chedule
 * @param _duration Duration of the created vesting schedule
 * @param _frequency Frequency of the created vesting schedule
 * @param _startTime Start time of the created vesting schedul
e
 */
function addSchedule(
    address _beneficiary,
    bytes32 _templateName,
    uint256 _numberOfTokens,
    uint256 _duration,
    uint256 _frequency,
    uint256 _startTime
)
    external
    withPerm(ADMIN)
{
```

Required checks

- `require(_beneficiary != address(0), "Invalid address")` beneficiary address shouldn't be empty
- `require(_name != bytes32(0), "Invalid name")` template name shouldn't be empty
- `require(!_isTemplateExists(_name), "Already exists")` template with an appropriate name shouldn't be created
- `require(_numberOfTokens > 0, "Zero amount")` number of tokens should be positive value
- `require(_duration % _frequency == 0, "Invalid frequency")` duration should be divided by frequency without a remainder
- `require(_numberOfTokens % periodCount == 0)` number of tokens should be divided by count of periods without a remainder
- `require(amountPerPeriod % ISecurityToken(securityToken).granularity() == 0, "Invalid granularity")` amount per period should be divided by token granularity without a remainder
- `Schedule` with an appropriate `template name` shouldn't be added to the beneficiary address
- `require(_startTime >= now, "Date in the past")` date shouldn't be in the past

Adding Schedule From Template

Summary: This function is for adding vesting schedules from template for each beneficiary.

```
/**
 * @notice Adds vesting schedules from template for the benefi
ciary
 * @param _beneficiary Address of the beneficiary for whom it
is scheduled
 * @param _templateName Name of the exists template
 * @param _startTime Start time of the created vesting schedul
e
```

```
 */
function addScheduleFromTemplate(address _beneficiary, bytes32
_templateName, uint256 _startTime) external withPerm(ADMIN) {
```

Required checks

- `require(_beneficiary != address(0), "Invalid address")` beneficiary address shouldn't be empty
- `require(_isTemplateExists(_templateName), "Template not found")` template should be already created
- `Schedule` with an appropriate `template name` shouldn't be added to the beneficiary address
- `require(_startTime >= now, "Date in the past")` date shouldn't be in the past

Modify Schedule

Summary: This function is used to modify vesting schedules for each beneficiary. Note that only the start time for schedule can be changed. If you need to modify other fields you have to remove and then add a schedule.

```
/**
 * @notice Modifies vesting schedules for each of the benefici
ary
 * @param _beneficiary Address of the beneficiary for whom it
is modified
 * @param _templateName Name of the template was used for sche
dule creation
 * @param _startTime Start time of the created vesting schedul
e
 */
function modifySchedule(address _beneficiary, bytes32 _templat
eName, uint256 _startTime) public withPerm(ADMIN) {
```

Required checks

- `require(_beneficiary != address(0), "Invalid address")` beneficiary address shouldn't be empty
- `Schedule` with the given `template name` should be already added to the beneficiary address
- `require(_startTime >= now, "Date in the past")` date shouldn't be in the past
- `require(now < schedule.startTime, "Schedule started")` schedule shouldn't be started

Revoking schedule

Summary: This function is used to revoke a beneficiary's schedule.

```
/**
 * @notice Revokes vesting schedule with given template name for given beneficiary
 * @param _beneficiary Address of the beneficiary for whom it is revoked
 * @param _templateName Name of the template was used for schedule creation
 */
function revokeSchedule(address _beneficiary, bytes32 _templateName) external withPerm(ADMIN) {
```

Required checks

- `require(_beneficiary != address(0), "Invalid address")` beneficiary address shouldn't be empty
- `Schedule` with the given `template name` should be already added to the beneficiary address

Revoke all schedules

Summary: This function is used to revoke all of the beneficiaries schedules.

```
/**
 * @notice Revokes all vesting schedules for given beneficiar
y's address
 * @param _beneficiary Address of the beneficiary for whom all
schedules will be revoked
 */
function revokeAllSchedules(address _beneficiary) public withP
erm(ADMIN) {
```

Required checks

- `require(_beneficiary != address(0), "Invalid address")` beneficiary address shouldn't be empty

Get schedule data

Summary: This function is called to return a specific beneficiary's schedule.

```
/**
 * @notice Returns beneficiary's schedule created using templa
te name
 * @param _beneficiary Address of the beneficiary who will rec
eive tokens
 * @param _templateName Name of the template was used for sche
dule creation
 * @return beneficiary's schedule data (numberOfTokens, durati
on, frequency, startTime, claimedTokens, State)
 */
function getSchedule(address _beneficiary, bytes32 _templateNa
me) external view returns(uint256, uint256, uint256, uint256,
uint256, State) {
```

Getting Template Names

Summary: This function is used to return a list of template names as well as for which have been used for schedule creation for some beneficiary.

```
/**
 * @notice Returns list of the template names for given benefi
ciary's address
 * @param _beneficiary Address of the beneficiary
 * @return List of the template names that were used for sched
ule creation
 */
function getTemplateNames(address _beneficiary) external view
returns(bytes32[]) {
```

Required checks

- `require(_beneficiary != address(0), "Invalid address")` beneficiary address shouldn’t be empty

Get schedule count

Summary: This function is used to return the count of a beneficiary’s schedules.

```
/**
 * @notice Returns count of the schedules were created for giv
en beneficiary
 * @param _beneficiary Address of the beneficiary
 * @return Count of beneficiary's schedules
 */
function getScheduleCount(address _beneficiary) external view
returns(uint256) {
```

Required checks

- `require(_beneficiary != address(0), "Invalid address")` beneficiary address shouldn’t be empty

Managing templates

The Admin can add a new template or delete an existing template (if template isn’t used by any schedule) at any time using the following functions:

Adding a Template

Summary: This function is used to add a template.
The template inputs needed are:

- Name
- Number of Tokens
- Vesting Duration
- Vesting Frequency

```
/**
 * @notice Adds template that can be used for creating schedul
e
 * @param _name Name of the template will be created
 * @param _numberOfTokens Number of tokens that should be assi
gned to schedule
 * @param _duration Duration of the vesting schedule
 * @param _frequency Frequency of the vesting schedule
 */
function addTemplate(bytes32 _name, uint256 _numberOfTokens, u
int256 _duration,
uint256 _frequency) external withPerm(ADMIN) {
```

Required checks

- `require(_name != bytes32(0), "Invalid name")` template name shouldn’t be empty

- `require(!_isTemplateExists(_name), "Already exists")` template with an appropriate name shouldn’t be created
- `require(_numberOfTokens > 0, "Zero amount")` number of tokens should be positive value
- `require(_duration % _frequency == 0, "Invalid frequency")` duration should be divided by frequency without a remainder
- `require(_numberOfTokens % periodCount == 0)` number of tokens should be divided by count of periods without a remainder
- `require(amountPerPeriod % ISecurityToken(securityToken).granularity() == 0, "Invalid granularity")` amount per period should be divided by token granularity without a remainder

---

Removing template

Summary: This function is used to remove templates and their respective name.

Note: Please take into consideration that the template can't be removed if at least one schedule already uses it.

```
/**
 * @notice Removes template with a given name
 * @param _name Name of the template that will be removed
 */
function removeTemplate(bytes32 _name) external withPerm(ADMIN) {
```

Required checks

- `require(_isTemplateExists(_templateName), "Template not found")` template should be already created
- `require(templateToUsers[_name].length == 0, "Template is used")` template shouldn’t be used by any schedules

Get template count

Summary: This function returns the count of the templates and issuer has.

```
/**
 * @notice Returns count of the templates those can be used fo
r creating schedule
 * @return Count of the templates
 */
function getTemplateCount() external view returns(uint256) {
```

Get template names

Summary: This function allows the issuer to get a list of the template names that they have set up.

```
/**
 * @notice Gets the list of the template names those can be us
ed for creating schedule
 * @return bytes32 Array of all template names were created
 */
function getAllTemplateNames() external view returns(bytes32
[]) {
```

Depositing/withdrawing tokens

An admin/issuer can deposit tokens to wallet and withdraw tokens from it. Depositing can be done by any delegate using account with tokens. On the other hand when tokens gets withdraw from the contract, `receiver` always be the treasury wallet.

Change Treasury Wallet

Summary: This function is used to change the treasury wallet address. Please note that only the issuer can change treasury wallet.

```
/**
 * @notice Used to change the treasury wallet address
 * @param _newTreasuryWallet Address of the treasury wallet
 */
function changeTreasuryWallet(address _newTreasuryWallet) publ
ic onlyOwner {
```

Required checks

- `require(_newTreasuryWallet != address(0))` treasury wallet address shouldn't be empty

Deposit Tokens

Summary: This function used for the issuer to deposit tokens from their treasury.

```
/**
 * @notice Used to deposit tokens from treasury wallet to the
vesting escrow wallet
 * @param _numberOfTokens Number of tokens that should be depo
sited
 */
function depositTokens(uint256 _numberOfTokens) external withP
erm(ADMIN) {
```

Required checks

- `require(_numberOfTokens > 0, "Should be > 0")`number of tokens should be a positive value

Send To Treasury

Summary: This function is allows the issuer to send the unassigned tokens to their treasury.

```
/**
 * @notice Sends unassigned tokens to the treasury wallet
 * @param _amount Amount of tokens that should be send to the treasury wallet
 */
function sendToTreasury(uint256 _amount) external withPerm(ADMIN) {
```

Required checks

- `require(_amount > 0, "Amount cannot be zero")` amount should be a positive value
- `require(_amount <= unassignedTokens, "Amount is greater than unassigned tokens")` Amount shouldn't be greater than unassigned tokens

Push Available Tokens

Summary: This function allows the issuer to push available tokens to the respective beneficiaries.

```
/**
 * @notice Pushes available tokens to the beneficiary's address
 * @param _beneficiary Address of the beneficiary who will receive tokens
 */
function pushAvailableTokens(address _beneficiary) public withPerm(ADMIN) {
```

Pull Available Tokens

Summary: This function allows the issuer to withdraw available tokens by the beneficiary.

```
/**
 * @notice Used to withdraw available tokens by beneficiary
 */
function pullAvailableTokens() external {
```

Multi-operations

These functions allow the admin/issuer to add a new schedule, modify or revoke existing schedule and send available tokens for few employees and/or affiliates at any time using the following functions:

Push Available Tokens Multi

Summary: This function allows the issuer to bulk send available tokens for each beneficiaries.

```
/**
 * @notice Used to bulk send available tokens for each of the
beneficiaries
 * @param _fromIndex Start index of array of beneficiary's add
resses
 * @param _toIndex End index of array of beneficiary's address
es
 */
function pushAvailableTokensMulti(uint256 _fromIndex, uint256
_toIndex) external withPerm(ADMIN) {
```

Required checks

- `require(_toIndex <= beneficiaries.length - 1, "Array out of bound")` _toIndex should be within the bounds of the beneficiary array

---

Add Schedule Multi

Summary: This function is used to bulk add respective vesting schedules for each of the beneficiaries.

```
/**
 * @notice Used to bulk add vesting schedules for each of bene
ficiary
 * @param _beneficiaries Array of the beneficiary's addresses
 * @param _templateNames Array of the template names
 * @param _numberOfTokens Array of number of tokens should be
assigned to schedules
 * @param _durations Array of the vesting duration
 * @param _frequencies Array of the vesting frequency
 * @param _startTimes Array of the vesting start time
 */
function addScheduleMulti(
    address[] _beneficiaries,
    bytes32[] _templateNames,
    uint256[] _numberOfTokens,
    uint256[] _durations,
    uint256[] _frequencies,
    uint256[] _startTimes
)
    public
    withPerm(ADMIN)
{
```

Required checks

- All arrays should have the same length

Add Schedule From Template Multi

Summary: This function is used to bulk add the vesting schedules from template for each of the beneficiaries.

```
/**
 * @notice Used to bulk add vesting schedules from template fo
r each of beneficiary
 * @param _beneficiaries Array of beneficiary's addresses
 * @param _templateNames Array of the template names were used
for schedule creation
 * @param _startTimes Array of the vesting start time
 */
function addScheduleFromTemplateMulti(address[] _beneficiarie
s, bytes32[] _templateNames, uint256[] _startTimes) external w
ithPerm(ADMIN) {
```

Required checks

- All arrays should have the same length

Revoking Schedules Multi

Summary: This function is used to bulk revoke vesting schedules for each of the beneficiaries.

```
/**
 * @notice Used to bulk revoke vesting schedules for each of t
he beneficiaries
 * @param _beneficiaries Array of the beneficiary's addresses
 */
```

```
function revokeSchedulesMulti(address[] _beneficiaries) extern
al withPerm(ADMIN) {
```

Modify Schedule Multi

Summary: This function is used to bulk modify vesting schedules for each of beneficiaries.

```
/**
 * @notice Used to bulk modify vesting schedules for each of t
he beneficiaries
 * @param _beneficiaries Array of the beneficiary's addresses
 * @param _templateNames Array of the template names
 * @param _startTimes Array of the vesting start time
 */
function modifyScheduleMulti(
    address[] _beneficiaries,
    bytes32[] _templateNames,
    uint256[] _startTimes
)
    public
    withPerm(ADMIN)
{
```

Required checks

- All arrays should have the same length

Special considerations / notes

None

Troubleshooting / FAQs

None

Know Issues / bugs

None

Changelog

- Updated the signature of sendToTreasury