Dividend Checkpoint Module (Includes: ERC20DividendCheckpoint EtherDividendCheckpoint)

- **Introduced in:** 1.3.0
- **Last updated:** 2.0.0
- **Contract name:** DividendCheckpointModule.sol
- **Compatible ST Protocol version range:** TBD
- **Type:** Checkpoint Module
- **Github Repo:** https://github.com/PolymathNetwork/polymath-core/blob/master/contracts/modules/Checkpoint/DividendCheckpoint.sol

How it works

Dividend Checkpoint Module:
This module allows the issuer to define checkpoints at which token balances and the total supply of a token can be consistently queried. The dividends checkpoint is for dividend payment mechanisms and on-chain governance, both of which need to be able to determine token balances consistently as of a specified point in time.

DISCLAIMER:
Under certain conditions, the function pushDividendPayment

- may fail due to block gas limits.
- If the total number of investors that ever held tokens is greater than ~15,000 then
- the function may fail. If this happens investors can pull their dividends, or the Issuer
- can use pushDividendPaymentToAddresses to provide an explict address list in batches

Key functionalities (as defined in the Smart Contract)

Dividend Checkpoint Module

Get Default Excluded

Summary: This function simply returns a list of the default excluded addresses.

```
/**
* @notice Return the default excluded addresses
* @return List of excluded addresses
*/
    function getDefaultExcluded() external view returns (addre
ss[]) {
        return excluded;
    }
```

Create Checkpoint

Summary: This function allows the issuer to create a checkpoint for their respective security token and returns a checkpoint ID.

```
/**
* @notice Creates a checkpoint on the security token
* @return Checkpoint ID
*/
    function createCheckpoint() public withPerm(CHECKPOINT) re
turns (uint256) {
        return ISecurityToken(securityToken).createCheckpoint
();
    }
```

Set Default Excluded

Summary: This function allows for the issuer to clear their dividends list and to set a new list of excluded addresses used for future dividend issuances.

```
/**
* @notice Function to clear and set list of excluded addresses used for future dividends
* @param _excluded Addresses of investors
*/
    function setDefaultExcluded(address[] _excluded) public withPerm(MANAGE){
        require(_excluded.length <= EXCLUDED_ADDRESS_LIMIT, "Too many excluded addresses");
        for (uint256 j = 0; j < _excluded.length; j++) {
            require (_excluded[j] != address(0), "Invalid address");
            for (uint256 i = j + 1; i < _excluded.length; i++) {
                require (_excluded[j] != _excluded[i], "Duplicate exclude address");
            }
        }
        excluded = _excluded;
        /*solium-disable-next-line security/no-block-members*/
        emit SetDefaultExcludedAddresses(excluded, now);
    }
```

SetWithholding

Summary: This function allows the issuer to set withholding tax rates for their investors.

There are a few requirements for the function:

1. Investors list length must equal the withholding list length (so we don't have mismatched input lengths)
2. withholding tax must be less than or equal to 10**18 or else it will be marked as incorrect witholding tax.

```
/**
* @notice set withholding tax rates for investors
* @param _investors Addresses of investors
* @param _withholding Withholding tax for individual investors (multiplied by 10**16)
*/
    function setWithholding(address[] _investors, uint256[] _withholding)
    }
```

SetWitholdingFixed

Summary: This function is allows the issuer to set the withholding tax rates for investors.

Requirement:

1. Withholding must be less than (or equal to) than 10**18 or else the function will return “Incorrect withholding tax”

```
 /**
* @notice Function to set withholding tax rates for investors
* @param _investors Addresses of investor
* @param _withholding Withholding tax for all investors (multiplied by 10**16)
*/
    function setWithholdingFixed(address[] _investors, uint256 _withholding)
    }
```

Push Dividend To Addresses

Summary: This function allows the issuer to push dividends to the provided list of addresses.

```
/**
* @notice push dividends to provided addresses
* @param _dividendIndex Dividend to push
* @param _payees Addresses to which to push the dividend
*/
    function pushDividendPaymentToAddresses(
        uint256 _dividendIndex,
        address[] _payees
    )
```

Push Dividend Payment

Summary: This function allows the issuer to push dividends to the provided list of addresses.

```
/**
* @notice push dividends using the investor list from the secu
rity token
* @param _dividendIndex Dividend to push
* @param _start Index in investor list at which to start pushi
ng dividends
* @param _iterations Number of addresses to push dividends for
*/
    function pushDividendPayment(
        uint256 _dividendIndex,
        uint256 _start,
        uint256 _iterations
    )
```

Pull Dividend Payment

Summary: This function allows investors to pull their own issued dividends.

```
/**
* @notice investors pull their own dividends
* @param _dividendIndex Dividend to pull
*/
    function pullDividendPayment(uint256 _dividendIndex)

    }
```

Pay Dividend

Summary: This internal function allows for the payment of dividends.

```
/**
* @notice paying dividends
* @param _payee Address of investor
* @param _dividend Storage with previously issued dividends
* @param _dividendIndex Dividend to pay
*/
    function _payDividend(address _payee, Dividend storage _di
vidend, uint256 _dividendIndex)
```

Reclaim Dividend

Summary: This function allows the issuer to reclaim the remaining unclaimed dividend amounts that have expired for investors.

```
/**
* @notice reclaim unclaimed dividend amounts for expired divid
ends
* @param _dividendIndex Dividend to reclaim
*/
    function reclaimDividend(uint256 _dividendIndex)
```

Calculate Dividend

Summary: This function is used to calculate the amount of dividends that are claimable.

Requirements:

1. Dividend Index must be less than the dividends length
2. Dividend storage size must be equal to the dividends index

```
/**
* @notice Calculate amount of dividends claimable
* @param _dividendIndex Dividend to calculate
* @param _payee Affected investor address
* @return claim, withheld amounts
*/
    function calculateDividend(uint256 _dividendIndex, address
_payee)
  }
```

Get Dividend Index

Summary: This function returns the index according to the inputted checkpoint ID.

```
/**
* @notice Get index from checkpoint id
* @param _checkpointId Checkpoint id to query
* @return uint256[]
*/
    function getDividendIndex(uint256 _checkpointId) public vi
ew returns(uint256[])
```

Withdraw Withholding

Summary: This function allows the issuer to withdraw withheld tax from the dividend index.

```
/**
* @notice withdraw withheld tax
* @param _dividendIndex Dividend to withdraw from
*/
    function withdrawWithholding(uint256 _dividendIndex)
```

Get Permissions

Summary: This function returns the permissions flag that are associated for the dividend checkpoint module.

```
/**
* @notice flags that are associated with this module
* @return bytes32 array
*/
    function getPermissions()
```

Changelog

- **Added:** Applied proxy pattern to Dividends modules

Ether Dividend Checkpoint Module

- **Introduced in:** 1.3.0
- **Last updated:** 2.0.0
- **Contract(s) name:** EtherDividendCheckpoint.sol, EtherDividendCheckpoint.sol
- **Compatible ST Protocol version range:** TBD
- **Type:** Checkpoint Module

- **Github Repo:** https://github.com/PolymathNetwork/polymath-core/blob/master/contracts/modules/Checkpoint/EtherDividendCheckpoint.sol

How it works

Summary: This checkpoint module for issuing ether dividends to investors.

Key functionalities (as defined in the Smart Contract)

Create Dividend

Summary: This function allows the issuer to create a dividend and a corresponding checkpoint for that dividend. It requires a global list of excluded addresses.

```
/**
* @notice Creates a dividend and checkpoint for the dividend
* @param _maturity Time from which dividend can be paid
* @param _expiry Time until dividend can no longer be paid, an
d can be reclaimed by issuer
* @param _name Name/title for identification
*/
    function createDividend(uint256 _maturity, uint256 _expir
y, bytes32 _name)
  }
```

Create Dividend With Checkpoint

Summary: This function allows the issuer to create a dividend with a provided checkpoint. This function also requires a global list of excluded addresses.

```
/**
* @notice Creates a dividend with a provided checkpoint
* @param _maturity Time from which dividend can be paid
* @param _expiry Time until dividend can no longer be paid, and can be reclaimed by issuer
* @param _checkpointId Id of the checkpoint from which to issue dividend
* @param _name Name/title for identification
*/
    function createDividendWithCheckpoint(
        uint256 _maturity,
        uint256 _expiry,
        uint256 _checkpointId,
        bytes32 _name
    )
```

Create Dividend With Exclusions

Summary: This function is used to create a dividend and checkpoint for the dividend and also allows the issuer to define a specific list of explicitly excluded addresses.

```
/**
* @notice Creates a dividend and checkpoint for the dividend, specifying explicit excluded addresses
* @param _maturity Time from which dividend can be paid
* @param _expiry Time until dividend can no longer be paid, and can be reclaimed by issuer
* @param _excluded List of addresses to exclude
* @param _name Name/title for identification
*/
    function createDividendWithExclusions(
```

```
        uint256 _maturity,
        uint256 _expiry,
        address[] _excluded,
        bytes32 _name
    )
```

Create Dividend With Checkpoint And Exclusions

Summary: This function is used to create a dividend with a provided checkpoint and also allows the issuer to define a specific list of explicitly excluded addresses.

Function Requirements:

1. Excluded address list needs to be less than or equal to the excluded address list limit
2. Dividend expiry must be greater than the maturity ("Expiry is before maturity")
3. Dividend expiry date must be greater than the present moment ("Expiry can’t be in the past")
4. Dividend sent must be greater than 0. ("No dividend sent")
5. The checkpointId must be less than or equal to the ISecurityToken(securityToken).currentCheckpointId())
6. Name cannot be 0
7. The zero address cannot be included in the excluded addresses list ("Invalid address")
8. Cannot dupe the system with excluded address: !dividends[dividendIndex].dividendExcluded[_excluded[j]]

```
/**
* @notice Creates a dividend with a provided checkpoint, speci
fying explicit excluded addresses
* @param _maturity Time from which dividend can be paid
* @param _expiry Time until dividend can no longer be paid, an
d can be reclaimed by issuer
* @param _checkpointId Id of the checkpoint from which to issu
e dividend
* @param _excluded List of addresses to exclude
```

```
* @param _name Name/title for identification
*/
    function _createDividendWithCheckpointAndExclusions(
        uint256 _maturity,
        uint256 _expiry,
        uint256 _checkpointId,
        address[] _excluded,
        bytes32 _name
    )
    }
```

Pay Dividend

Summary: This internal function is used to pay out the dividends to investors.

```
/**
* @notice Internal function for paying dividends
* @param _payee address of investor
* @param _dividend storage with previously issued dividends
* @param _dividendIndex Dividend to pay
*/
    function _payDividend(address _payee, Dividend storage _di
vidend, uint256 _dividendIndex)

}
```

Reclaim Dividend

Summary: This function allows the issuer to have the ability to reclaim remaining unclaimed dividend amounts, for expired investor dividends.

Function Requirements:

1. _dividendIndex must be less than the dividends.length ("Incorrect dividend index")
2. Current time period must be great than or equal to the dividends[_dividendIndex].expiry time ("Dividend expiry is in the future")

```
/**
* @notice Issuer can reclaim remaining unclaimed dividend amou
nts, for expired dividends
* @param _dividendIndex Dividend to reclaim
*/
    function reclaimDividend(uint256 _dividendIndex)
}
```

WithdrawWithholding

Summary: This function allows the issuer to withdraw withheld tax.
Requirements:

1. The dividendIndex must be less than the dividends.length ("Incorrect dividend index")
2. The Dividend storage dividend must equal the dividends[_dividendIndex]

```
/**
* @notice withdraw withheld tax
* @param _dividendIndex Dividend to withdraw from
*/
    function withdrawWithholding(uint256 _dividendIndex)
  }
```

Changlog

- **Changed** version of `EtherDividendCheckpointFactory` from `1.0.0` to `2.1.0`.

ERC20 Dividend Checkpoint Module

- **Introduced in:** 1.3.0
- **Last updated:** 2.0.0
- **Contract(s) name:** ERC20DividendCheckpoint.sol
- **Compatible ST Protocol version range:** TBD
- **Type:** Checkpoint Module
- **Github Repo:** https://github.com/PolymathNetwork/polymath-core/blob/master/contracts/modules/Checkpoint/ERC20DividendCheckpoint.sol

How it works

Summary: Checkpoint module for issuing ERC20 dividends. The function works by having a mapping to token addresses for each dividend.

Key functionalities (as defined in the Smart Contract)

CreateDividend

Summary: This function allows the issuer to create a dividend and a corresponding checkpoint for that dividend. It requires a global list of excluded addresses.

```
/**
* @notice Creates a dividend and checkpoint for the dividend
* @param _maturity Time from which dividend can be paid
* @param _expiry Time until dividend can no longer be paid, and can be reclaimed by issuer
* @param _token Address of ERC20 token in which dividend is to be denominated
* @param _amount Amount of specified token for dividend
* @param _name Name/Title for identification
*/
    function createDividend(
```

```
        uint256 _maturity,
        uint256 _expiry,
        address _token,
        uint256 _amount,
        bytes32 _name
    )
```

CreateDividendCheckpoint

Summary: This function allows the issuer to create a dividend with a provided checkpoint. This function also requires a global list of excluded addresses.

```
/**
* @notice Creates a dividend with a provided checkpoint
* @param _maturity Time from which dividend can be paid
* @param _expiry Time until dividend can no longer be paid, an
d can be reclaimed by issuer
* @param _token Address of ERC20 token in which dividend is to
be denominated
* @param _amount Amount of specified token for dividend
* @param _checkpointId Checkpoint id from which to create divi
dends
* @param _name Name/Title for identification
*/
    function createDividendWithCheckpoint(
        uint256 _maturity,
        uint256 _expiry,
        address _token,
        uint256 _amount,
        uint256 _checkpointId,
        bytes32 _name
```

```
    )
```

CreateDividendWithExclusions

Summary: This function is used to create a dividend and checkpoint for the dividend and also allows the issuer to define a specific list of explicitly excluded addresses.

```
/**
* @notice Creates a dividend and checkpoint for the dividend
* @param _maturity Time from which dividend can be paid
* @param _expiry Time until dividend can no longer be paid, an
d can be reclaimed by issuer
* @param _token Address of ERC20 token in which dividend is to
be denominated
* @param _amount Amount of specified token for dividend
* @param _excluded List of addresses to exclude
* @param _name Name/Title for identification
*/
    function createDividendWithExclusions(
        uint256 _maturity,
        uint256 _expiry,
        address _token,
        uint256 _amount,
        address[] _excluded,
        bytes32 _name
    )
```

CreateDividendCheckpointAndExclusions

Summary: This function allows the issuer to create a dividend with a provided checkpoint ID.

Important Function Requirements:

1. Excluded address list needs to be less than or equal to the excluded address list limit
2. Dividend expiry must be greater than the maturity ("Expiry is before maturity")
3. Dividend expiry date must be greater than the present moment ("Expiry can’t be in the past")
4. Dividend sent must be greater than 0. ("No dividend sent")
5. Token cannot be the zero address(0) ("Invalid token")
6. checkpointId must be less than or equal to the securityTokenInstance.currentCheckpointId()("Invalid checkpoint")
7. IERC20(_token).transferFrom(msg.sender, address(this), _amount) needs to have enough allowance to make a transfer ("insufficient allowance")
8. The name cannot be 0.
9. Cannot dupe the system with excluded address: !dividends[dividendIndex].dividendExcluded[_excluded[j]], "duped exclude address");

```
/**
* @notice Creates a dividend with a provided checkpoint
* @param _maturity Time from which dividend can be paid
* @param _expiry Time until dividend can no longer be paid, and can be reclaimed by issuer
* @param _token Address of ERC20 token in which dividend is to be denominated
* @param _amount Amount of specified token for dividend
* @param _checkpointId Checkpoint id from which to create dividends
* @param _excluded List of addresses to exclude
* @param _name Name/Title for identification
*/
    function createDividendWithCheckpointAndExclusions(
        uint256 _maturity,
        uint256 _expiry,
```

```
        address _token,
        uint256 _amount,
        uint256 _checkpointId,
        address[] _excluded,
        bytes32 _name
    )
```

_emitERC20DividendDepositedEvent

Summary: This function is used to emit the ERC20DividendDeposited Event. It is separated into a different function as a workaround for the stack too deep error.

```
/**
* @notice Emits the ERC20DividendDeposited event.
*/
    function _emitERC20DividendDepositedEvent(
        uint256 _checkpointId,
        uint256 _maturity,
        uint256 _expiry,
        address _token,
        uint256 _amount,
        uint256 currentSupply,
        uint256 dividendIndex,
        bytes32 _name
    )
```

_payDividend

Summary: This internal function is used for paying out dividends.

```
/**
* @notice Internal function for paying dividends
* @param _payee Address of investor
```

```
* @param _dividend Storage with previously issued dividends
* @param _dividendIndex Dividend to pay
*/
    function _payDividend(address _payee, Dividend storage _di
vidend, uint256 _dividendIndex)


  }
```

reclaimDividend

Summary: This function is used by the issuer in order to reclaim remaining unclaimed dividend amounts, specifically for expired dividends.

```
/**
* @notice reclaim remaining unclaimed dividend amounts for exp
ired dividends
* @param _dividendIndex Dividend to reclaim
*/
    function reclaimDividend(uint256 _dividendIndex)
  }
```

withdrawWithholding

Summary: This function allows issuer to withdraw withheld tax.

```
**/
* @notice withdraw withheld tax
* @param _dividendIndex Dividend to withdraw from
*/
    function withdrawWithholding(uint256 _dividendIndex)
  }
```

Changlog

- **Changed** version of `ERC20DividendCheckpointFactory` from `1.0.0` to `2.1.0`.