YieldContract.md

# YieldContract

TO NOTE Store collateral and provide interest MXX or burn MXX
Interest (contractFee, penaltyFee etc) is always represented 10 power 6 times the actual value
Note that only 4 decimal precision is allowed for interest
If interest is 5%, then value to input is 0.05 * 10 pow 6 = 5000
mFactor or mintFactor is represented 10 power 18 times the actual value.
If value of 1 ETH is 380 USD, then mFactor of ETH is (380 * (10 power 18))
Collateral should always be in its lowest denomination (based on the coin or Token)
If collateral is 6 USDT, then value is 6 * (10 power 6) as USDT supports 6 decimals
startTime and endTime are represented in Unix time
tenure for contract is represented in days (90, 180, 270) etc
mxxToBeMinted or mxxToBeMinted is always in its lowest denomination (8 decimals)
For e.g if mxxToBeMinted = 6 MXX, then actual value is 6 * (10 power 8)

### `constructor(address _mxxAddress, uint256 _mxxmFactor)` (public)

CONSTRUCTOR FUNCTION

### `getIndex(address _inputAddress, address[] _inputAddressList) → uint32 index` (internal)

This function will check the array for an element and retun the index

### `emitContractStatus(bytes32 _contractId)` (internal)

This function will emit contract status

### `getContractStatusKey(enum YieldContract.Status _status) → string` (internal)

- Fails with improper input Access Control: This contract or derived contract

This function will return the key of the contract status in string

### `setInterest(uint16 _tenure, uint256 _interestRate) → bool` (public)

This function will set interest rate for the tenure in days

### `setParamType(enum YieldContract.paramType _parameter, uint256 _value) → bool` (public)

This function will set value based on paramType (contract fee, mxx penalty fee, max penalty fee, total allocated)

### `addValidERC20(address _ERC20Address, uint256 _mFactor) → bool` (public)

- Access control: Only Owner

Adds a valid ERC20 address into the contract

### `addValidERC20List(address[] _ERC20AddressList, uint256[] _mFactorList) → bool` (public)

- The length of _ERC20AddressList and _mFactorList must be the same
- Access control: Only Owner

Adds a list of valid ERC20 addresses into the contract

### `removeValidERC20(address _ERC20Address) → bool` (public)

- Access control: Only Owner

Removes a valid ERC20 addresses from the contract

### `delistValidERC20(address _ERC20Address) → bool` (public)

- Access control: Only Owner

Delists ERC20 address to prevent adding new yield contracts with this ERC20 collateral

### `undelistERC20(address _ERC20Address) → bool` (public)

- Access control: Only Owner

Undelists ERC20 address to resume adding new yield contracts with this ERC20 collateral

### `updateMFactor(address _ERC20Address, uint256 _mFactor) → bool` (public)

- Access control: Only Owner

Updates the mint factor of a coin/token

### `updateMFactorList(address[] _ERC20AddressList, uint256[] _mFactorList) → bool` (public)

- Length of the 2 input arrays must be the same
- Access control: Only Owner

Updates the mint factor for list of coin(s)/token(s)

### `getNoOfValidTokens() → uint256` (public)

- Access control: Public

Returns number of valid Tokens/Coins supported

### `getSubsetValidERC20(uint32 _start, uint32 _end) → address[]` (public)

- Access control: Public

Returns subset list of valid ERC20 contracts

### `createYieldContract(address _ERC20Address, uint256 _collateral, uint16 _tenure) → bytes32` (public)

- Collateral to be input - Actual value * (10 power decimals)
- For e.g If collateral is 5 USDT (Tether) and decimal is 6, then _collateral is (5 * (10 power 6))
  Non Reentrant modifier is used to prevent re-entrancy attack
- Access control: Public

Creates a yield contract

### `earlyRedeemContract(bytes32 _contractId) → bool` (public)

- Access control: Public

Early Redeem a yield contract

### `acquireYieldContract(bytes32 _contractId) → bool` (public)

- Access control: Public

Acquire a yield contract in the open market

### `claimYieldContract(bytes32 _contractId) → bool` (public)

- Access control: Public

Claim a yield contract in the active market

### `getSubsetYieldContracts(uint32 _start, uint32 _end) → bytes32[]` (public)

This function will subset of yield contract

### `yieldContractStatus(address contractOwner, address tokenAddress, uint256 collateral, uint256 startTime, uint256 endTime, uint256 interest, uint16 tenure, uint256 mxxToBeMinted, uint256 mxxToBeBurnt, string contractStatus)`

contractOwner - The owner of the yield contract
tokenAddress - ERC20 contract address (if ETH then address(0))
collateral - Value of collateral (multiplied by 10 power 18 to handle decimals)
startTime - Start time of the yield contract (in unix timestamp)
endTime - End time of the yield contract (in unix timestamp)
interest - APY or Annual Percentage Yield (returned from tenureApyMap)
tenure - The agreement tenure in days
mxxToBeMinted - The final MXX token value to be returned to the contract owner
mxxToBeBurnt - The MXX value to be burnt from user's account when creating a contract
contractStatus - The status of a contract (can be inactive/active/openMarket/Claimed)

- An event to be emitted when a contract is created/updated