# TypeX NATGX Token

The TypeX NATGX Token contract is an ERC-20 compatible token. It allows minting/burning of tokens by multiple entities, pausing all activity, freezing of individual addresses, and a way to upgrade the contract so that bugs can be fixed or features added.

## Roles

The NATGXToken has a number of roles (addresses) which control different functionality:

* masterMinter - adds and removes minters and increases their minting allowance
* minters - create and destroy tokens
* pauser - pause the contract, which prevents all transfers, minting, and burning
* blacklister - prevent all transfers to or from a particular address, and prevents that address from minting or burning
* owner - re-assign any of the roles except for admin
* admin - upgrade the contract, and re-assign itself

TypeX will control the address of all roles except for minters, which will be controlled by the entities that TypeX elects to make minters

## ERC-20

The NATGXToken implements the standard methods of the ERC-20 interface with some changes:

* A blacklisted address will be unable to call transfer, transferFrom, or approve, and will be unable to receive tokens.
* transfer, transferFrom, and approve will fail if the contract has been paused.

## Issuing and Destroying tokens

The NATGX Token allows multiple entities to create and destroy tokens. These entities will have to be members of TypeX, and will be vetted by TypeX before they are allowed to create new tokens. TypeX will not mint any tokens itself, it will approve members to mint and burn tokens.

Each minter has a mintingAllowance, which TypeX configures. The mintingAllowance is how many tokens that minter may issue, and as a minter issues tokens, its mintingAllowance declines. TypeX will periodically reset the mintingAllowance as long as a minter remains in good standing with TypeX and maintains adequate reserves for the tokens it has issued. The mintingAllowance is to limit the damage if any particular minter is compromised.

### Adding Minters

TypeX adds minters via the configureMinter method. When a minter is configured a mintingAllowance is specified, which is the number of tokens that address is allowed to mint. As a minter mints tokens, the mintingAllowance will decline.

* Only the masterMinter role may call configureMinter.

### Resetting Minting Allowance

The minters will need their allowance reset periodically to allow them to continue minting. When a minter’s allowance is low, TypeX can make another call to configureMinter to reset the mintingAllowance to a higher value.

### Removing Minters

TypeX removes minters via the removeMinter method. This will remove the minter from the list of minters and set its mintingAllowance to 0. Once a minter is removed it will no longer be able to mint or burn tokens.

* Only the masterMinter role may call removeMinter.

### Minting

A minter mints tokens via the mint method. The minter specifies the amount of tokens to create, and a \_to address which will own the newly created tokens. A minter may only mint an amount less than or equal to its mintingAllowance. The mintingAllowance will decrease by the amount of tokens minted, and the balance of the \_to address and totalSupply will each increase by amount.

* Only a minter may call mint.
* Minting fails when the contract is paused.
* Minting fails when the minter or \_to address is blacklisted.
* Minting emits a Mint(minter, \_to, amount) event and a Transfer(0x00, \_to, amount) event.

### Burning

A minter burns tokens via the burn method. The minter specifies the amount of tokens to burn, and the minter must have a balance greater than or equal to the amount. Burning tokens is restricted to minter addresses to avoid accidental burning of tokens by end users. A minter with a mintingAllowance of 0 is allowed to burn tokens. A minter can only burn tokens which it owns. When a minter burns tokens, its balance and the totalSupply are reduced by amount.

Burning tokens will not increase the mintingAllowance of the address doing the burning.

* Only a minter may call burn.
* Burning fails when the contract is paused.
* Burning fails when the minter is blacklisted.
* Burning emits a Burn(minter, amount) event, and a Transfer(minter, 0x00, amount) event.

## Blacklisting

Addresses can be blacklisted. A blacklisted address will be unable to transfer tokens, approve, mint, or burn tokens.

### Adding a blacklisted address

TypeX blacklists an address via the blacklist method. The specified account will be added to the blacklist.

* Only the blacklister role may call blacklist.
* Blacklisting emits a Blacklist(account) event

### Removing a blacklisted address

TypeX removes an address from the blacklist via the unblacklist method. The specified account will be removed from the blacklist.

* Only the blacklister role may call unblacklist.
* Unblacklisting emits an UnBlacklist(account) event.

## Pausing

The entire contract can be paused in case a serious bug is found or there is a serious key compromise. All transfers, minting, burning, and adding minters will be prevented while the contract is paused. Other functionality, such as modifying the blacklist, removing minters, changing roles, and upgrading will remain operational as those methods may be required to fix or mitigate the issue that caused TypeX to pause the contract.

### Pause

TypeX will pause the contract via the pause method. This method will set the paused flag to true.

* Only the pauser role may call pause.
* Pausing emits a Pause() event

### Unpause

TypeX will unpause the contract via the unpause method. This method will set the paused flag to false. All functionality will be restored when the contract is unpaused.

* Only the pauser role may call unpause.
* Unpausing emits an Unpause() event

## Upgrading

The NATGX Token uses the zeppelinos Unstructured-Storage Proxy pattern [https://docs.zeppelinos.org/docs/upgradeability\_AdminUpgradeabilityProxy.html]. [NATGXTokenV1.sol](../contracts/FiatTokenV1.sol) is the implementation, the actual token will be a Proxy contract

Note that this is how USDC appears to be deployed currently

([NATGXTokenProxy.sol](../contracts/FiatTokenProxy.sol)) which will forward all calls to NATGXToken via delegatecall. This pattern allows TypeX to upgrade the logic of any deployed tokens seamlessly.

* TypeX will upgrade the token via a call to upgradeTo or upgradeToAndCall if initialization is required for the new version.
* Only the admin role may call upgradeTo or upgradeToAndCall.

## Reassigning Roles

The roles outlined above may be reassigned. The owner role has the ability to reassign all roles (including itself) except for the admin role.

### Admin

* changeAdmin updates the admin role to a new address.
* changeAdmin may only be called by the admin role.

### Master Minter

* updateMasterMinter updates the masterMinter role to a new address.
* updateMasterMinter may only be called by the owner role.

### Pauser

* updatePauser updates the pauser role to a new address.
* updatePauser may only be called by the owner role.

### Blacklister

* updateBlacklister updates the blacklister role to a new address.
* updateBlacklister may only be called by the owner role.

### Owner

* transferOwnership updates the owner role to a new address.
* transferOwnership may only be called by the owner role.