Advanced ERC20 Token Specification

Version 1.1.6_11/24/24

Advanced ERC20 is a feature rich lightweight ERC20 implementation, designed independently from the OpenZeppelin ERC20 framework, offering enhanced flexibility and performance.

Table of Contents

Features and Implemented Requirements

Basic Features Summary

- Symbol: configurable (set on deployment)
- Name: configurable (set on deployment)
- Decimals: 18
- Initial/maximum total supply: configurable (set on deployment)
- Initial supply holder (initial holder) address: configurable (set on deployment)
- Mintable: configurable (set on deployment); revocable
- Burnable: configurable (set on deployment); revocable
- Upgradable: no
- DAO Support: supports voting delegation
- Deployment: standalone or EIP-1167 cloning

Advanced Features Summary

- Supports atomic allowance modification, resolves well-known ERC20 issue with approve (arXiv:1907.00903)
- Voting delegation and delegation on behalf via EIP-712 (like in Compound CMP token) – gives the token powerful governance capabilities by allowing holders to form voting groups by electing delegates
- Unlimited approval feature (like in 0x ZRX token) – saves gas for transfers on behalf by eliminating the need to update “unlimited” allowance value
- ERC-1363 Payable Token - ERC721-like callback execution mechanism for transfers, transfers on behalf, approvals, and restricted access mints (which are sometimes viewed as transfers from zero address);
  allows creation of smart contracts capable of executing callbacks – in response to token transfer, approval, and token minting – in a single transaction
- EIP-2612: permit - 712-signed approvals – improves user experience by allowing to use a token without having an ETH to pay gas fees
- EIP-3009: Transfer With Authorization – improves user experience by allowing to use a token without having an ETH to pay gas fees

Functional Requirements Summary

1. Fully ERC20 compliant according to “EIP-20: ERC-20 Token Standard” [1]
2. Initial token supply is minted to an initial holder address
3. Token holders are able to participate in governance protocol(s) and vote with their tokens (see below – “Voting Delegation Requirements”)
4. Implemented functional improvements: ERC-1363 Payable Token, EIP-2612 permit, and EIP-3009 Transfer With Authorization [2, 3, 4, 5]

Voting Delegation Requirements

1. Token holders posses voting power associated with their tokens

2. Token holders are able to act as delegators and to delegate their voting power to any other address – a delegate
3. Any address may become a delegate; delegates are not necessarily token owners
4. Voting power delegation doesn't affect token balances
5. It is possible to retrieve voting power of any delegate for any point in time, defined by the Ethereum block number (block height)
6. Delegators are able to revoke their voting power delegation
7. Voting delegation requirements are inspired by the Compound's COMP token; refer to Compound documentation for more information not directly covered in the requirements [8]

ERC20 Improvements Requirements

1. Support for atomic allowance modification, resolution of well-known ERC20 issue with approve [6]
2. Tokens owner is able to set ERC20 allowance to "unlimited" value – the value which is not updated when token transfer is performed (like in 0x ZRX token), effectively allowing to save gas for transfers on behalf
3. Support for ERC-1363 Payable Token – ERC721-like callback execution mechanism for transfers, transfers on behalf and approvals; allow creation of smart contracts capable of executing callbacks in response to transfer or approval in a single transaction
4. Support for EIP-2612: permit – 712-signed approvals – allow token holders to approve token transfers without having an ETH to pay gas fees
5. Support for EIP-3009: Transfer With Authorization – allow token holders to send tokens without having an ETH to pay gas fees

Implied Limitations

1. Token holders are responsible not to send their tokens to non-ERC20-compliant addresses (external and/or smart contracts) via standard ERC20 interface; an attempt to do so will succeed and tokens sent may get lost forever
2. Voting power delegation cannot be chained: an address may delegate only voting power associated with its balance, but not the voting power delegated to it
3. Token owners are delegators but not delegates by default; they should delegate their voting power to themselves and become delegates in order to use their own voting power for voting

Gas Usage

Token functions execution gas consumption depends on the token state.

Lowest gas consumption is reached for basic ERC20 functions (transfers, approvals, minting and burning) when the addresses involved don't use voting delegation. Gas consumption increases when delegation is involved.

Governance related functions (delegation and delegation on behalf) consume less gas when the balances of the delegates involved are zero, gas consumption increases when the balances are not zero.

ERC-1363 gas consumption depends on the target smart contract's callback, therefore we measure the minimum value when target callback does nothing but only event logging.

The table below summarizes gas requirements for the token operations.

Table 1. Gas usage requirements for the AdvancedERC20 Token

Function	Gas Usage		
	No Delegation	Some Delegation	Delegation Involved
ERC-20			
Transfers	61,663	94,801	141,971
Transfers on Behalf	71,614	104,749	151,919
Transfers on Behalf (Unlimited Allowance)	64,450	97,588	144,758
Approvals	48,508		
Atomic Approvals [ISBN:978-1-7281-3027-9]			
Atomic Approval Increase	48,864		
Atomic Approval Decrease	31,818		
ERC-1363			
Transfers	68,810	101,945	149,115
Transfers on Behalf	78,798	111,933	159,103
Transfers on Behalf (Unlimited Allowance)	71,637	104,772	151,942
Approvals	57,446		
EIP-2612			
Permits	79,412		
EIP-3009			
Transfers with Auth	87,671	120,818	167,989
Receptions with Auth	87,710	120,857	168,028
Mint/Burn Addons			
Function	Gas Usage		
	No Delegation		Delegation Involved
Minting	91,431		138,677

ERC-1363 like Minting (`mintAndCall`)	98,690	145,933
Burning	76,568	109,706
Burning on Behalf	86,060	119,198
Burning on Behalf (Unlimited Allowance)	78,940	112,078

Voting Delegation			
Function	Gas Usage		
	Empty Balances	One Empty Another Not	Non-empty Balances
Delegations	50,629	97,873	113,835
Delegations on Behalf	80,893	128,122	144,096

Token smart contract deployment costs 297,055 gas if deployed as EIP-1167 Clone, 3,765,389 gas when full deployment is done.

Implementation Overview

The token smart contract is a composition of reference ERC-20, ERC-1363, EIP-2612, EIP-3009, voting delegation implementations with unlimited allowance and atomic allowance improvements added. Relevant tests from the reference implementations fully adopted to guarantee no modifications in the functionality expected.

Following reference implementations were used as a source of inspiration and tests:

- Atomic allowance:
  - https://github.com/OpenZeppelin/openzeppelin-contracts
- Unlimited allowance:
  - https://github.com/0xProject/protocol
- Voting delegation:
  - https://github.com/compound-finance/compound-protocol
  - https://github.com/OpenZeppelin/openzeppelin-contracts
- ERC-1363:
  - https://github.com/vittominacori/erc1363-payable-token
- EIP-2612:
  - https://github.com/Uniswap/uniswap-v2-core
- EIP-3009:
  - https://github.com/centrehq/centre-tokens
  - https://github.com/CoinbaseStablecoin/eip-3009

Role-based Access Control (RBAC)

Advanced ERC20 is built on top of Role-based Access Control (RBAC) [9] module, more specifically its “initializable” version `InitializableAccessControl` [10].

All public functions (ERC20, ERC1363, EIP2612, EIP3009) are controlled with the feature flags and can be enabled/disabled during the token deployment and setup process.

Restricted access functions (`mint` and `burn`) are controlled by user roles/permissions and can be executed by the deployer during token deployment and setup processes.

It is possible to deploy the token without any privileged access with all or some public functions enabled (see the section below).

It is possible to revoke privileged access to the restricted functions at any time (see the section below).

Features and Roles

Features control the behavior of publicly available functions, such as token transfers and burning. In most cases enabling/disabling a feature allows to enable/disable corresponding functionality.

The table below summarizes the features defined in the Token smart contract.

Table 2. Summary of the Token smart contract features.

Feature	Bit	Description
`TRANSFERS`	0	Enables own tokens transfers, `transfer`
`TRANSFERS_ON_BEHALF`	1	Enables transfers on behalf, `transferFrom`
`UNSAFE_TRANSFERS`	2	Disables recipient ERC20-compliance check
`OWN_BURNS`	3	Enables burning own tokens, `burn`
`BURNS_ON_BEHALF`	4	Enables burning on behalf, `burn`
`DELEGATIONS`	5	Enables voting delegation, `delegate`
`DELEGATIONS_ON_BEHALF`	6	Enables voting delegation on behalf, `delegateWithAuthorization`
`ERC1363_TRANSFERS`	7	Enables ERC-1363 transfers, `transferFromAndCall`
`ERC1363_APPROVALS`	8	Enables ERC-1363 approvals, `approveAndCall`
`EIP2612_PERMITS`	9	Enables EIP-2612 permits, `permit`
`EIP3009_TRANSFERS`	10	Enables EIP-3009 transfers, `transferWithAuthorization`
`EIP3009_RECEPTIONS`	11	Enables EIP-3009 receptions, `receiveWithAuthorization`

Roles control access to smart contract functions not to be used publicly (restricted functions). These include minting, burning, etc.

The table below summarizes the roles defined in the Token smart contract.

Table 3. Summary of the Token smart contract special permissions (roles).

Role	Bit	Description
`TOKEN_CREATOR`	16	Allows minting tokens, `mint`
`TOKEN_DESTROYER`	17	Allows burning any tokens, `burn`
`ERC20_RECEIVER`	18	Disables recipient ERC20-compliance check
`ERC20_SENDER`	19	Disables ERC20-compliance check when performed by sender in `ERC20_SENDER` role
`ACCESS_MANAGER`	255	Reserved, defined in `AccessManager` smart contract, allows modifying features and roles

Public Functions

The table below summarizes public functions designed to transfer tokens between accounts, burn tokens and authorize other addresses to perform these operations on token owner behalf.

Table 4. Summary of the public functions

Function Name	Description
`transfer`	ERC20 transfer, transfers own tokens Requires the receiver to be ERC20 compliant (EOA or `ERC1363Receiver` smart contract) unless the feature `UNSAFE_TRANSFERS` is enabled
`transferFrom`	ERC20 transfer on behalf, transfers own tokens, or transfers tokens on behalf of address which approved the transfer Requires the receiver to be ERC20 compliant (EOA or `ERC1363Receiver` smart contract) unless the feature `UNSAFE_TRANSFERS` is enabled
`safeTransferFrom`	Same as `transferFrom`, but always requires the receiver to be ERC20 compliant (EOA or `ERC1363Receiver` smart contract)
`unsafeTransferFrom`	Same as `transferFrom`, but never requires the receiver to be ERC20 compliant (EOA or `ERC1363Receiver` smart contract)
`transferAndCall`	ERC-1363 family of transfer functions, expecting the receiver to be a `ERC1363Receiver` smart contract
`transferFromAndCall`	ERC-1363 family of transfer on behalf functions,

	expecting the receiver to be a `ERC1363Receiver` smart contract
`approve`	ERC20 approve, approves transfers on behalf and burning tokens on behalf of the account which invoked `approve`
`approveAndCall`	ERC-1363 family of approve functions, expecting the receiver to be a `ERC1363Spender` smart contract
`transferWithAuthorization`	EIP-3009 transfer, transfers the tokens owned by signer Requires the receiver to be ERC20 compliant (EOA or `ERC1363Receiver` smart contract) unless the feature `UNSAFE_TRANSFERS` is enabled
`receiveWithAuthorization`	EIP-3009 transfer, transfers the tokens owned by signer, can be executed only by the receiver Requires the receiver to be ERC20 compliant (EOA or `ERC1363Receiver` smart contract) unless the feature `UNSAFE_TRANSFERS` is enabled
`cancelAuthorization`	EIP-3009 cancellation, cancels the EIP-3009 request to transfer tokens, or cancels the voting delegation request
`increaseAllowance`	Increases ERC20 approval value by some amount
`decreaseAllowance`	Decreases ERC20 approval value by some amount
`permit`	EIP-2612 permit, approves transfers on behalf and burning tokens on behalf of the account which signed the permit
`delegate`	Delegates voting power of the transaction sender
`delegateWithAuthorization`	Delegates voting power on behalf of the address which signed the delegation request, EIP712 compliant
`burn`	Burns own tokens, or burns tokens on behalf of address which approved burning

Restricted Functions

The table below summarizes restricted access functions to be used by smart contract administrators and approved helpers to interact with the Token smart contract.

Table 5. Summary of the functions with restricted access

Function Name	Description

`mint`	Mints tokens
`safeMint`	Mints tokens and if the receiver is a smart contract, executes the ERC-1363 callback on the receiver
`mintAndCall`	Mints tokens and executes the ERC-1363 callback on the receiver, expecting it to be a `ERC1363Receiver` smart contract
`burn`	Burns tokens on behalf of any address, independently of if it approved operation or not

Note that the Token smart contract admin may revoke the permission to access any of the restricted functions from any address, including its own. This means that any of the restricted functions can be disabled forever.

Voting Delegation

Voting delegation is implemented similarly to Compound voting delegation, taking into account the non-fixed total supply nature of the token (total supply may decrease over time by design).

For better code readability, and in order to follow best practices and naming conventions, the names of the mappings, events and functions were changed. Table below summarizes the changes.

Table 6. Summary of the naming changes for voting delegation related functions, events and mappings.

Advanced ERC20 Name	Compound Name
`votingDelegates`	`delegates`
`votingPowerHistory`	`checkpoints`
`votingPowerHistoryLength`	`numCheckpoints`
`totalSupplyHistory`	`_totalSupplyCheckpoints`
`usedNonces`	`nonces`
`VotingPowerChanged`	`DelegateVotesChanged`
`votingPowerOf`	`getCurrentVotes`
`votingPowerAt`	`getPriorVotes`
`totalSupplyAt`	`getPriorTotalSupply`
`delegateWithAuthorization`	`delegateBySig`

Note: the `nonces` mapping has not only been renamed, but also has been functionally enhanced to support the use of random nonces instead of the sequential ones. This is

similar to EIP-3009, the same EIP-3009 `cancelAuthorization` function can be used to cancel the signed delegation on behalf.

References

1. EIP-20: ERC-20 Token Standard
   https://eips.ethereum.org/EIPS/eip-20
2. EIP-1363: ERC-1363 Payable Token
   https://eips.ethereum.org/EIPS/eip-1363
3. EIP-777: ERC777 Token Standard
   https://eips.ethereum.org/EIPS/eip-777
4. EIP-2612: permit – 712-signed approvals
   https://eips.ethereum.org/EIPS/eip-2612
5. EIP-3009: Transfer With Authorization
   https://eips.ethereum.org/EIPS/eip-3009
6. ERC20 API: An Attack Vector on Approve/TransferFrom Methods
   https://docs.google.com/document/d/1YLPtQxZu1UAvO9cZ1O2RPXBbT0mooh4DYKjA_jp-RLM/
7. 0x: An open protocol for decentralized exchange on the Ethereum blockchain
   https://0x.org/pdfs/0x_white_paper.pdf
8. Compound: Governance
   https://compound.finance/docs/governance
9. Role-based Access Control (RBAC): A shortcut to a modular and easily pluggable dapp architecture
   https://github.com/lazy-sol/access-control
10. Upgradable Role-based Access Control (U-RBAC)
    https://github.com/lazy-sol/access-control-upgradeable

Prepared by Basil Gorin on November 24, 2024