PlayX Token Specification

Overview

PlayX is a utility token designed for use within the Playnance ecosystem. It shares a similar framework to Playnance's USDP token—such as taxation, whitelisting, and supply control—but is aimed at incentivizing participation, powering rewards, and fueling partnerships on the Playnance chain (ChainID: 1829). Unlike USDP, PlayX is **not pegged** to a fiat currency, and its value may fluctuate based on adoption and market dynamics.

Token Details

- **Token Name**: PlayX
- **Token Symbol**: X
- **Decimals**: 2
- **Blockchain**: Playnance (ChainID: 1829)

Pegging

Since PlayX is **not** a stablecoin, it does not maintain a 1:1 peg to the US Dollar (unlike USDP). Its market value may change over time depending on usage and demand in the Playnance ecosystem.

Key Features

1. Minting and Burning

- **Minting**: The owner can `mint` new tokens to designated addresses.
- **Burning**: The owner can `burn` tokens from the owner's balance, removing them from circulation.
- **burnFrom** (from OpenZeppelin's `ERC20Burnable`): Allows allowance-based token burning if a user wants to burn their own (or someone's, with permission) tokens.

2. Governance and Operational Integrity

- **Owner Role**: A single owner (or a multi-signature owner) controls minting, burning, tax rate changes, and whitelist management.
- **Audited**: The Playnance suite, including PlayX, is among the top audited smart contracts by Certik (audit report referenced in the broader documentation).

3. Pause and Unpause

- **Emergency Mechanism**: The owner can temporarily halt all transfers by calling `pause()`, and later `unpause()` when appropriate. This uses the ERC20Pausable extension.

4. Token Recovery

- **Accidental Transfers**: A function allows the owner to recover non-PlayX tokens accidentally sent to the contract, reducing losses from user mistakes.

5. Whitelist Mechanism

- **Tax Exemptions**: Certain addresses (e.g., partners, special users) can be whitelisted to avoid paying transaction taxes.
- **Management Authority**: Only the token owner can add or remove addresses from the tax-exempt whitelist.

6. Taxation (Adjustable. Currently set to 0%)

- **Initial Tax Rate**: 0% to encourage early adoption.
- **Maximum Tax Rate**: 1% (i.e., `_taxRate <= 100` in basis points).
- **Tax Wallet**: A wallet designated to receive any taxed amounts, settable by the owner.

Token Allocation

Below is the planned **PlayX** token distribution, including on-chain wallet addresses for transparency. The **total supply** is **20,000,000,000 X**.

Category	Allocation	Tokens	Wallet
Ecosystem Development	20%	4bn	0xF89313869e433022B8a8D1E189366101C32F386B
Partner Rewards & Incentives	20%	4bn	0x34Fb459ba738E7dd99244348f1D0d9Bd966F4bA7
Presale Allocation	25%	5bn	0xb6A846C1C5027150a9aC101cbAf6C48Ef658c23E
Liquidity & Reserves	15%	3bn	0xD4472e409155505a83916dF618369bCcA05376d1
Airdrops for Early	10%	2bn	0x8f3b2369DA40a24C789Bf41FACa2e15334b57cE5

Category	Allocation	Tokens	Wallet
Adopters			
Team & Advisors	5%	1bn	0x9a80df29665800649634CE81B502Cb887b4CFe24
Marketing &	5%	1bn	0x3a118b674957D844Af78cF83dF42FFDa6eDcA913

Smart Contract: PlayX (X)

Below is the **PlayX** smart contract source code, referencing OpenZeppelin libraries for ERC20 functionality, permissioned operations, and pausing.

```
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.20;

import "@openzeppelin/contracts/token/ERC20/ERC20.sol";
import "@openzeppelin/contracts/token/ERC20/extensions/ERC20Permit.sol";
import "@openzeppelin/contracts/token/ERC20/extensions/ERC20Burnable.sol";
import "@openzeppelin/contracts/token/ERC20/extensions/ERC20Pausable.sol";
import "@openzeppelin/contracts/access/Ownable.sol";

contract X is ERC20, ERC20Permit, ERC20Burnable, ERC20Pausable, Ownable {
    uint256 private _taxRate; // Tax rate in basis points (1 bps = 0.01%)
    address private _taxWallet;
    mapping(address => bool) private _whitelist;
    address[] private _whitelistedAddresses;

    event TokenRecovered(address tokenAddress, uint256 tokenAmount);

    constructor(address initialOwner, address taxWallet)
        ERC20("PlayX", "X")
        ERC20Permit("PlayX")
        Ownable(initialOwner)
    {
        _taxRate = 0; // Initialize tax rate at 0%
        _taxWallet = taxWallet;
    }

    function mint(address to, uint256 amount) public onlyOwner {
        _mint(to, amount);
    }

    function burn(uint256 amount) public override onlyOwner {
        super._burn(owner(), amount);
    }
```

```
function pause() public onlyOwner {
    _pause();
}

function unpause() public onlyOwner {
    _unpause();
}

function _update(address from, address to, uint256 amount)
    internal
    override(ERC20, ERC20Pausable)
{
    uint256 taxAmount = 0;

    // Apply tax if neither party is whitelisted and _taxRate > 0
    if (!_whitelist[from] && !_whitelist[to] && _taxRate > 0) {
        taxAmount = (amount * _taxRate) / 10000;
        super._update(from, _taxWallet, taxAmount);
    }

    uint256 amountAfterTax = amount - taxAmount;
    super._update(from, to, amountAfterTax);
}

function recoverToken(address tokenAddress, uint256 tokenAmount)
    public
    onlyOwner
{
    require(
        tokenAddress != address(this),
        "Cannot recover this token type"
    );
    IERC20(tokenAddress).transfer(owner(), tokenAmount);
    emit TokenRecovered(tokenAddress, tokenAmount);
}

function setTaxRate(uint256 newTaxRate) public onlyOwner {
    require(newTaxRate <= 100, "Tax rate cannot exceed 1%");
    _taxRate = newTaxRate;
}

function getTaxRate() public view returns (uint256) {
    return _taxRate;
}

function setTaxWallet(address newTaxWallet) public onlyOwner {
    require(
        newTaxWallet != address(0),
        "Tax wallet cannot be the zero address"
    );
    _taxWallet = newTaxWallet;
}
```

```
    function getTaxWallet() public view returns (address) {
        return _taxWallet;
    }

    function addToNoTaxWhitelist(address account) public onlyOwner {
        require(!_whitelist[account], "Account is already whitelisted");
        _whitelist[account] = true;
        _whitelistedAddresses.push(account);
    }

    function removeFromNoTaxWhitelist(address account) public onlyOwner {
        require(_whitelist[account], "Account is not whitelisted");
        _whitelist[account] = false;

        // Remove from _whitelistedAddresses array
        for (uint256 i = 0; i < _whitelistedAddresses.length; i++) {
            if (_whitelistedAddresses[i] == account) {
                _whitelistedAddresses[i] = _whitelistedAddresses[
                    _whitelistedAddresses.length - 1
                ];
                _whitelistedAddresses.pop();
                break;
            }
        }
    }

    function getNoTaxWhitelistedWallets() public view returns (address[] memory) {
        return _whitelistedAddresses;
    }

    function decimals() public view virtual override returns (uint8) {
        return 2;
    }
}
```

Conclusion

By incorporating adjustable taxation, a whitelisting feature, owner-controlled minting/burning, and the ability to pause or recover tokens, **PlayX** aligns with Playnance's vision of a secure and adaptable token ecosystem. The allocations across development, partnerships, presales, liquidity, airdrops, and team reflect a strategic approach to fostering growth, usability, and community engagement on the Playnance chain.