# Technical Documentation for UMB Token Smart Contract

**Overview:**

The **UMB Token** smart contract is an **ERC-20** token implementation with additional features such as burning, pausing, and transaction taxation. It is based on OpenZeppelin's secure Solidity libraries and includes administrative control via the **Ownable** pattern.

**Contract Details:**

**Contract Name**: UMBToken

**Token Name**: UMB Token

**Token Symbol**: UMB

**Maximum Supply**: 100,000,000 UMB (100 million tokens)

**Decimals**: 18

**Standards Implemented**:

- ERC20 (Standard fungible token functionality)
- ERC20Burnable (Allows token holders to burn their tokens)
- ERC20Pausable (Enables the contract owner to pause all transfers)
- Ownable (Restricts administrative functions to the contract owner)

**Dependencies:**

The contract imports the following OpenZeppelin libraries:

- @openzeppelin/contracts/token/ERC20/ERC20.sol
- @openzeppelin/contracts/token/ERC20/extensions/ERC20Burnable.sol

- @openzeppelin/contracts/token/ERC20/extensions/ERC20Pausable.sol
- @openzeppelin/contracts/access/Ownable.sol

**Key Features:**

**1. Minting and Initialization**

- At deployment, the full **MAX_SUPPLY** of 100,000,000 UMB is minted to the contract owner.
- The deployer (owner) is assigned as the **tax collector**.
- The owner is automatically **excluded from tax**.

**2. Transaction Tax Mechanism**

- A **transaction tax** (percentage) is applied to each transfer.
- The tax is deducted and sent to the **tax collector**.
- The contract allows the owner to:
    - Update the **transaction tax** (maximum of 10%).
    - Assign a **new tax collector**.
    - Exclude specific addresses from taxation.

**3. Burnable Tokens**

- Any token holder can burn their tokens using the **ERC20Burnable** extension.
- Reduces the total supply permanently.

**4. Pausable Transfers**

- The **ERC20Pausable** extension enables the owner to pause all token transfers.
- Useful in case of security vulnerabilities or system maintenance.

**Functions**

🔹 **constructor()**

- Initializes the token and assigns full supply to the owner.

🔹 **_update(address from, address to, uint256 value)**

- Internal function that deducts tax before processing a transfer.
- Ensures tax is not applied to excluded addresses.

🔹 **setTransactionTax(uint256 _tax)**

- Allows the owner to **set a transaction tax percentage** (max 10%).
- Emits a TaxUpdated event.

🔹 **setTaxCollector(address _collector)**

- Allows the owner to **change the tax collector address**.
- Emits a TaxCollectorUpdated event.

🔹 **excludeFromTax(address account, bool status)**

- Allows the owner to **exclude or include an address from taxation**.
- Emits an ExcludedFromTax event.

🔹 **pause() and unpause()**

- Owner-only functions to **pause and resume transfers**.

**Events**

**Security Considerations**

- The contract follows **OpenZeppelin's best practices**.

- Tax rate cannot exceed **10%**.
- **Pausable** feature ensures emergency control.
- **Only the owner** can modify tax settings.

**Deployment & Usage**

- **Blockchain Compatibility**: Works on **Ethereum and EVM-compatible chains**.
- **Token transfers follow ERC-20 standards**.
- **Administrative functions** are only accessible to the owner.