

한 번에 끝내는 블록체인 개발 A to Z

Chapter 3

Defi 기초 컨셉 구현

Chapter 3

Defi 기초 컨셉 구현

ERC20, Wrapped Token의 이해

Goal

이번 챕터에서는 디파이에서 주로 사용될 ERC20의 함수(approve, trasferFrom, mint, burn)와 Wrapped Token을 알아봅니다.

이에 대해서 이미 학습 하셨다면 건너 뛰어도 괜찮습니다.

공지사항

[안내] ETH 및 일부 ERC20 계열 디지털 자산 출금 수수료 조정 안내

등록일 2021.04.02 15:00 | 조회수 356472

안녕하세요. 가장 신뢰받는 글로벌 표준 디지털 자산 거래소 업비트입니다.

디지털 자산의 네트워크 비용 급증 시에도 업비트는 수수료보다 높은 네트워크 비용을 부담하여 안정적인 출금 서비스를 제공하고 있습니다.

다만 최근 들어 이더리움 및 ERC20 계열 디지털 자산에서 과도하게 높은 네트워크 비용이 장기간 발생되고 있어, 안정적인 출금 서비스 제공을 위해不得已하게 일부 디지털 자산의 출금 수수료가 인상될 예정임을 안내드립니다.

또한 최근 가격이 급등한 일부 디지털 자산은 출금수수료가 인하됩니다.

[수수료 조정 관련 안내]

- 수수료 조정 일정 : 2021-04-03 (토), 12:00 (KST)
- 조정 대상 : ETH 및 일부 ERC20 계열 디지털 자산

(ETH, ANKR, AERGO, SXP, STORJ, MTL, ENJ, SAND, STMX, MANA, LINK, OMG, BORA, SNT, ELF, ADX, STPT, MATIC, T, LINA, ZRX, BASIC, BAT, CVC, POWR, RINGX, LAMB, CRE, KNC, POLY, REP, OBSR, MVL, GLM, PRO, CHR, DENT, SOLVE, AQT, RFR, LOOM, FX, QTCON, CTSI, OGN, NCASH, EDR, FCT2, ANT, TSHP, OXT, UNI, AUCTION, CRV, TON, BFT, RSR, DAWN, HUM, DAI, FOR, AXS, DMT, RLC, HUNT, RCN, ITAM, PUNDIX, BNT, ONIT, COMP, DNT, IOTX, NJ, NMR, LRC)

- 가상화폐 거래소에서도 심심치않게 ERC20이라는 용어를 볼 수 있다.
- ERC20에 해당하는 코인들이 매우 많은 것을 볼 수 있다.
- 대부분의 디파이 서비스들에서도 LP토큰과 거버넌스 토큰을 ERC20으로 발행하여 지급한다.



ERC20

ETH와 ERC20은 어떻게 다를까?

ERC20

- ERC20은 이더리움에 20번째로 제안되어 채택된 표준이다.
- ERC20 표준은 이더리움의 토큰을 구현하는 표준이다.
- ETH는 코인, ERC20은 토큰이라고 할 수 있다.
- ETH는 이더리움 네트워크의 기축 통화이고 ERC20은 이더리움 네트워크에서 정해진 규칙에 따라 누구나 프로그래밍으로 만들 수 있는 토큰이다.
- 거래소에서 거래되는 거의 대부분의 이더리움 계열의 토큰들은 ERC20 토큰이다.
- 잔액, 전송, 사용 권한, 생성, 소각 등의 함수들이 정의가 되어있다.
- ERC20들의 함수를 활용하여 Defi가 탈중앙화된 방식으로 동작한다.

ERC20

```
function name() public view returns (string)
function symbol() public view returns (string)
function decimals() public view returns (uint8)
function totalSupply() public view returns (uint256)
function balanceOf(address _owner) public view returns (uint256
balance)
function transfer(address _to, uint256 _value) public returns (bool
success)
function transferFrom(address _from, address _to, uint256 _value)
public returns (bool success)
function approve(address _spender, uint256 _value) public returns
(bool success)
function allowance(address _owner, address _spender) public view
returns (uint256 remaining)
```

ERC20

- name, symbol: 토큰의 이름과 심볼.
 - decimals: 소숫점 몇째짜리까지 단위로 표현할 것인가를 나타냄.
18이면 0.000000000000000001개까지 다를 수 있다.
 - balanceOf, transfer: 잔액을 조회한다, 특정 주소로 토큰을 전송한다.
 - **transferFrom**: 특정 주소가 코인을 다른 주소로 전송한다.
 - **approve**: 특정 주소가 코인을 전송하는 것을 허용한다.
 - allowance: 특정 주소가 코인을 전송하는 것을 허용한 개수를 확인한다.
 - 각 함수를 구현체는 토큰 개발자의 몫이다.
- 하지만 널리 사용 되는 오픈소스가 있다.

```
name() public  
symbol() publ  
decimals() pu  
totalSupply()  
balanceOf(addr  
  
transfer(addr  
  
transferFrom(  
eturns (bool s  
approve(addre  
ccess)  
allowance(add
```

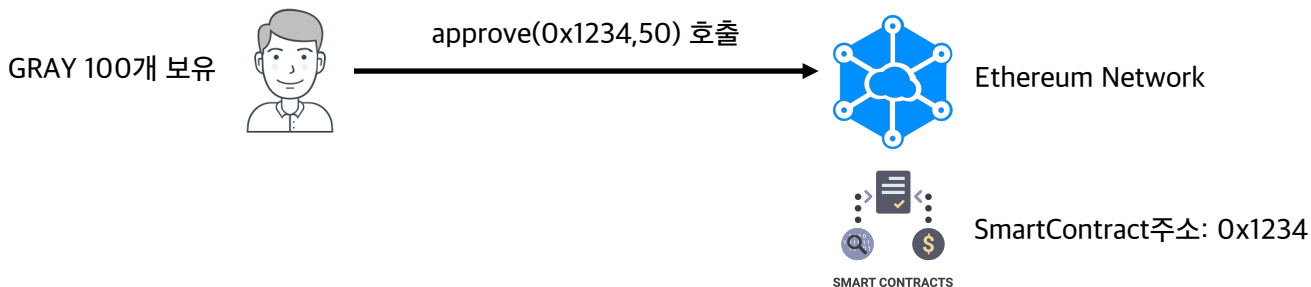
Token attributes

Property	Value
Symbol	MANA
Token Type	ERC20
Name	Decentraland MANA
Number of Decimals	18

ERC20(transferFrom, approve)

- `transferFrom`, `approve` 두개의 함수가 ERC20을 디파이에서 활용하는데 핵심이다.

```
function approve(address _spender, uint256 _value)
```



- 0x1234 SmartContract는 나의 GRAY 50개를 언제든지 다른 주소로 전송 할 수 있다.
- 내가 0x1234에게 GRAY 50개 사용 권한을 approve 했기 때문이다.

ERC20(approve)

```
function approve(address spender, uint256 amount) public virtual override returns (bool) {
    address owner = _msgSender();
    _approve(owner, spender, amount);
    return true;
}

function _approve(
    address owner,
    address spender,
    uint256 amount
) internal virtual {
    require(owner != address(0), "ERC20: approve from the zero address");
    require(spender != address(0), "ERC20: approve to the zero address");

    _allowances[owner][spender] = amount;
    emit Approval(owner, spender, amount);
}
```

- owner: msgSender(이 함수를 호출한 주체)
- spender: 0x1234
- amount: 50
- _allowances라는 자료구조의 값이 바뀌었다.

ERC20(transferFrom)

- 이제 다른 주소에서 transferFrom 함수를 호출하여 누구든지 나의 GRAY 500개를 0x1234로 전송 할 수 있다.
- 다시 한번 approve를 호출 하는 것을 볼 수 있다.
- approve 전에 현재 approve된 개수를 확인한다. 여기서 allowance가 사용된다.
- transfer 하기 전에 현재 허용된 개수 만큼 감소시킨다.
- 유동성 공급 할 때 유동성을 가지는 SmartContract에게 허용한 개수만큼 나의 토큰을 전송할 권한을 준다.

```
function transferFrom(
    address from,
    address to,
    uint256 amount
) public virtual override returns (bool) {
    address spender = _msgSender();
    _spendAllowance(from, spender, amount);
    _transfer(from, to, amount);
    return true;
}
```

```
function _spendAllowance(
    address owner,
    address spender,
    uint256 amount
) internal virtual {
    uint256 currentAllowance = allowance(owner, spender);
    if (currentAllowance != type(uint256).max) {
        require(currentAllowance >= amount, "ERC20: insufficient allowance");
        unchecked {
            _approve(owner, spender, currentAllowance - amount);
        }
    }
}
```

ERC20(mint/burn)

```
function _mint(address account, uint256 amount) internal virtual {
    require(account != address(0), "ERC20: mint to the zero address");

    _beforeTokenTransfer(address(0), account, amount);

    _totalSupply += amount;
    unchecked {
        // Overflow not possible: balance + amount is at most totalSupply + amount,
        _balances[account] += amount;
    }
    emit Transfer(address(0), account, amount);

    _afterTokenTransfer(address(0), account, amount);
}

function _burn(address account, uint256 amount) internal virtual {
    require(account != address(0), "ERC20: burn from the zero address");

    _beforeTokenTransfer(account, address(0), amount);

    uint256 accountBalance = _balances[account];
    require(accountBalance >= amount, "ERC20: burn amount exceeds balance");
    unchecked {
        _balances[account] = accountBalance - amount;
        // Overflow not possible: amount <= accountBalance <= totalSupply.
        _totalSupply -= amount;
    }

    emit Transfer(account, address(0), amount);

    _afterTokenTransfer(account, address(0), amount);
}
```

- mint와 burn은 LP토큰을 생성하고 소각하는데 쓰인다.
- 유동성 공급을 하면 공급자에게 mint 하고
유동성을 철회하면 공급자의 LP토큰을 burn한다.
- totalSupply 값이 변하므로 전체 LP토큰 개수에서 나의 LP토큰 개수가 차지하는 비율을 구할 수 있다.
이를 통해 전체 유동성 풀에서 나의 유동성이 차지하는 비율을 구할 수 있다.






Wrapped Token

WBTC, WETH는 뭐고 왜 쓰일까?

Wrapped Token

- Wrapped Token의 대표적인 토큰으로는 WBTC, WETH가 있다.
- 시가총액 18위에 WBTC가 보이고, 유니스왑에는 WBTC/ETH 풀이 있다.
- NFT 마켓플레이스인 OpenSea의 Offer 기능에도 WETH가 사용이 된다.
- WBTC/WETH는 비트코인과 이더리움의 ERC20 형태이다.
- BTC:WBTC, ETH:WETH는 1:1 비율의 가격을 가진다.

17	 TRON TRX		\$0.07019
18	 Wrapped Bitcoin	WBTC	\$23,455.38
19	 Ethereum Classic	ETC	\$38.21

시가총액 18위인 Wrapped Bitcoin



유니스왑의 WBTC/ETH 유동성 풀

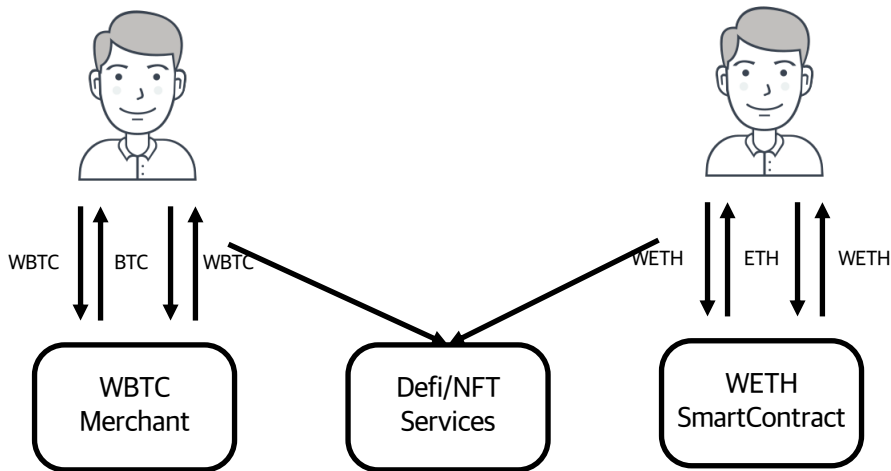
Offers

Price	USD Price	Floor Difference	Expiration
♦ 10 WETH	\$17,098.80	64% below	3 days
♦ 0.2 WETH	\$341.98	99% below	about 22 hours
♦ 0.081 WETH	\$138.50	100% below	about 13 hours

OpenSea에서도 활용되는 WETH

Wrapped Token

- WBTC와 WETH를 활용하면 비트코인과 이더리움을 디파이에서도 활용 할 수 있다.
- BTC의 경우에는 이더리움과 다른 네트워크이니 WBTC는 필수로 사용 해야 하는 것으로 생각된다.
- ETH는 같은 이더리움 네트워크인데 왜 사용이 될까?



Wrapped Token

- ERC20 표준은 ETH 보다 나중에 나온 표준이고 ETH와 ERC20은 구현이 다르다.
- ETH는 이더리움의 프로토콜 레벨에서 구현이 된 것이기 때문에 SmartContract로 컨트롤 하는 것에 한계가 있다.
ETH 전송 정도는 가능하지만 ERC20에서 보았던 approve, allowance 같은 함수는 사용 할 수 없다.
- 디파이 서비스의 구현과 동작을 더 간단하게 하기 위해서 이더리움을 사용하지 않고 이더리움을 ERC20 형태로 만든 토큰을 사용한다.

