



# GRC20 and Coins

---

## Token Standards in the Gno Ecosystem

---

How native coins and tokens differs

How to implement tokens and their use cases

# Coins

Managed by the `banker` module, separate from GnoVM,

- ✓ **Native type**
- ✓ Efficient gas use -- Lightweight
- ✓ Used for staking, fees

- ✗ **No custom logic**
- ✗ Limited dApp usage

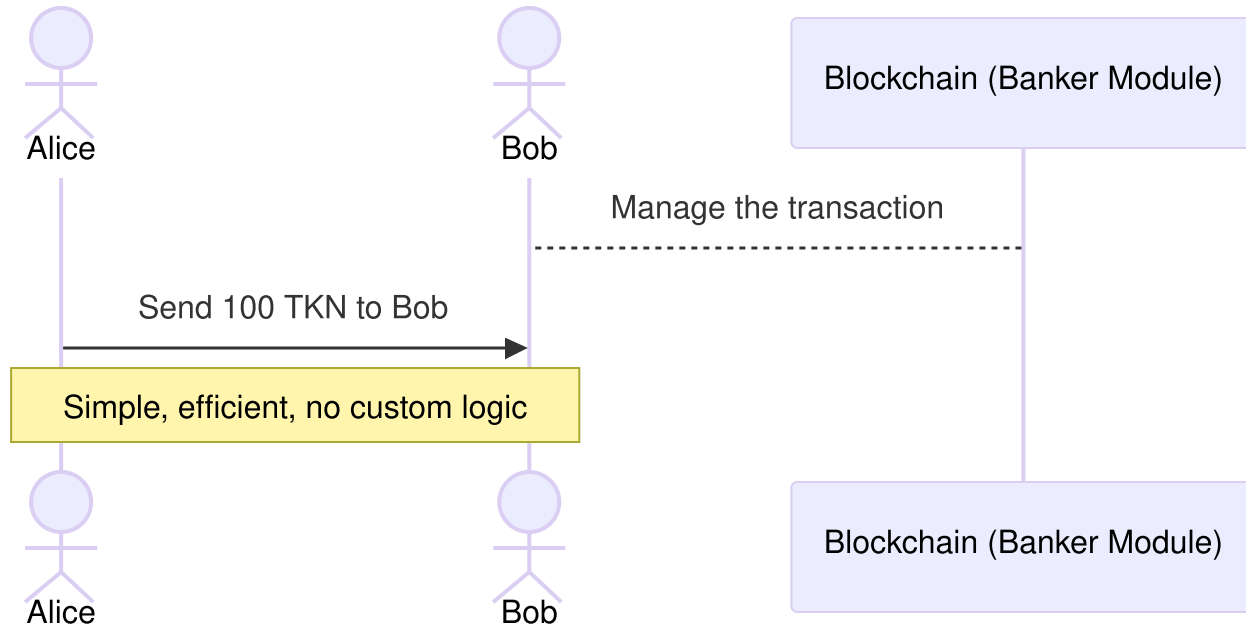
 [Read the Coin and Bankers Docs](#)



A Coin is a Gno type made of a denomination and an amount.

```
// Defines a basic Coin with a denomination and amount
type Coin struct {
    Denom  string `json:"denom"` // e.g., "GNOT"
    Amount int64  `json:"amount"` // e.g., 1000
}
```



```
banker.SendCoins(from, to, coins)
banker.IssueCoin(addr, denom, amount)
banker.RemoveCoin(addr, denom, amount)
coins := banker.GetCoins(addr)
```





# GRC20 Tokens

## ERC20-style Smart Contracts

-  **Fungible**, programmable token standard
-  Stored and executed in **Gno realm**



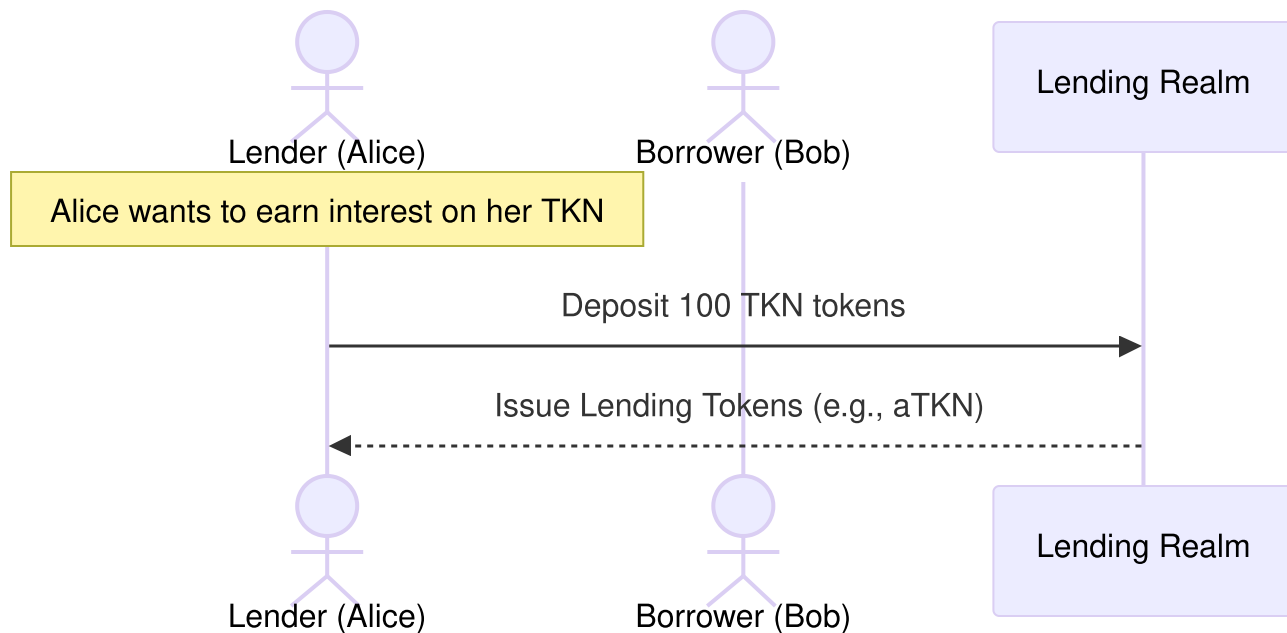
**GRC20 = Gno's version of ERC20/CW20**



Fully programmable logic on-chain

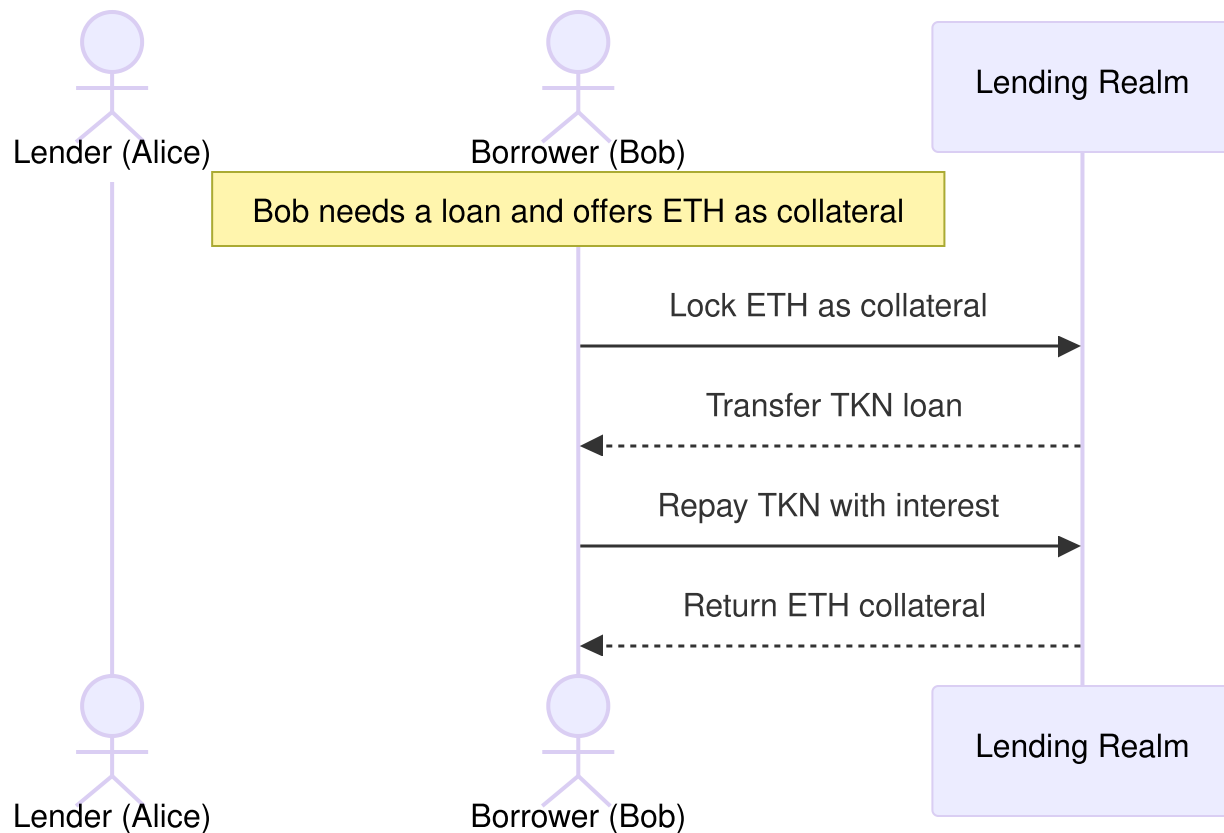


# GRC20 Tokens



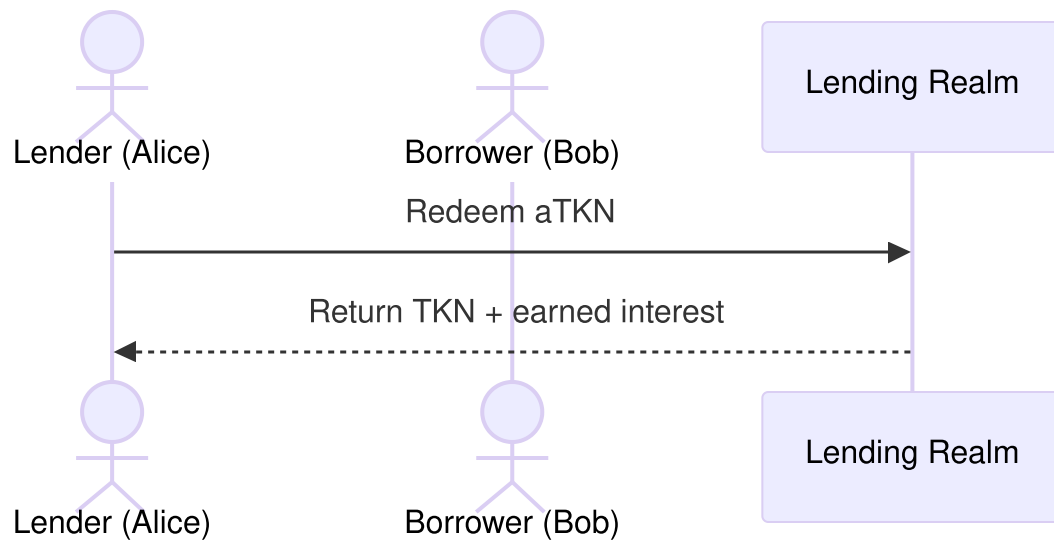


# GRC20 Tokens





# GRC20 Tokens










**Why use GRC20 ?**

## Use Case 1: Token Gating

# Token Gating

Use GRC20 tokens to:



-  Unlock gated content
-  Access private organizations
-  Control premium event access

```
if (!hasGRC20(user)) {  
  return "Access Denied"  
}
```

## Use Case 2: **Vaults**



# Vaults: Passive Income



- Deposit GRC20 → Get yield-bearing shares
-  Earn interest as the vault grows
-  Withdraw anytime with rewards

```
vault.deposit(user, GRC20.amount)  
shares = calculateShares(user)
```

## Use Case 3: Wrapping Coins



## Wrapping Native Coins

-  Works in DeFi (Decentralized Finance)
-  Composable – Connects to other apps













### Enables:

- Liquidity pools
- Lending protocols
- Cross-chain assets



# Comparison Table

Feature	 Coins (Banker)	 GRC20 Token
Native to chain		
Composable in dApps		
Custom Logic		
Governance Control	Centralized	Decentralized
Efficiency		 Slight overhead

Coins vs Grc20





# Let's build our own GRC20

Let's fill our functions !

```
func init() {}

// Informations
func TotalSupply() uint64 {}
func BalanceOf(owner std.Address) uint64 {}

// Create / Delete
func Mint(to std.Address, amount uint64) {}
func Burn(from std.Address, amount uint64) {}

// Send
func Transfer(to std.Address, amount uint64) {}

// Send from another address
func Allowance(owner, spender std.Address) uint64 {}
func Approve(spender std.Address, amount uint64) {}
func TransferFrom(from, to std.Address, amount uint64) {}
```

# Sources

- [!\[\]\(467d80e979964f7f8c752fb22248b5b7\_img.jpg\) gno.land](#)
- [foo20](#)
- [bar20](#)

 Fully on-chain Gno smart contracts

Read the [Coin](#) and [Bankers Docs](#)