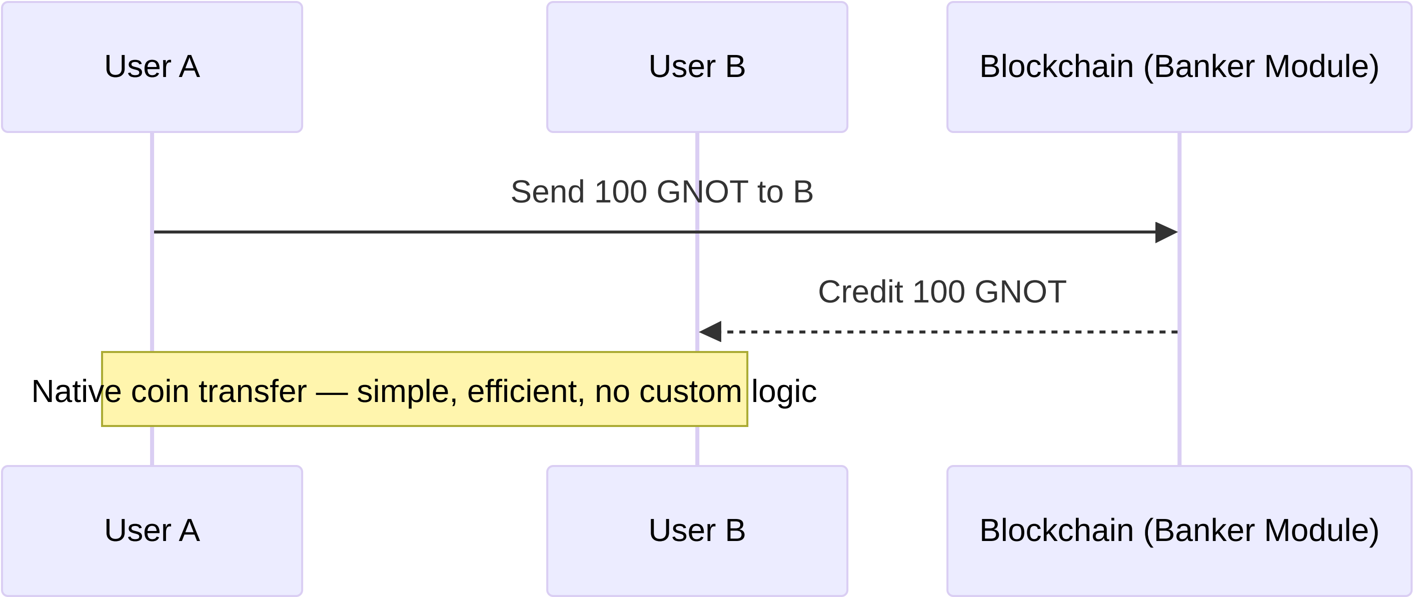
🪙 GRC20 and Coins

Token Standards in the Gno Ecosystem

How native coins and smart contract tokens differ — and why it matters

Use case


User A
User B
Blockchain (Banker Module)
Send 100 GNOT to B
Credit 100 GNOT
Native coin transfer — simple, efficient, no custom logic
User A
User B
Blockchain (Banker Module)

⚙️ Native Coins

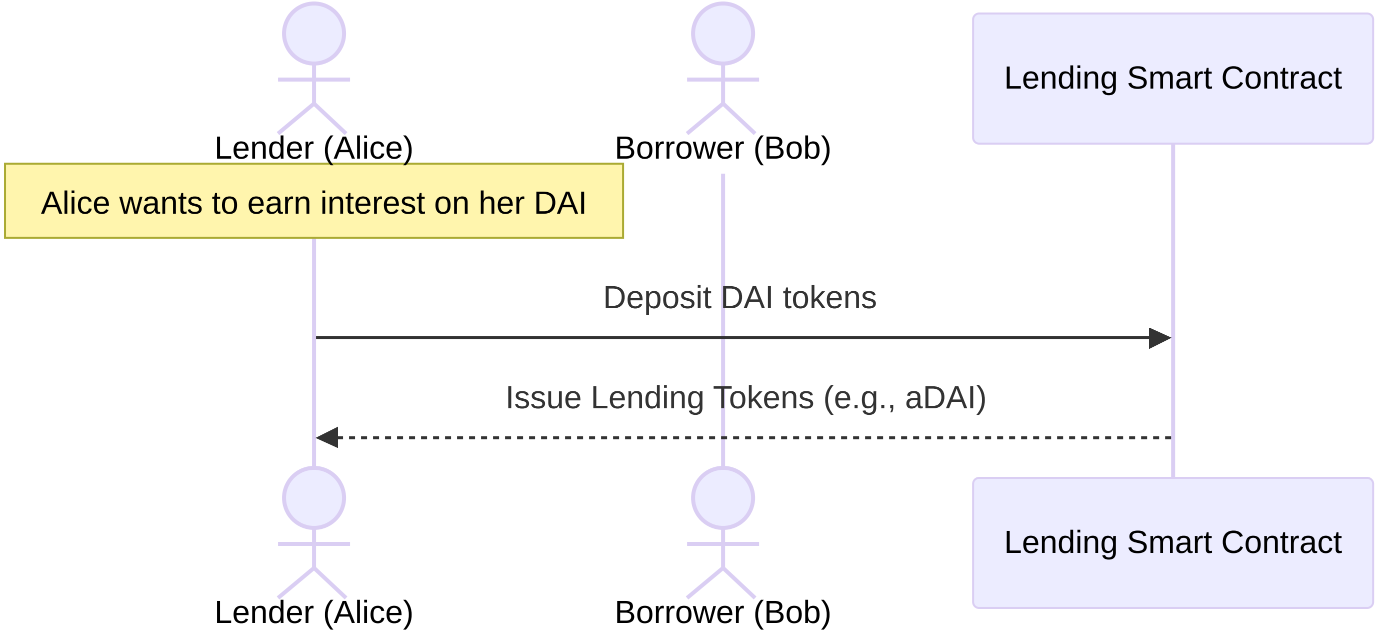
Banker Module (stdlib)

✅ **Native to chain**
✅ Efficient gas use
✅ Used for staking, fees

❌ **No custom logic**
❌ Not composable
❌ Limited dApp usage

🔗 Read the Banker Docs

Use case



Lender (Alice)

Borrower (Bob)

Lending Smart Contract

Alice wants to earn interest on her DAI

Deposit DAI tokens

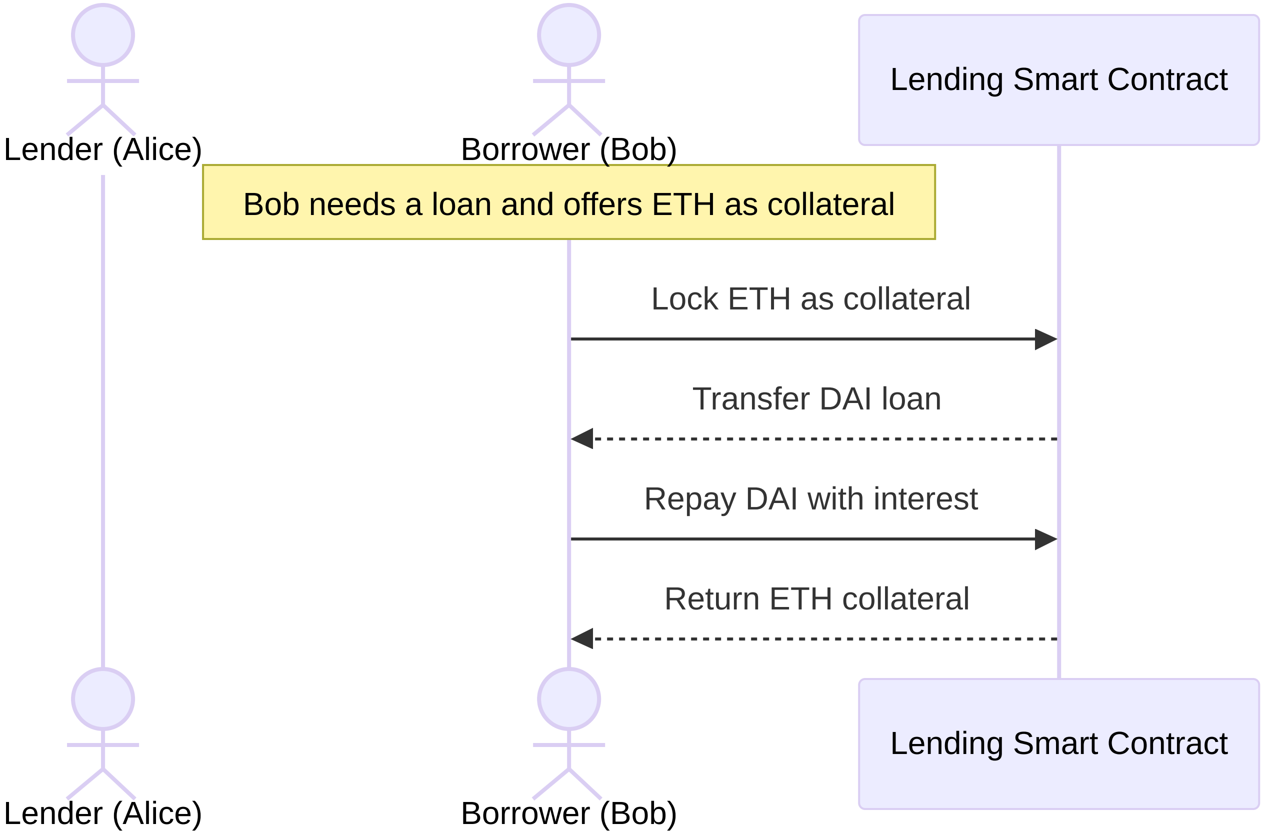
Issue Lending Tokens (e.g., aDAI)

Lender (Alice)

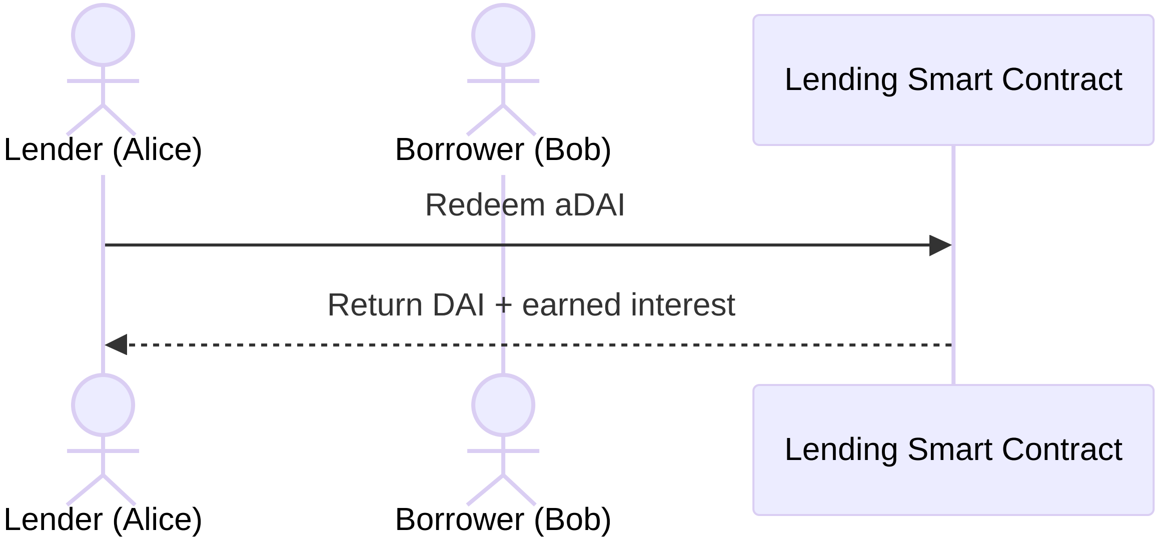
Borrower (Bob)

Lending Smart Contract

Use case


Lender (Alice)
Borrower (Bob)
Lending Smart Contract
Bob needs a loan and offers ETH as collateral
Lock ETH as collateral
Transfer DAI loan
Repay DAI with interest
Return ETH collateral
Lender (Alice)
Borrower (Bob)
Lending Smart Contract

Use case


Lender (Alice)
Borrower (Bob)
Lending Smart Contract
Redeem aDAI
Return DAI + earned interest
Lender (Alice)
Borrower (Bob)
Lending Smart Contract

🧬 GRC20 Tokens

ERC20-style Smart Contracts

- 🔁 **Fungible**, programmable token standard
- ⚙️ Stored and executed in **Gno smart contracts**
- ✅ Patterns: `transfer`, `allowance`, `approval`

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

> 🛠 Fully programmable logic on-chain

⚔️ Comparison Table

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

🧠 Why Use GRC20?

🔄 Interoperable

Integrates with wallets, dApps, DeFi

🧩 Modular Logic

Custom minting, access control, burn rules

✨ Enables decentralized finance & token ecosystems

🎟️ Use Case 1: Token Gating

🎟️ Token Gating

Use GRC20 tokens or NFTs to:

- 🔐 Unlock gated content
- 🗳️ Access private DAOs or groups
- 🎫 Control premium event access

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

> Block access unless token is held — exclusive by design

💰 Use Case 2: Vaults

💰 Vaults: Yield Strategy

- Deposit GRC20 → vault
- Receive yield-bearing **shares**
- 📈 Earn passive returns

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

Use Cases:

- 🪙 Savings contracts
- 🌾 Yield farming strategies
- 🔐 Staked lockups

♻️ Use Case 3: Wrapping Coins

♻️ Wrapping Native Coins

Convert GNOT to GRC20:

```
GRC20{GNOT}
```

- 🧃 DeFi-ready
- 🧮 Tradable
- 🔁 Composable in dApps

✅ Enables:

- AMMs / liquidity pools
- Lending protocols
- Cross-chain assets

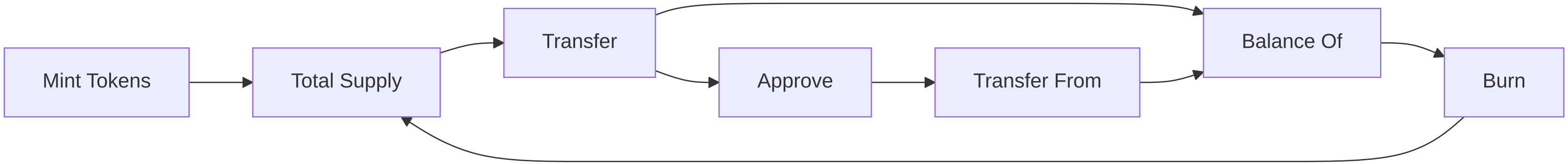
🧪 Summary Table

	🪙 Coins (Banker)	🧬 GRC20 Tokens
Chain-native	✅	❌
Smart contract support	❌	✅
Composable in DeFi / dAppps	❌	✅
Gas-efficient	✅	⚠️ Minor cost
Ideal for	Fees, Gas	dApps, DAOs, DeFi

🔧 Let’s build our own GRC20

```go
func init() {}

func TotalSupply() uint64 {}

func BalanceOf(owner std.Address) uint64 {}

func Allowance(owner, spender std.Address) uint64 {}

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

func Approve(spender std.Address, amount uint64) {}

func TransferFrom(from, to std.Address, amount uint64) {}

func Faucet() {}

func Mint(to std.Address, amount uint64) {}

func Burn(from std.Address, amount uint64) {}

func Render(path string) string {}
```

🔧 Let’s build our own GRC20



Start coding today:

🛠️ This contract serves as a foundational example for creating GRC20 tokens on Gno.land. For a more detailed guide on implementing GRC20 tokens, refer to the Gno.land Documentation.

- foo20
- bar20

> 💻 Fully on-chain Gno smart contracts

🎬 Thanks!

- 🌐 gno.land
- 🧠 Built with Gno smart contracts
- 🐢 Fast. Lightweight. Deterministic.