Celo for Ethereum Developers

Overview of the similarities and differences between the Celo and Ethereum blockchains.

tip For a general overview of the Celo network and architecture, seethe Celo Overview page .

What is Celo's Relationship to Ethereum?

Celo is a layer 1 protocol and blockchain platform, and the Celo Mainnet is entirely separate from the Ethereum network. While the Celo client originated as a fork of Ethereum Go language client,go-ethereum (or geth), it has several significant differences, including a proof-of-stake based PBFT consensus mechanism. All the cryptoassets on Celo have ERC-20 compliant interfaces, meaning that while they are not ERC-20 tokens on the Ethereum Mainnet, all familiar tooling and code that support ERC-20 tokens can be easily adapted for Celo assets, including the Celo Native Asset (CELO) and the Celo Dollar (cUSD).

In terms of programmability, Celo is similar to Ethereum. Both networks run the Ethereum Virtual Machine (EVM) to support smart contract functionality. This means that all programming languages, developer tooling and standards that target the EVM are relevant for both Celo and Ethereum. Developers building on Celo can write smart contracts inSolidity , useTruffle for smart contract management and take advantage of smart contract standards that have already been developed for Ethereum.

The ERC-20 Token Standard

TheERC20 token standard is a standard API for tokens within smart contracts. This standard interface allows any tokens to be re-used by different applications. The ERC20 token standard is blockchain agnostic, so ERC20 tokens can be implemented on any blockchain.

The standard includes theoptional functions

function

name ( )

public view returns

(string) function

symbol ( )

public view returns

(string) function

decimals ( )

public view returns

(uint8) and therequired functions

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) and includes the following events

event Transfer (address indexed _from , address indexed _to , uint256 _value) event Approval (address indexed _owner , address indexed _spender , uint256 _value) An ERC20 compliant contract must include the required functions and events atminimum . It can include additional functions and events and still be ERC20 compliant.

The Celo Native Asset and the Celo Dollar

This interface is relevant for two important assets on the Celo network, the Celo native asset (CELO) and the Celo Dollar (cUSD).

CELO was called Celo Gold (cGLD) when the contract was deployed, so you will often see references to Celo Gold in the codebase. CELO and cGLD are the same thing. You canview the CELO implementation here.

CELO has an ERC20 interface, so users can interact with CELO via the token standard, but it is important to note that not all CELO transfers are required to go through the token contract. CELO can also be transferred by specifying the value field of a transaction, in the same way that ETH can be transferred in Ethereum. To properly monitor balance changing operations of CELO, it can be helpful to useCelo Rosetta. Celo Rosetta provides an easy way to obtain changes that are not easily queryable using the celo-blockchain RPC.

The Celo Dollar (cUSD) is implemented solely as a smart contract, so all cUSD actions are mediated by the smart contract. You canview the implementation here.

Key differences between Celo and Ethereum

Features exclusive to Celo

1. Celo allows users to pay transaction fees in cryptoassets other than the native asset. On Ethereum, users must pay transaction fees in Ether. For example, users can send cUSD, and then pay any transaction fees in cUSD as well.
2. The Celo protocol uses BFT Proof-of-Stake for maintaining consensus. This allows blocks on Celo to be created in 5 seconds, as compared to ~12+ seconds on Ethereum. In addition, all blocks are finalized immediately, so there is no need to wait for more than 1 block confirmation to ensure that a transaction won't be reverted.

Things to watch out for

1. As previously mentioned, CELO transfers are not required to happen via the ERC20 interface. A user's CELO balance may change without any interaction with the CELO contract, as they may transfer CELO natively.
2. Celo transaction objects are slightly different from transaction objects on Ethereum.
3. Ethereum transaction objects include fields to, value, gas, gasPrice, data, nonce, signature (v,r,s).
4. Celo transaction objects include the same fields as Ethereum transaction objects, plus the feeCurrency field.
5. This additional field is included to allow users topay transaction fees in different currencies.
6. As of May 19th, 2021, with theDonut hardfork

7. , the Celo network accepts both Celo transaction objects and Ethereum transaction objects as valid Celo transactions. This means that you can use most Ethereum tools with Celo, right out of the box (just point them at the Celo network). When sending Ethereum formatted transactions on Celo, you will not be able to use Celo features of specifying transaction fee currencies or full node incentives.

8. When using mnemonic seed phrases (or secret phrases), Celo accounts (a private key and corresponding address) are derived differently from Ethereum accounts. The Celo key derivation path ism/44'/52752'/0'/0

9. whereas Ethereum's ism/44'/60'/0'/0
10. . This means that going from a seed phrase to accounts will be different when using Ethereum vs Celo wallets.
11. The Valora wallet uses two types of accounts: externally owned accounts and meta-transaction wallets. There are important consequences for wallet developers and dapp developers building on Celo as Valora is one of the main interfaces for Celo users. You can find more information aboutValora accounts here
12. .

Deploying Ethereum Contracts to Celo

Celo runs the EVM which means that smart contracts written for Ethereum can easily be deployed to Celo, the main difference being that you just need to connect to a Celo node instead of an Ethereum node. You can connect to your own Celo node or to a Celo node service provider likeFigment Datahub .

This tutorial goes over how to start an ultralight node that runs locally and use it to deploy a contract to the Alfajores testnet using Truffle.

This tutorial goes over how to connect to a remote node and use ContractKit to deploy a contract to Alfajores using Truffle.

Protocol Differences

OPCODES & Block headers

Celo does not support theDIFFICULTY orGASLIMIT opcodes. These fields are also absent from Celo block headers.

Precompiled Contracts

Celo includes all of the precompiled contracts in Ethereum, but also adds additional contracts.Here is the list of Celo precompiled contracts as of Celo version 1.3.2. You can find the latest updates by selecting the most recent release tag.

Core Contract Calls

The blockchain client makes some core contract calls at the end of a block, outside of transactions. Many are done on epoch blocks (epoch rewards ,validator elections , etc.), but not all. For example, thegas price minimum update can happen on any block. Logs created by these contract changes are included in a single additional receipt in that block, which references the block hash as its transaction hash, even though there is no transaction with this hash. If no logs were created by such calls in that block, no receipt is added.

Node management APIs

Celo nodes have a slightly different RPC interface than geth nodes. There are some additional RPC endpoints to help validators manage their nodes, they can be foundhere andhere .

You can find the full list of RPC API endpoints inthis file . Edit this page Previous Token Addresses Next Launch Checklist