Circle Web3 Services Primitives

Suggest Edits

Discover the essential primitives powering Circle's Web3 Services:

1. Environments:
2. Understand the differences between Testnet (Sandbox) and Mainnet (Production) environments that facilitate the development and deployment.
3. Wallets:
4. Explore the core functionality of wallets, facilitating secure storage and management of digital assets for your end-users.
5. Wallet Sets:
6. Understand the concept behind wallet sets, which enable efficient management of multiple developer-controlled wallets.
7. Users:
8. Streamline user interactions within Circle's Web3 ecosystem by simplifying pin and security question management for web3 wallets.
9. Smart Contracts:
10. Explore smart contracts and our features that enable deployment and interaction.
11. Transactions:
12. Delve into the transactional capabilities offered by Circle's Web3 Services, empowering asset transfers, contract interactions, and deployments.
13. Paymaster Policies:
14. Understand Gas Station Paymaster policies and how they provide control for the sponsoring of gas fees.
15. Monitored Tokens:
16. Enhanced control over token visibility, allowing selective filtering of tokens to enhance user experience.

By understanding these foundational components, you can unlock the full potential of Circle's Web3 Services and enhance your decentralized web experiences.

Video Tutorial: Understanding Programmable Wallets Primitives

Watch this video to gain a fundamental understanding of Circle's Web3 Services primitives. This video will help you grasp the basics of wallet functionality, the role of private keys, and how Circle's API extends wallet capabilities with user-friendly flows and smart contract integration.

undefined

Environments

Circle supports two environments: Testnet (Sandbox) and Mainnet (Production). When you initially sign up, you will be provided access to Testnet, which is free to use. Once ready to promote to Mainnet, select “Access Mainnet” from the developer dashboard and start the onboarding process.

- Testnet:
- Circle's Testnet environment acts as a Sandbox, enabling developers to safely test the functionality of Web3 Services without the risks associated with a Production environment. Connected to blockchain Testnets, it provides a secure space to build applications and conduct testing before deployment.
- Mainnet:
- The Mainnet environment represents the Production environment where real user transactions and interactions occur. It is connected to the live blockchain network, ensuring the handling of real-world tokens. Developers should consider potential differences in blockchain confirmation times and gas fees when migrating from the Testnet to the Mainnet environment.

Wallets

A Web3 Wallet, a decentralized or blockchain wallet, is a digital wallet that allows users to securely store, manage, and interact with their crypto assets and digital identities on the Web3 ecosystem. Unlike traditional wallets, Web3 wallets integrate with decentralized applications (dApps), enabling users to authenticate, sign transactions, and interact with blockchain-based services. With features like encryption and decentralized key management, Web3 wallets provide users with control, privacy, and security over their digital assets and interactions within the decentralized web.

Wallet attributes:

- Address:
- An on-chain identifier specific to a blockchain. It's worth noting that multiple wallets can share the same address on different EVM blockchains.

- Custody Type:
- Determines the control over the invocation of the wallet's private key, distinguishing between user-controlled and developer-controlled wallets. Explore theProgrammable Wallets infrastructure models
- for further insights.
- Account Type:
- Specifies whether the wallet is of the externally owned account (EOA) type or smart contract account (SCA) type.
- State:
- Indicates the current state of the wallet, signifying whether it is active or frozen.
- Blockchain:
- Identifies the blockchain and network to which the wallet belongs, providing context and compatibility information.

Wallet Sets

A wallet set is ahierarchical deterministic (HD) wallet , where all the children's wallets are created and managed by the same unique cryptographic private key. For each developer-controlled wallet created, you must provide a wallet set from which the wallet will be created.

Wallet set attributes:

- Wallets:
- A wallet set can contain one or many wallets. A wallet set can have wallets with addresses on multiple blockchains. On EVM chains, a wallet set can have wallets on different chains sharing the same address.

Users

A User represents an end-user of your application, allowing you to map their account to associated wallets, tokens, and transactions. The userId , provided during account creation, is a unique identifier connecting the user to these components.

User attributes:

- PIN:
- A Personal Identification Number (PIN) enhances security by authenticating users when signing wallet transactions. Users must set up a unique numerical code PIN to access their account and perform sensitive actions.
- Security Questions:
- Users must set up security questions for account recovery, providing additional protection.
- User Token:
- The user token is a 60-minute session token utilized for requests requiring a user challenge, such as entering a PIN code. After expiration, a new user token is generated via the same endpoint.
- Challenge:
- Challenges act as checkpoints, prompting users to enter their PIN code to authorize critical actions like transactions or executing smart contracts. During user creation, the initial challenge guides users in setting their PIN code and Recovery Method, enabling account recovery if necessary.
- Status:
- Denotes whether the user is enabled or disabled, indicating their account's current operational state.

Smart Contracts

Smart contracts are the cornerstone of your deployed and on-chain imported contracts within the Smart Contract Platform (SCP). To interact with a smart contract on-chain, it is necessary to import it first. All smart contracts deployed via the web3 services APIs can be readily interacted with by default. Once imported or deployed, developers can leverage the functionality provided by the smart contract. SCP offers seamless accessibility to smart contracts through the developer console and APIs.

With SCP, developers can perform a wide range of tasks, including deploying NFT contracts, creating on-chain loyalty programs, interacting with popular DeFi projects like Uniswap with just a few clicks, seamlessly integrating Circle contracts (such as CCTP) into their apps, and deploying custom contracts written specifically for their application needs.

ABI (Application Binary Interface)

A smart contract Application Binary Interface (ABI) is a standardized interface description for interacting with a smart contract deployed on a blockchain. It defines the smart contract's structure, data types, and functions, enabling seamless communication between the contract and external entities, such as user interfaces, other contracts, or off-chain applications.

The smart contract ABI serves as a contract's public interface, outlining how to interact with it. It typically includes the following components:

1. Function Signatures
2. : The names of the functions within the smart contract, along with their input and output parameter types. These

signatures define the available operations that can be performed on the contract.
3. Event Definitions:
4. Events provide information about specific occurrences or state changes within the contract. The ABI includes event definitions, specifying their names and the data associated with each event.
5. Data Structures:
6. If the contract uses complex data structures or custom types, the ABI includes their definitions. Examples can include structs, arrays, or mappings.

The ABI allows developers and external applications to interact with a smart contract without understanding the contract's underlying implementation details or source code. It provides a standardized and language-agnostic way to communicate with smart contracts, facilitating interoperability and seamless integration with various platforms, tools, and services.

Smart contract attributes:

- ID:
- Once a smart contract is imported or deployed, Circle will assign an ID. This ID will be used in subsequent requests to read and execute the contract.
- ABI Function Signature:
- Used to specify the desired function to be read or executed. By providing the function signature, developers can effectively read data from the contract or trigger actions defined by that function.
- ABI Parameters:
- The parameters provided to the ABI function signature. Parameters are provided to the ABI function signature in the same order as received.
- ABI JSON:
- The contract's ABI in a JSON stringified format.
- Functions:
- Read and execute functions supported by the contract. The names of the functions within the smart contract, along with their input and output parameter types. These signatures define the available operations that can be performed on the contract.
- Events:
- Events the contract can emit. Events provide information about specific occurrences or state changes within the contract. The ABI includes event definitions, specifying their names and the data associated with each event.
- Transactions:
- List of on-chain transactions executed on the contract. Currently, it is only available in the developer dashboard.

Transactions

A transaction encompasses all blockchain activities, such as token transfers (inbound or outbound), smart contract deployment, and smart contract execution.

Transactions also include metadata that outlines their state. Once a transaction is created, Circle monitors the transactions on its respective networks containing a state field. You can find more information about transaction states in this guide.

Each transaction includes an Operation field specifying the transfer's purpose or use case. The table below highlights the various operations and their corresponding destination address:

Operation Type Operation Field Value Description of Destination Address Field Value Transfer TRANSFER Externally owned account (EOA) that receives the transfer. Execute CONTRACT_EXECUTION Contract address. Deploy CONTRACT_DEPLOY Deployment of a transaction.

Paymaster Policies

The Gas Station Paymaster Policy is a set of rules and configurations that govern the gas fee sponsorship functionality, defining parameters and limitations for subsidizing gas fees on a blockchain network for your application end-users.

To enable gas fee sponsorship, a policy must be activated. For Testnet, Circle automatically preconfigures a policy for you at the time of account creation. For Mainnet, you are required to configure them. Please see Policy Management for more details.

Policy Attributes:

- Network:
- Specifies the blockchain network where the policy applies, ensuring compatibility and accurate gas fee calculation.
- Maximum Daily Spend:
- Sets the maximum allocated budget in USD for gas fee sponsorship within a 24-hour period. This limit helps manage and control the expenditure on gas fees.
- Maximum Transaction Spend:
- Optionally defines a cap on the amount of gas fees sponsored per individual transaction in USD. This allows for precise control over the financial commitment made for each transaction.

- Maximum Operations per Day:
- Optionally limits the number of daily operations (e.g., transactions) that can be sponsored. This parameter helps maintain a predetermined frequency and volume of sponsored operations.
- Blocklist:
- Optionally limits wallets that are eligible for gas sponsorship. This means that transactions initiated from these wallets will not have their gas fees covered. The blocklist allows for the selective exclusion of certain wallet addresses from receiving gas sponsorship.

Monitor Tokens

The Monitored Tokens feature in the Programmable Wallet empowers developers with enhanced control over viewing token balances and transactions. This feature enables developers to easily manage and filter out tokens unrelated to their use case from our API responses, thereby selectively screening which tokens end-users can see, providing a more streamlined and personalized experience. Using the APIs, developers can easily add tokens via their unique token IDs.

By default, the system does not have any Monitored Tokens. This means that all tokens are monitored without specific filtering or restrictions. The system will track and monitor all tokens by default, providing comprehensive visibility and monitoring for all token activities.

Currently, the monitor token supports native coins, ERC-20, ERC-721, and ERC-1155.

Monitor tokens attributes:

- Token ID:
- Each token deposited into the programmable wallet generates a specific token ID, which developers can obtain by inspecting the tokens in wallets.
- Token:
- Associated attributes for the token, including names, the contract standard, the blockchain, decimal point, symbol, and the token’s address. Updated3 days ago
- Table of Contents