

Chain Agnostic Wallet Interfaces

Pedro Gomes (WalletConnect)

Why are CAIPs important?

- Interoperability
- Compatibility
- Composibility

What problem is it solving?

CAIP-2 ChainId

chain = namespace + reference

Ethereum mainnet
eip155:1

Bitcoin mainnet
bip122:000000000019d6689c085ae165
831e93

Cosmos Hub
cosmos:cosmoshub-4

CAIP-10 AccountId

account = chain + address

Ethereum mainnet
eip155:1:0xab16a96d359ec26a11e2c2b3d8f8b894
2d5bfcdb

Bitcoin mainnet
bip122:000000000019d6689c085ae165831e93:128
Lkh3S7CkDTBZ8W7BbpsN3YYizJMp8p6

Cosmos Hub
cosmos:cosmoshub-4:cosmos1t2uflqwqe0fsj0shcfk
rvpukewcw40yjj6hdc0

How Wallet Interfaces use JSON-RPC?

CAIP-25 Provider Handshake

Handshake = scoping the Wallet API

```
{
  "id": 1,
  "jsonrpc": "2.0",
  "method": "caip_handshake",
  "params": {
    "chains": ["eip155:1"],
    "methods": [
      "eth_sendTransaction",
      "eth_signTransaction",
      "eth_sign",
      "personal_sign"
    ]
  }
}
```

CAIP-27 Provider Request

Request = executes a method

```
{
  "id": 1,
  "jsonrpc": "2.0",
  "method": "caip_request",
  "params": {
    "chainId": "eip155:1",
    "request": {
      "method": "personal_sign",
      "params": [
        "0x68656c6c6f20776f7226c642c207369676
        e2074657374206d65737361676521",
        "0xa89Df33a6f26c29ea23A9Ff582E865C03
        132b140"
      ]
    }
  }
}
```

How does WalletConnect v2.0 uses CAIPs?

WalletConnect v2.0

Quick Start

Dapps

Standalone Client

Cosmos Provider

Ethereum Provider

Polkadot Provider

Wallets

Kotlin Client (Android)

React-Native Client

Swift Client (iOS)

Protocol

Client Communication

Glossary

Reason Codes

Session Management

Technical Specification

UX Considerations

Migrating from v1.0

Mobile Linking

JSON-RPC Methods

API Reference

JSON-RPC Payloads

Now that WalletConnect 2.0 protocol is agnostic to the blockchain interface, it was important to dictate the rules upfront before session settlement to ensure a consistent end-user experience across multiple blockchain applications being interoperable with multiple blockchain wallets.

Therefore we will add support for CAIP-27 provider requests which will allow JSON-RPC requests to be accompanied by chainId target, this is translated as JSON-RPC request in WalletConnect between the two clients as follows:

```
interface CAIP27Request {
  id: number;
  jsonrpc: string;
  method: "caip_request";
  params: {
    request: {
      method: string;
      params: any;
    };
    chainId?: string;
  };
}
```

Copy

This allows a blockchain application to be connected to a blockchain wallet on multiple chains at the same time and target requests individually. While chainId is optional, is highly recommended that is used by all providers built on top of the WalletConnect 2.0 protocol.

Session Management

Contrary to its predecessor, WalletConnect 2.0 protocol is opinionated about session management on two fronts: lifecycles and duration.

Overview

Goals

Architecture

Requirements

Backwards Compatibility

Relay Protocol API

Out-of-Band Sequences

Pairing Signal

Pairing Proposal

Pairing Response

Pairing Settlement

Session Signal

Session Proposal

Session Response

Session Settlement

JSON-RPC Payloads

Session Management

Controller Client

Lifecycles


Duration




Persistent Storage

Client Communication



How does WalletConnect v2.0 uses CAIPs?

 **WalletConnect** 2.0 beta ▼

[Website](#) [GitHub](#)   

WalletConnect v2.0

Quick Start

- Dapps ▼
 - Standalone Client
 - Cosmos Provider
 - Ethereum Provider
 - Polkadot Provider
- Wallets ▼
 - Kotlin Client (Android)
 - React-Native Client
 - Swift Client (iOS)
- Protocol ▼
 - Client Communication**
 - Glossary
 - Reason Codes
 - Session Management
 - Technical Specification
 - UX Considerations
- Migrating from v1.0
- Mobile Linking
- JSON-RPC Methods >
- API Reference >

wc_sessionPayload

This request is used to relay payloads that match the list of methods agreed upon session settlement. Any requests sent with unauthorized methods will be immediately rejected by the client.

```
// request
interface WcSessionPayloadRequest {
  id: number;
  jsonrpc: "2.0";
  method: "wc_sessionPayload";
  params: {
    request: {
      method: string;
      params: any;
    };
    chainId?: string;
  };
}

// response
interface WcSessionPayloadResponse {
  id: number;
  jsonrpc: "2.0";
  result: any;
}
```

wc_sessionPing

This request is used to internally ping the other client to verify that is online. Pings are responded automatically internally by either client. By default, clients will throw a timeout if a ping is not responded within 30 seconds.

```
// request
interface WcSessionPingRequest {
  id: number;
```

JSON-RPC API

- wc_pairingApprove
- wc_pairingReject
- wc_pairingUpdate
- wc_pairingUpgrade
- wc_pairingDelete
- wc_pairingPayload
- wc_pairingPing
- wc_pairingNotification
- wc_sessionPropose
- wc_sessionApprove
- wc_sessionReject
- wc_sessionUpdate
- wc_sessionUpgrade
- wc_sessionDelete
- [wc_sessionPayload](#)
- wc_sessionPing
- wc_sessionNotification

Implementation drives Adoption

CAIP Index

- [CAIP-1 - CAIP Purpose and Guidelines](#)
- [CAIP-2 - Blockchain ID Specification](#)
- [CAIP-3 - Blockchain Reference for the EIP155 Namespace](#)
- [CAIP-4 - Blockchain Reference for the BIP122 Namespace](#)
- [CAIP-5 - Blockchain Reference for the Cosmos Namespace](#)
- [CAIP-6 - Blockchain Reference for the LIP9 Namespace](#)
- [CAIP-7 - Blockchain Reference for the EOSIO Namespace](#)
- [CAIP-10 - Account ID Specification](#)
- [CAIP-13 - Blockchain Reference for the Polkadot Namespace](#)
- [CAIP-19 - Asset Type and Asset ID Specification](#)
- [CAIP-20 - Asset Reference for the SLIP44 Asset Namespace](#)
- [CAIP-21 - Asset Reference for the ERC20 Asset Namespace](#)
- [CAIP-22 - Asset Reference for the ERC721 Asset Namespace](#)
- [CAIP-23 - Blockchain Reference for Filecoin Namespace](#)
- [CAIP-25 - Chain Agnostic Provider Handshake](#)
- [CAIP-26 - Blockchain Reference for the Tezos Namespace](#)
- [CAIP-27 - Chain Agnostic Provider Request](#)
- [CAIP-28 - Blockchain Reference for the Stellar Namespace](#)
- [CAIP-29 - Asset Reference for the ERC1155 Asset Namespace](#)
- [CAIP-30 - Blockchain Reference for the Solana Namespace](#)
- [CAIP-74 - CACAO: Chain Agnostic CApability Object](#)
- [CAIP-76 - Account Address for the Hedera namespace](#)

What's next? Javascript Provider API!

EIP-1193 Provider API

enable() a.k.a. eth_requestAccounts
request(method, params)
on("message", { type, data })
on("accountsChanged", accounts)
on("chainChanged", chainId)

CAIP-XX Provider API

connect() -> CAIP-25
request(method, params, chainId) -> CAIP-27
getAccounts() -> CAIP-xx
getChains() -> CAIP-xx
on("newAccounts", accounts) -> CAIP-xx
on("newChains", chains) -> CAIP-xx