ChainSafe Gaming SDK

Mobile & Desktop

Sign through Mobile and Desktop

```
string response = await Web3Wallet.Sign("hello");
print(response);
```

Hash Message (SHA3)

```
string message = "hello";
string hashedMessage = Web3Wallet.Sha3(message);
// 0x1c8aff950685c2ed4bc3174f3472287b56d9517b9c948127319a09a7a36deac8
print(hashedMessage);
```

Sending Transaction through Mobile and Desktop

```
// https://chainlist.org/
string chainId = "4"; // rinkeby
// account to send to
string to = "0xdD4c825203f97984e7867F11eeCc813A036089D1";
// value in wei
string value = "12300000000000000";
// data OPTIONAL
string data = "";
// gas limit OPTIONAL
string gasLimit = "";
// gas price OPTIONAL
string gasPrice = "";
// send transaction
string response = await Web3Wallet.SendTransaction(chainId, to, value, data, gasLimit, gasP
print(response);
```

Transfer ERC-1155 NFT Token through Mobile and Desktop

```
// https://chainlist.org/
string chainId = "4"; // rinkeby
// contract to interact with
```

```
string contract = "0x6b0bc2e986b0e70db48296619a96e9ac02c5574b";
// value in wei
string value = "0";
// abi in json format
string abi = "[ { \"inputs\": [ { \"internalType\": \"string\", \"name\": \"uri_\", \"type
// smart contract method to call
string method = "safeTransferFrom";
// account to sent tokens to
string toAccount = PlayerPrefs.GetString("Account");
// token id to send
string tokenId = "2";
// amount of tokens to send
string amount = "1";
// array of arguments for contract
string[] obj = { PlayerPrefs.GetString("Account"), toAccount, tokenId, amount, "0x" };
string args = JsonConvert.SerializeObject(obj);
// create data to interact with smart contract
string data = await EVM.CreateContractData(abi, method, args);
// gas limit OPTIONAL
string gasLimit = "";
// gas price OPTIONAL
string gasPrice = "";
// send transaction
string response = await Web3Wallet.SendTransaction(chainId, contract, value, data, gasLimi
print(response);
```

Transfer ERC-721 NFT Token through Mobile and Desktop

```
// https://chainlist.org/
string chainId = "4"; // rinkeby
// contract to interact with
string contract = "0xde458cd3deaa28ce67beefe3f45368c875b3ffd6";
// value in wei
string value = "0";
// abi in json format
string abi = "[{ \"inputs\": [ { \"internalType\": \"address\", \"name\": \"from\", \"type
// smart contract method to call
string method = "safeTransferFrom";
// account to send erc721 to
string toAccount = PlayerPrefs.GetString("Account");
// token id to send
string tokenId = "5";
// array of arguments for contract
string[] obj = { PlayerPrefs.GetString("Account"), toAccount, tokenId };
string args = JsonConvert.SerializeObject(obj);
// create data to interact with smart contract
string data = await EVM.CreateContractData(abi, method, args);
// gas limit OPTIONAL
string gasLimit = "";
// gas price OPTIONAL
string gasPrice = "";
```

```
// send transaction
string response = await Web3Wallet.SendTransaction(chainId, contract, value, data, gasLimi
print(response);
```

In Game Signing

`Web3PrivateKey` will allow games to sign and broadcast directly in game. No need for an external wallet. These methods will work for WebGL, Desktop and Mobile.

CAUTION: These methods will use raw private keys. Exposing private keys can be dangerous. Use with caution.

Tutorial

Get Account from Private Key

```
// private key of account
string privateKey = "0x78dae1a22c7507a4ed30c06172e7614eb168d3546c13856340771e63ad3c0081";
// get account from private key
string account = Web3PrivateKey.Address(privateKey);
print("Account: " + account);
```

Sign with Private Key

```
string privateKey = "0x78dae1a22c7507a4ed30c06172e7614eb168d3546c13856340771e63ad3c0081";
string message = "hello";
string response = Web3PrivateKey.Sign(privateKey, message);
print(response);
```

Send Transaction with Private Key

```
// private key of account
string privateKey = "0x78dae1a22c7507a4ed30c06172e7614eb168d3546c13856340771e63ad3c0081";
// set chain: ethereum, moonbeam, polygon etc
string chain = "ethereum";
// set network mainnet, testnet
string network = "rinkeby";
// account of player
string account = Web3PrivateKey.Address(privateKey);
// account to send to
string to = "0x428066dd8A212104Bc9240dCe3cdeA3D3A0f7979";
// value in wei
string value = "123";
// optional rpc url
string rpc = "";

string chainId = await EVM.ChainId(chain, network, rpc);
```

```
string gasPrice = await EVM.GasPrice(chain, network, rpc);
string data = "";
string gasLimit = "21000";
string transaction = await EVM.CreateTransaction(chain, network, account, to, value, data,
string signature = Web3PrivateKey.SignTransaction(privateKey, transaction, chainId);
string response = await EVM.BroadcastTransaction(chain, network, account, to, value, data,
print(response);
```

Transfer ERC-1155 NFT Token with Private Key

```
// private key of account
string privateKey = "0x78dae1a22c7507a4ed30c06172e7614eb168d3546c13856340771e63ad3c0081";
// set chain: ethereum, moonbeam, polygon etc
string chain = "ethereum";
// set network mainnet, testnet
string network = "rinkeby";
// smart contract method to call
string method = "safeTransferFrom";
// account of player
string account = Web3PrivateKey.Address(privateKey);
// ERC-1155 contract address
string contract = "0x3a8a85a6122c92581f590444449ca9e66d8e8f35";
// account to send to
string toAccount = "0x428066dd8A212104Bc9240dCe3cdeA3D3A0f7979";
// ERC-1155 token id
string tokenId = "5";
// amount of erc1155 tokens
string amount = "1";
// amount of wei to send
string value = "0";
// abi to interact with contract
string abi = "[ { \"inputs\": [ { \"internalType\": \"string\", \"name\": \"uri_\", \"type
// optional rpc url
string rpc = "";

string[] obj = { account, toAccount, tokenId, amount, "0x" };
string args = JsonConvert.SerializeObject(obj);
string chainId = await EVM.ChainId(chain, network, rpc);
string gasPrice = await EVM.GasPrice(chain, network, rpc);
string data = await EVM.CreateContractData(abi, method, args);
string gasLimit = "75000";
string transaction = await EVM.CreateTransaction(chain, network, account, contract, value,
string signature = Web3PrivateKey.SignTransaction(privateKey, transaction, chainId);
string response = await EVM.BroadcastTransaction(chain, network, account, contract, value,
print(response);
```

Transfer ERC-721 NFT Token with Private Key

```
// private key of account
```

```
string privateKey = "0x78dae1a22c7507a4ed30c06172e7614eb168d3546c13856340771e63ad3c0081";
// set chain: ethereum, moonbeam, polygon etc
string chain = "ethereum";
// set network mainnet, testnet
string network = "rinkeby";
// smart contract method to call
string method = "safeTransferFrom";
// account of player
string account = Web3PrivateKey.Address(privateKey);
// ERC-721 contract address
string contract = "0xae70a9accf2e0c16b380c0aa3060e9fba6718daf";
// account to send to
string toAccount = "0x428066dd8A212104Bc9240dCe3cdeA3D3A0f7979";
// ERC-721 token id
string tokenId = "2543";
// amount of wei to send
string value = "0";
// abi to interact with contract
string abi = "[{ \"inputs\": [ { \"internalType\": \"address\", \"name\": \"from\", \"type
string rpc = "";

// array of arguments for contract
string[] obj = { account, toAccount, tokenId, "0x" };
string args = JsonConvert.SerializeObject(obj);
string chainId = await EVM.ChainId(chain, network, rpc);
string gasPrice = await EVM.GasPrice(chain, network, rpc);
string data = await EVM.CreateContractData(abi, method, args);
string gasLimit = "75000";
string transaction = await EVM.CreateTransaction(chain, network, account, contract, value,
string signature = Web3PrivateKey.SignTransaction(privateKey, transaction, chainId);
string response = await EVM.BroadcastTransaction(chain, network, account, contract, value,
print(response);
```

Transfer ERC-20 Token with Private Key

```
// private key of account
string privateKey = "0x78dae1a22c7507a4ed30c06172e7614eb168d3546c13856340771e63ad3c0081";
// set chain: ethereum, moonbeam, polygon etc
string chain = "ethereum";
// set network mainnet, testnet
string network = "rinkeby";
// smart contract method to call
string method = "transfer";
// account of player
string account = Web3PrivateKey.Address(privateKey);
// smart contract address: https://rinkeby.etherscan.io/address/0xc7ad46e0b8a400bb3c915120
string contract = "0xc7ad46e0b8a400bb3c915120d284aafba8fc4735";
// account to send to
string toAccount = "0x428066dd8A212104Bc9240dCe3cdeA3D3A0f7979";
// amount of erc20 tokens to send. usually 18 decimals
string amount = "1000000000000000000";
```

```
// amount of wei to send
string value = "0";
// abi to interact with contract
string abi = "[ { \"inputs\": [ { \"internalType\": \"string\", \"name\": \"name_\", \"type
// optional rpc url
string rpc = "";

string[] obj = { toAccount, amount };
string args = JsonConvert.SerializeObject(obj);
string chainId = await EVM.ChainId(chain, network, rpc);
string gasPrice = await EVM.GasPrice(chain, network, rpc);
string data = await EVM.CreateContractData(abi, method, args);
string gasLimit = "75000";
string transaction = await EVM.CreateTransaction(chain, network, account, contract, value,
string signature = Web3PrivateKey.SignTransaction(privateKey, transaction, chainId);
string response = await EVM.BroadcastTransaction(chain, network, account, contract, value,
print(response);
```