

ChainSafe Gaming SDK

Mobile & Desktop

Sign through Mobile and Desktop

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

Hash Message (SHA3)

```
1 string message = "hello";
2 string hashedMessage = Web3Wallet.Sha3(message);
3 // 0x1c8aff950685c2ed4bc3174f3472287b56d9517b9c948127319a09a7a36deac8
4 print(hashedMessage);
```

Sending Transaction through Mobile and Desktop

```
1 // https://chainlist.org/
2 string chainId = "4"; // rinkeby
3 // account to send to
4 string to = "0xD4c825203f97984e7867F11eeCc813A036089D1";
5 // value in wei
6 string value = "1230000000000000000";
7 // data OPTIONAL
8 string data = "";
9 // gas limit OPTIONAL
10 string gasLimit = "";
11 // gas price OPTIONAL
12 string gasPrice = "";
13 // send transaction
14 string response = await Web3Wallet.SendTransaction(chainId, to, value, data, gasLimit, gasPrice);
15 print(response);
```

Transfer ERC-1155 NFT Token through Mobile and Desktop

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

```

4 string contract = "0x6b0bc2e986b0e70db48296619a96e9ac02c5574b";
5 // value in wei
6 string value = "0";
7 // abi in json format
8 string abi = "[ { \"inputs\": [ { \"internalType\": \"string\", \"name\": \"uri_\", \"type\":
9 // smart contract method to call
10 string method = "safeTransferFrom";
11 // account to sent tokens to
12 string toAccount = PlayerPrefs.GetString("Account");
13 // token id to send
14 string tokenId = "2";
15 // amount of tokens to send
16 string amount = "1";
17 // array of arguments for contract
18 string[] obj = { PlayerPrefs.GetString("Account"), toAccount, tokenId, amount, "0x" };
19 string args = JsonConvert.SerializeObject(obj);
20 // create data to interact with smart contract
21 string data = await EVM.CreateContractData(abi, method, args);
22 // gas limit OPTIONAL
23 string gasLimit = "";
24 // gas price OPTIONAL
25 string gasPrice = "";
26 // send transaction
27 string response = await Web3Wallet.SendTransaction(chainId, contract, value, data, gasLimit);
28 print(response);

```

Transfer ERC-721 NFT Token through Mobile and Desktop

```

1 // https://chainlist.org/
2 string chainId = "4"; // rinkeby
3 // contract to interact with
4 string contract = "0xde458cd3deaa28ce67beefe3f45368c875b3ffd6";
5 // value in wei
6 string value = "0";
7 // abi in json format
8 string abi = "[{ \"inputs\": [ { \"internalType\": \"address\", \"name\": \"from\", \"type\":
9 // smart contract method to call
10 string method = "safeTransferFrom";
11 // account to send erc721 to
12 string toAccount = PlayerPrefs.GetString("Account");
13 // token id to send
14 string tokenId = "5";
15 // array of arguments for contract
16 string[] obj = { PlayerPrefs.GetString("Account"), toAccount, tokenId };
17 string args = JsonConvert.SerializeObject(obj);
18 // create data to interact with smart contract
19 string data = await EVM.CreateContractData(abi, method, args);
20 // gas limit OPTIONAL
21 string gasLimit = "";
22 // gas price OPTIONAL
23 string gasPrice = "";

```

```
24 // send transaction
25 string response = await Web3Wallet.SendTransaction(chainId, contract, value, data, gasLimit);
26 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

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

Sign with Private Key

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

Send Transaction with Private Key

```
1 // private key of account
2 string privateKey = "0x78dae1a22c7507a4ed30c06172e7614eb168d3546c13856340771e63ad3c0081";
3 // set chain: ethereum, moonbeam, polygon etc
4 string chain = "ethereum";
5 // set network mainnet, testnet
6 string network = "rinkeby";
7 // account of player
8 string account = Web3PrivateKey.Address(privateKey);
9 // account to send to
10 string to = "0x428066dd8A212104Bc9240dCe3cdeA3D3A0f7979";
11 // value in wei
12 string value = "123";
13 // optional rpc url
14 string rpc = "";
15
16 string chainId = await EVM.ChainId(chain, network, rpc);
```

```

17 string gasPrice = await EVM.GasPrice(chain, network, rpc);
18 string data = "";
19 string gasLimit = "21000";
20 string transaction = await EVM.CreateTransaction(chain, network, account, to, value, data,
21 string signature = Web3PrivateKey.SignTransaction(privateKey, transaction, chainId);
22 string response = await EVM.BroadcastTransaction(chain, network, account, to, value, data,
23 print(response);

```

Transfer ERC-1155 NFT Token with Private Key

```

1 // private key of account
2 string privateKey = "0x78dae1a22c7507a4ed30c06172e7614eb168d3546c13856340771e63ad3c0081";
3 // set chain: ethereum, moonbeam, polygon etc
4 string chain = "ethereum";
5 // set network mainnet, testnet
6 string network = "rinkeby";
7 // smart contract method to call
8 string method = "safeTransferFrom";
9 // account of player
10 string account = Web3PrivateKey.Address(privateKey);
11 // ERC-1155 contract address
12 string contract = "0x3a8a85a6122c92581f590444449ca9e66d8e8f35";
13 // account to send to
14 string toAccount = "0x428066dd8A212104Bc9240dCe3cdeA3D3A0f7979";
15 // ERC-1155 token id
16 string tokenId = "5";
17 // amount of erc1155 tokens
18 string amount = "1";
19 // amount of wei to send
20 string value = "0";
21 // abi to interact with contract
22 string abi = "[ { \"inputs\": [ { \"internalType\": \"string\", \"name\": \"uri_\", \"type\":
23 // optional rpc url
24 string rpc = "";
25
26 string[] obj = { account, toAccount, tokenId, amount, "0x" };
27 string args = JsonConvert.SerializeObject(obj);
28 string chainId = await EVM.ChainId(chain, network, rpc);
29 string gasPrice = await EVM.GasPrice(chain, network, rpc);
30 string data = await EVM.CreateContractData(abi, method, args);
31 string gasLimit = "75000";
32 string transaction = await EVM.CreateTransaction(chain, network, account, contract, value,
33 string signature = Web3PrivateKey.SignTransaction(privateKey, transaction, chainId);
34 string response = await EVM.BroadcastTransaction(chain, network, account, contract, value,
35 print(response);

```

Transfer ERC-721 NFT Token with Private Key

```

1 // private key of account

```

```
2 string privateKey = "0x78dae1a22c7507a4ed30c06172e7614eb168d3546c13856340771e63ad3c0081";
3 // set chain: ethereum, moonbeam, polygon etc
4 string chain = "ethereum";
5 // set network mainnet, testnet
6 string network = "rinkeby";
7 // smart contract method to call
8 string method = "safeTransferFrom";
9 // account of player
10 string account = Web3PrivateKey.Address(privateKey);
11 // ERC-721 contract address
12 string contract = "0xae70a9accf2e0c16b380c0aa3060e9fba6718daf";
13 // account to send to
14 string toAccount = "0x428066dd8A212104Bc9240dCe3cdeA3D3A0f7979";
15 // ERC-721 token id
16 string tokenId = "2543";
17 // amount of wei to send
18 string value = "0";
19 // abi to interact with contract
20 string abi = "[{ \"inputs\": [ { \"internalType\": \"address\", \"name\": \"from\", \"type\": \"address\" }, { \"internalType\": \"uint256\", \"name\": \"value\", \"type\": \"uint256\" } ], \"name\": \"safeTransferFrom\", \"outputs\": [ { \"internalType\": \"bool\", \"name\": \"success\", \"type\": \"bool\" } ], \"stateMutability\": \"nonpayable\", \"type\": \"function\" }]";
21 string rpc = "";
22
23 // array of arguments for contract
24 string[] obj = { account, toAccount, tokenId, "0x" };
25 string args = JsonConvert.SerializeObject(obj);
26 string chainId = await EVM.ChainId(chain, network, rpc);
27 string gasPrice = await EVM.GasPrice(chain, network, rpc);
28 string data = await EVM.CreateContractData(abi, method, args);
29 string gasLimit = "75000";
30 string transaction = await EVM.CreateTransaction(chain, network, account, contract, value, gasPrice, gasLimit);
31 string signature = Web3PrivateKey.SignTransaction(privateKey, transaction, chainId);
32 string response = await EVM.BroadcastTransaction(chain, network, account, contract, value, gasPrice, gasLimit);
33 print(response);
```

Transfer ERC-20 Token with Private Key

```
1 // private key of account
2 string privateKey = "0x78dae1a22c7507a4ed30c06172e7614eb168d3546c13856340771e63ad3c0081";
3 // set chain: ethereum, moonbeam, polygon etc
4 string chain = "ethereum";
5 // set network mainnet, testnet
6 string network = "rinkeby";
7 // smart contract method to call
8 string method = "transfer";
9 // account of player
10 string account = Web3PrivateKey.Address(privateKey);
11 // smart contract address: https://rinkeby.etherscan.io/address/0xc7ad46e0b8a400bb3c9151200
12 string contract = "0xc7ad46e0b8a400bb3c915120d284aaafba8fc4735";
13 // account to send to
14 string toAccount = "0x428066dd8A212104Bc9240dCe3cdeA3D3A0f7979";
15 // amount of erc20 tokens to send. usually 18 decimals
16 string amount = "1000000000000000000";
```

```
17 // amount of wei to send
18 string value = "0";
19 // abi to interact with contract
20 string abi = "[ { \"inputs\": [ { \"internalType\": \"string\", \"name\": \"name_\", \"type\": \"string\" } ], \"name\": \"set\", \"outputs\": [ { \"internalType\": \"string\", \"name\": \"name\", \"type\": \"string\" } ], \"stateMutability\": \"nonpayable\", \"type\": \"function\" } ]";
21 // optional rpc url
22 string rpc = "";
23
24 string[] obj = { toAccount, amount };
25 string args = JsonConvert.SerializeObject(obj);
26 string chainId = await EVM.ChainId(chain, network, rpc);
27 string gasPrice = await EVM.GasPrice(chain, network, rpc);
28 string data = await EVM.CreateContractData(abi, method, args);
29 string gasLimit = "75000";
30 string transaction = await EVM.CreateTransaction(chain, network, account, contract, value, gasPrice, gasLimit);
31 string signature = Web3PrivateKey.SignTransaction(privateKey, transaction, chainId);
32 string response = await EVM.BroadcastTransaction(chain, network, account, contract, value, gasPrice, gasLimit);
33 print(response);
```