---

title: SUAVE Standard Library description: A collection of contracts and libraries to empower SUAPP developers and make life easier for builders. keywords: - application - build - suave - solidity

---

suave-std is a collection of Solidity contracts and libraries to help you create Suapps. It is included as a submodule in the SUAPP Examples repo, which we used in the previous tutorial.

In this tutorial, we will be diving deeper into the different capabilities offered bysuave-std.

:::info

suave-std is the main entrypoint for developers who are interacting with SUAVE. It is a place to create building blocks, sdks, and common patterns that are useful to SUAVE developers.

:::

Some functionalities that suave-std will help you handle include:

- Transaction parsing and construction
- JSON object decoding and encoding
- ♾ A wrapper for all the precompiles to make calling any of them more convenient

Transactions

Encoding and decoding transactions from other chains is often necessary when matching different transactions in anorder flow auction, or turning bundles into (partial) blocks in a block building contract, or weaving together different transactions on different chains for your unique use case.

Having to encode and decode such transactions is a pain, which is exactly the sort of scenario wheresuave-std can help:

```solidity import "suave-std/Transactions.sol";

contract Example { function example() { Transactions.Legacy memory legacyTxn0 = Transactions.Legacy({ to: address(0x095E7BAea6a6c7c4c2DfeB977eFac326aF552d87), gas: 50000, gasPrice: 10, value: 10, ... });

    // Encode to RLP
    bytes memory rlp = Transactions.encodeRLP(legacyTxn0);

    // Decode from RLP
    Transactions.Legacy memory legacyTxn1 = Transactions.decodeRLP(rlp);
}

} ```

## SUAVE Enabled Tests

It can also be difficult to test specific precompiles when using Forge. In order to make this possible,suave-std provides a "SUAVE-enabled" test contract, which you can import and use like this:

```solidity // SPDX-License-Identifier: Unlicense pragma solidity ^0.8.13;

import "forge-std/Test.sol"; import "suave-std/Test.sol"; import "suave-std/Suave.sol";

contract TestForge is Test, SuaveEnabled { address[] public addressList = [0xC8df3686b4Afb2BB53e60EAe97EF043FE03Fb829];

function testConfidentialStore() public {
    Suave.DataRecord memory record = Suave.newDataRecord(0, addressList, addressList, "namespace");

    bytes memory value = abi.encode("suave works with forge!");
    Suave.confidentialStore(record.id, "key1", value);

    bytes memory found = Suave.confidentialRetrieve(record.id, "key1");
    assertEq(keccak256(found), keccak256(value));
}

} ```

## MevShare

While not yet in the main branch ofsuave-geth, the MevShare library illustrates all of the major functionality we've discussed

so far.

## MevBoost

The same is true for the MevBoost library.

The goal of these two libraries is to illustrate how products and services which currently rely on a known set of actors to run can be encoded as smart contracts on SUAVE, and made to work on an open, permissionless network.

It is in this sense that we make the claim that applications built on SUAVE can interface with any component in the MEV supply network.