

ink! Smart Contract Deployment and Interactions

[Shiden](#) Network is a multi-chain decentralized application layer on Kusama Network. [Shibuya](#) is the Shiden's parachain testnet with EVM functionalities. We choose it for deployment as Shiden supports EVM, Wasm, and Layer2 solutions. We prepared this [shared document with screenshots](#) to illustrate the flow of deploying the contract then step-by-step flow of ordering food on the blockchain. Here's the expected output when running test from the docker image:

Deployment process

Deployment

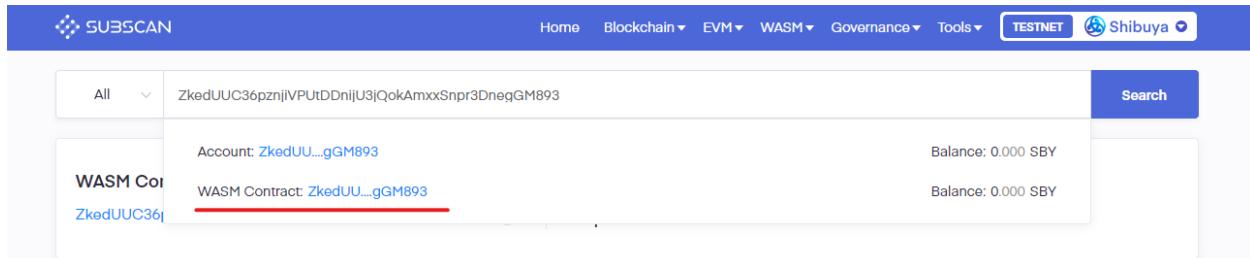
```
swanky contract deploy foodorder --account deploy --gas 100000 --network shibuya
```

Log:

- ✓ Initialising OK
 - ✓ Getting WASM OK
 - ✗ Connecting to node2023-08-11 11:47:18 API/INIT: shibuya/105: Not decorating runtime apis without matching versions: EthereumRuntimeRPCApi/5 (4 known)

 - ✓ Connecting to node OK
 - ✓ Deploying OK
 - ✓ Writing config OK
- Contract deployed!
- Contract address: Yn1dHJTbKuMhA6rLLsRXQtDu4mSFGC6xtvDTueNz1axJ5Dz

After successfully deployed, you can check the deployed contract on the shibuya blockexplorer <https://shibuya.subscan.io/>.



The screenshot shows the Subscan block explorer interface for the Shibuya testnet. The top navigation bar includes links for Home, Blockchain, EVM, WASM, Governance, Tools, TESTNET, and Shibuya. A search bar is located at the top right. The main content area displays a table with the following data:

WASM Contract	Account: ZkedUU...gGM893	Balance: 0.000 SBY
ZkedUUC36	WASM Contract: ZkedUU...gGM893	Balance: 0.000 SBY

WASM Contract   

ZkedUUC36pznjiVPUTDDnijU3jQokAmxxSnpr3DnegGM893 

Owner:	ZvAlqKqv5HftVhgH4ze7oPN5f8uMmktY3TZvY3jqBgZsUuC
Code Hash:	0x9ce71f...b88a65d0
Deposit:	0 SBY

WASM Transactions Events Timeline **Contract** 

Since the wasm contract bytecode compiled on different machines and operating systems differs, we use docker to build WASM contracts deterministically. Users can use various machines and operating systems to generate the same WASM bytecode.
We now provide the following [docker image](#) to compile. This ensures that Subscan is consistent with the contract deployer compilation environment.

Contract Address Verify Other Contract

Interacting with the smart contract

Click the button “Add an existing contract”

add an existing contract X

 FOODORDER
ZkedUUC3...

contract address
ZkedUUC36pznjiVPUTDDnijU3jQokAmxxSnpr3DnegGM893

contract name
FoodOrder

contract abi
click to select or drag and drop a JSON file

 Save

Type the contract address to the relevant input and upload [the ABI JSON file](#) in the contract abi box

add an existing contract



FOODORDER
ZkedUUC3...

contract address

ZkedUUC36pznjiVPUtDDnijU3jQokAmxxSnpr3DnegGM893

contract name

FoodOrder

contract abi

Constructors (1) ▾

Messages (31) ▾

remove abi

Save

After that, you can see a new smart contract named FOODORDER

The screenshot shows the Substrate UI interface for the Shibuya Testnet. The top navigation bar includes the network name "Shibuya Testnet" with the address "shibuya/105" and block number "#4,404,402", along with dropdown menus for "Accounts", "Network", and "Governance". A sidebar on the left has a "Contracts" section, which is currently active, indicated by a blue bracket and underline. The main content area is titled "Contracts" and displays the following information:

- addresses**: 0
- contracts**:
 - CONTRACT1**: Messages (31) ▾
 - FOODORDER**: Messages (31) ▾
- code hashes**: No code hashes available

Here, you can see a list of all messages of the smart contract, some read(able) some exec(cutable). Now we can manually simulate a food order flow in steps.

contracts

 CONTRACT1	Messages (31) ▾
 FOODORDER	Messages (31) ▾

▶ exec `courierService::pickupDelivery (deliveryId: PickupDeliveryInput1): Result<Result<u64, LogicImplsTypesFoodOrderError>, InkPrimitivesLangError>` 
No documentation provided

▶ exec `customerService::acceptDelivery (deliveryId: AcceptDeliveryInput1): Result<Result<u64, LogicImplsTypesFoodOrderError>, InkPrimitivesLangError>` 
No documentation provided

▶ exec `customerService::addCustomer (customerName: AddCustomerInput1, customerAddress: AddCustomerInput2, phoneNumber: AddCustomerInput3): Result<Result<u64, LogicImplsTypesFoodOrderError>, InkPrimitivesLangError>` 
No documentation provided

▶ exec `customerService::submitOrder (foodId: SubmitOrderInput1, deliveryAddress: SubmitOrderInput2): Result<Result<u64, LogicImplsTypesFoodOrderError>, InkPrimitivesLangError>` 
No documentation provided

▶ exec `managerService::addCourier (courierAccount: AddCourierInput1, courierName: AddCourierInput2, courierAddress: AddCourierInput3, phoneNumber: AddCourierInput4): Result<Result<u64, LogicImplsTypesFoodOrderError>, InkPrimitivesLangError>` 
No documentation provided

▶ exec `managerService::addRestaurant (restaurantAccount: AddRestaurantInput1, restaurantName: AddRestaurantInput2, restaurantAddress: AddRestaurantInput3, phoneNumber: AddRestaurantInput4): Result<Result<u64, LogicImplsTypesFoodOrderError>, InkPrimitivesLangError>` 
No documentation provided

▶ exec `managerService::changeFeeRate (rate: ChangeFeeRateInput1): Result<Result<Text, LogicImplsTypesFoodOrderError>, InkPrimitivesLangError>` 
No documentation provided

▶ exec `managerService::changeManager (newAccount: ChangeManagerInput1): Result<Result<Text, LogicImplsTypesFoodOrderError>, InkPrimitivesLangError>` 
No documentation provided

▶ exec `restaurantService::addFood (foodName: AddFoodInput1, description: AddFoodInput2, price: AddFoodInput3, eta: AddFoodInput4): Result<Result<u64, LogicImplsTypesFoodOrderError>, InkPrimitivesLangError>` 
No documentation provided

1. Add a restaurant.

call a contract

contract to use
FOODORDER

call from account
No accounts are available for selection.

message to send
managerService::addRestaurant (restaurantAccount: AddRestaurantInput1, restaurantName: AddRestaurantInput2, restaurantAddress: AddRestaurantInput3, phoneNumber: AddRestaurantInput4): Result<Result<u64, LogicImplsTypesFoodOrderError>, InkPrimitivesLangError>

restaurantAccount: AddRestaurantInput1
FOODORDER

restaurantName: AddRestaurantInput2
<any string>

restaurantAddress: AddRestaurantInput3
<any string>

phoneNumber: AddRestaurantInput4
<any string>

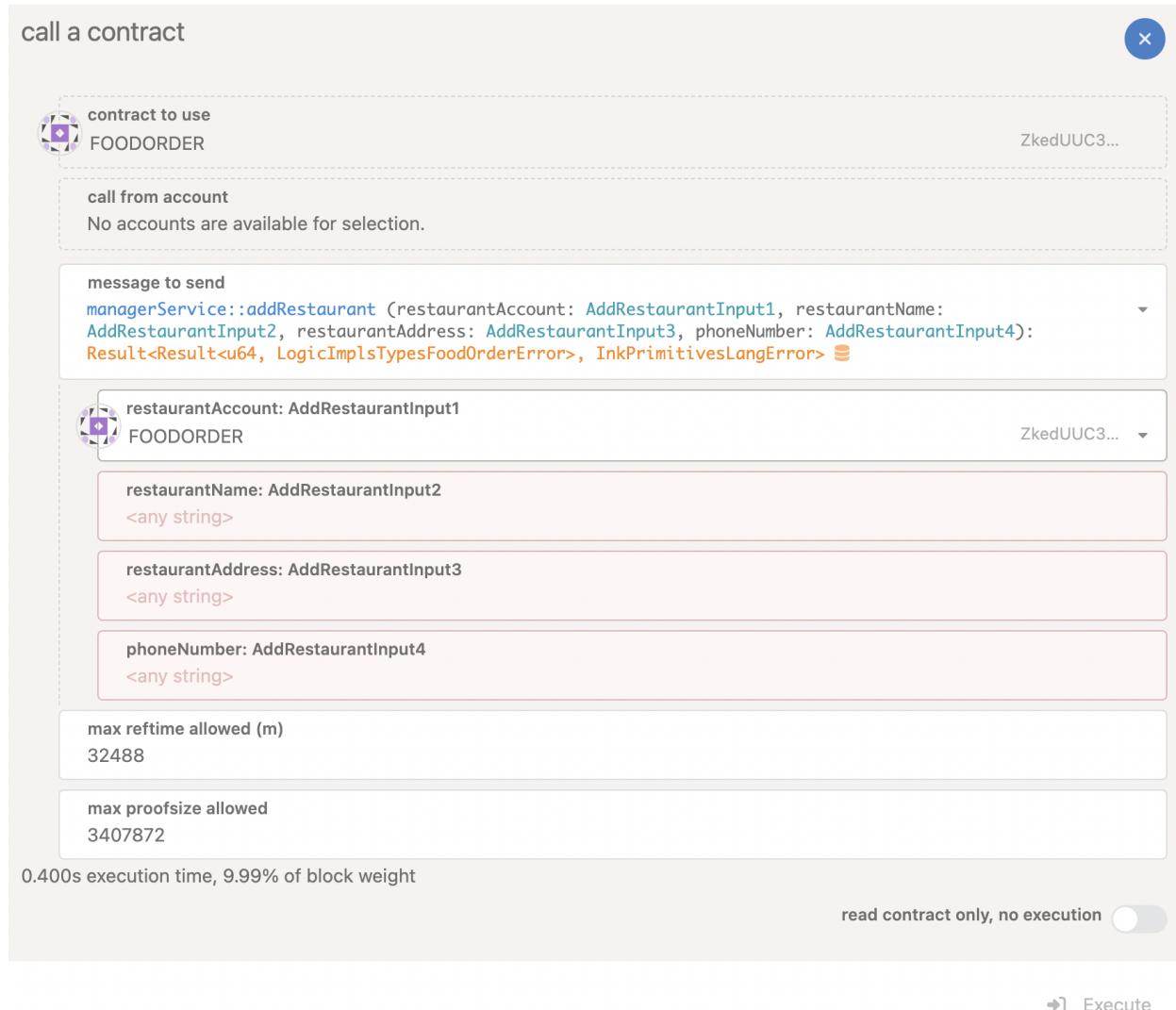
max reftime allowed (m)
32488

max proofsize allowed
3407872

0.400s execution time, 9.99% of block weight

read contract only, no execution

Execute



After you added a restaurant, you can check the result here by invoking the `getRestaurantAll` message. Here, you can set from (0) and to (e.g. 10) as parameters to paginate through existing restaurants listing in contract storage:

call a contract

contract to use
FOODORDER

call from account
ACCOUNT 1 (SUBWALLET-JS)

transferrable 0.0000 sby
ZPJU7ALjd...

message to send
getService::getRestaurantAll (from: GetRestaurantAllInput1, to: GetRestaurantAllInput2):
Result<Result<Vec<LogicImplsTypesRestaurant>, LogicImplsTypesFoodOrderError>, InkPrimitivesLangError>

from: GetRestaurantAllInput1
0

to: GetRestaurantAllInput2
10

max reftime allowed (m)
1

max proofsize allowed
1000000

max read gas

0.400s execution time, 9.99% of block weight

Call results ▲

getservice::getrestaurantall (from: 0, to: 10): result<result<vec<logicimplstypesrestaurant>, inkprimitivestypeslangerror> {Ok: {restaurantAccount: ZWexd44p3IVqbn5JDDXtcZvSULYqemq5423r2siVERab9b restaurantName: name1 restaurantAddress: 123 main st phoneNumber: 123456789 }}}

→ Read

2. Get free SBY tokens from a faucet.

Before you are trying to execute a transaction, you will need to get native tokens. You can get faucet from [here](#).

The screenshot shows the Shibuya wallet interface. The left sidebar has a dark blue background with white icons and text. The 'Assets' icon is highlighted with a light blue background. The main content area has a dark blue background with white text. The title 'Assets' is at the top. Below it, 'Astar Native Account' is displayed. The account section shows 'Account 1 (subwallet-js)' with address 'ZPJu7A.....Yrakso' and balance '0 USD'. Below this is another account section for 'SBY' with address 'Shibuya' and balance '10 SBY (Transferable) 10 SBY'. The bottom of the main content area has a dark blue background with white text, showing the title 'Assets' again.

Account	Address	Balance
Account 1 (subwallet-js)	ZPJu7A.....Yrakso	0 USD
SBY	Shibuya	10 SBY (Transferable) 10 SBY

If it succeeds, then you can see such a result.

3. Add a customer.

contract to use
FOODORDER

ZkedUUC3...

call from account
ACCOUNT 1 (SUBWALLET-JS)

transferrable 10.0000 SBY
ZPJU7ALjd... ▾

message to send
customerService::addCustomer (customerName: `AddCustomerInput1`, customerAddress: `AddCustomerInput2`,
 phoneNumber: `AddCustomerInput3`): `Result<Result<u64, LogicImplsTypesFoodOrderError>,
 InkPrimitivesLangError>` 📜

customerName: `AddCustomerInput1`
Customer2

customerAddress: `AddCustomerInput2`
789 Broadway Way

phoneNumber: `AddCustomerInput3`
0987654321

max reftime allowed (m)
32488

max proofsize allowed
3407872

.400s execution time, 9.99% of block weight

read contract only, no execution

➡ Execute

After you complete the form, then click the “Sign and Submit” button from your Sub wallet browser popup to pay the gas fees.

authorize transaction

contracts.call
Makes a call to an account, optionally transferring some balance
Fees of 2.1102 milli SBY will be applied to the submission

sending from my account
ACCOUNT 1 (SUBWALLET-JS)

ZPJU7ALjd...

The details of the transaction including the type, the description (as available from the chain metadata) as well as any parameters and fee estimations (as available) for the specific type of call.

do not include a tip for the block author

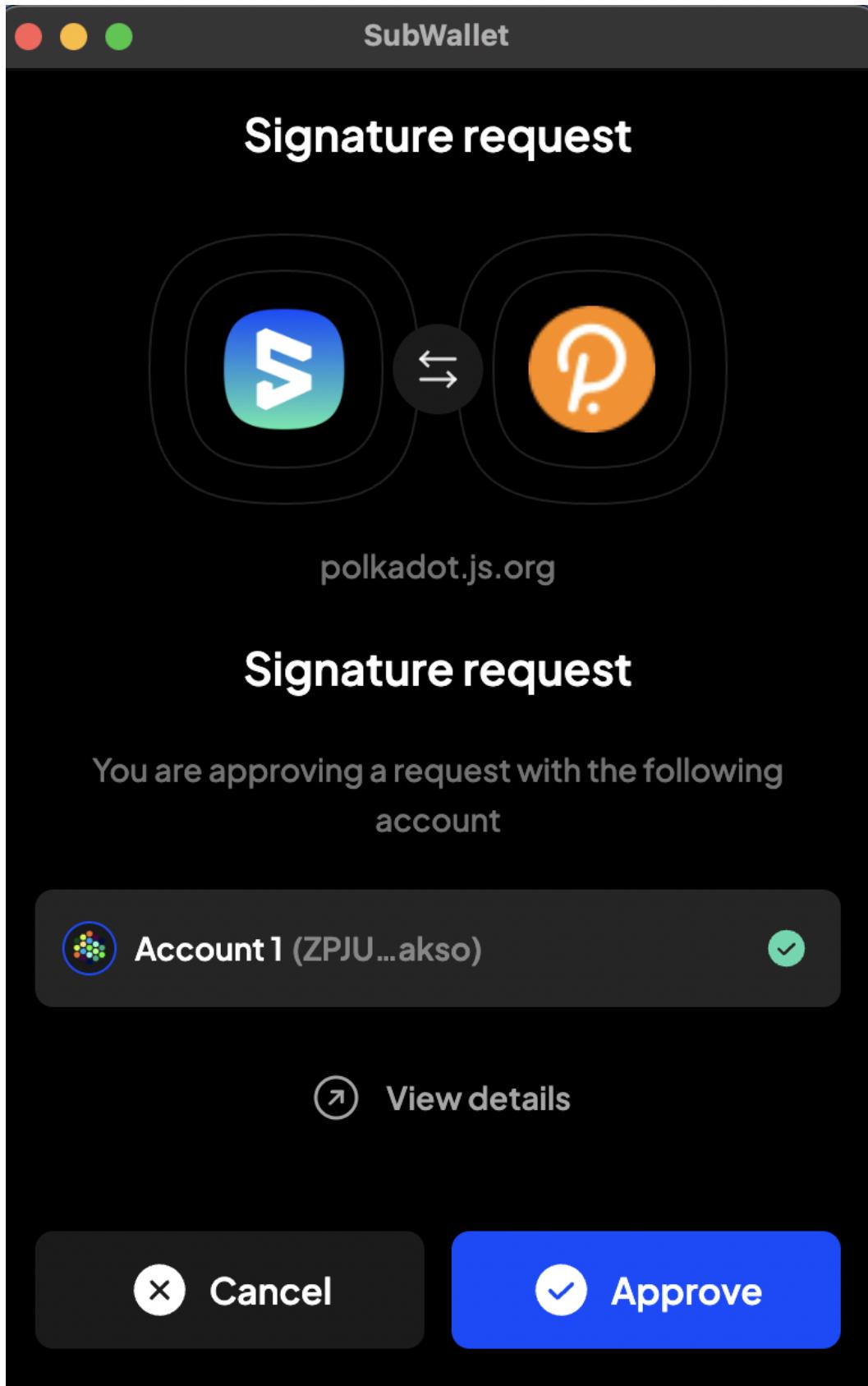
The sending account that will be used to send this transaction. Any applicable fees will be paid by this account.

call hash
0x4a7acae4c61f73dec85e7d83870445b4df6b229101a0e19cf318c9e0c8e4821a

➡ The call hash as calculated for this transaction

sign and submit

➡ Sign and Submit



contract to use


FOODORDER

ZkedUUC3...

call from account


ACCOUNT 1 (SUBWALLET-JS)

transferrable 9.9978 sBV

ZPJU7ALjd...

message to send

```
getService::getCustomerAll (from: GetCustomerAllInput1, to: GetCustomerAllInput2):
Result<Result<Vec<LogicImplsTypesCustomer>, LogicImplsTypesFoodOrderError, InkPrimitivesLangError>
```

▼

from: GetCustomerAllInput1

0

to: GetCustomerAllInput2

10

max reftime allowed (m)

1

max read gas

max proofsize allowed

1000000

0.400s execution time, 9.99% of block weight

Call results ▲

▼



```
getservice::getcustomerall (from: 0, to: 10): result<result<vec<logicimplstypescustomer>, InkPrimitivesLangError>
{OK{[ok: {customerAccount: ZPJU7ALjdVXfyp, customerName: "Customer1", customerAddress: "4534 main st", phoneNumber: "654321987"}, {customerAccount: ZPJU7ALjdVXfyp, customerName: "Customer2", customerAddress: "789 Broadway Way", phoneNumber: "0987654321"}]}{}}
```

✖

➡ Read

4. List food items

contract to use
FOODORDER

call from account
ACCOUNT 1 (SUBWALLET-JS)

transferrable 9.9978 sBV
ZPJU7ALjd...

message to send
getService::getFoodAll (from: GetFoodAllInput1, to: GetFoodAllInput2):
Result<Result<Vec<LogicImplsTypesFood>, LogicImplsTypesFoodOrderError>, InkPrimitivesLangError>

from: GetFoodAllInput1
0

to: GetFoodAllInput2
10

max reftime allowed (m)
1

max read gas

max proofsize allowed
1000000

0.400s execution time, 9.99% of block weight

Call results ▲

getservice::getfoodall (from: 0, to: 10): result<result<vec<logicimplstypesfood>, logicimplstypesfoodordererror>, inkprimitiveslangerror> 8/11/2023 11:01:58 am
[OK. [{ foodname: food1, restaurantid: 1, description: food1 price: 1,000 eta: 2 timestamp: 1,691,776,038, 345 }]]

 Read

5. Consumer submits a food order

contract to use
FOODORDER

ZkedUUC3...

call from account
ACCOUNT 1 (SUBWALLET-JS)

transferrable 9.9978 SBY
ZPJU7ALjd... ▾

message to send
customerService::submitOrder (foodId: SubmitOrderInput1, deliveryAddress: SubmitOrderInput2):
Result<Result<u64, LogicImplsTypesFoodOrderError>, InkPrimitivesLangError> ⚡

foodId: SubmitOrderInput1
1

deliveryAddress: SubmitOrderInput2
345 captain lane, CA

value
0.0000000000000001| SBY

max reftime allowed (m)
32488

max proofsize allowed
3407872

400s execution time, 9.99% of block weight

read contract only, no execution

 Execute

authorize transaction

contracts.call
Makes a call to an account, optionally transferring some balance
Fees of 2.0302 milli SBY will be applied to the submission

authorizing account
ACCOUNT 1 (SUBWALLET-JS) ZPJU7ALjd...

The details of the transaction including the type, the description (as available from the chain metadata) as well as any parameters and fee estimations (as available) for the specific type of call.

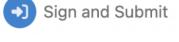
do not include a tip for the block author

The sending account that will be used to send this transaction. Any applicable fees will be paid by this account.

call hash
0x4579f1b008eeab3b126fadccb97c3487c3eba4b94678be9d871148e5738faef 

Adding an optional tip to the transaction could allow for higher priority, especially when the chain is busy.

The call hash as calculated for this transaction

 Sign and Submit

sign and submit

6. Confirm a order

contract to use
FOODORDER

call from account
ACCOUNT 1 (SUBWALLET-JS)

message to send
restaurantService::confirmOrder (orderId: ConfirmOrderInput1, eta: ConfirmOrderInput2): Result<Result<u64, LogicImplsTypesFoodOrderError>, InkPrimitivesLangError>

orderId: ConfirmOrderInput1
1

eta: ConfirmOrderInput2
200

max reftime allowed (m)
32488

max proofsize allowed
3407872

0.400s execution time, 9.99% of block weight

read contract only, no execution

 Execute

After you confirm an order, let's check its status.

contract to use
FOODORDER

ZkedUUC36...

call from account
ACCOUNT 1 (SUBWALLET-JS)

transferrable 15.3547 SBY
ZWexd44p3r...

message to send
getService::getOrderAll (from: GetOrderAllInput1, to: GetOrderAllInput2):
Result<Result<Vec<LogicImplsTypesOrder>, LogicImplsTypesFoodOrderError>, InkPrimitivesLangError>

from: GetOrderAllInput1
0

to: GetOrderAllInput2
10

max reftime allowed (m)
1

max proofsize allowed
1000000

max read gas

.400s execution time, 9.99% of block weight

all results ▲

```
getservice::getorderall (from: 0, to: 10): result<result<vec<logicimplstypesorder>, logicimplstypesfoodordererror>, inkprimitiveslangerror>
{ Ok: { Ok: [ { foodId: 1 restaurantId: 1 customerId: 1 courierId: 0 deliveryAddress: 123 main st status: OrderConfirmed timestamp: 1,691,776,638,103 price: 1,000 eta: 200 } { foodId: 1 restaurantId: 1 customerId: 2 courierId: 0 deliveryAddress: 345 captain lane, CA status: OrderSubmitted timestamp: 1,691,777,304,084 price: 1,000 eta: 0 } ] } }
```

Read

7. Finish cooking

After food is cooked, you can call `finish_cook`.

contract to use
FOODORDER

ZkedUUC36...

call from account
ACCOUNT 1 (SUBWALLET-JS)

transferrable 15.3547 sBY
ZWexd44p3r... ▾

message to send
restaurantService::finishCook (orderId: FinishCookInput1): Result<Result<u64, LogicImplsTypesFoodOrderError>, InkPrimitivesLangError> ⚒

orderId: FinishCookInput1
1

max reftime allowed (m)
32488

max proofsize allowed
3407872

0.400s execution time, 9.99% of block weight

read contract only, no execution

And check the order status.

call from account
ACCOUNT 1 (SUBWALLET-JS)

transferrable 15.3530 SBY
ZWexd44p3r... ▾

message to send
getService::getOrderAll (from: GetOrderAllInput1, to: GetOrderAllInput2):
Result<Result<Vec<LogicImplsTypesOrder>, LogicImplsTypesFoodOrderError>, InkPrimitivesLangError>

from: GetOrderAllInput1
0

to: GetOrderAllInput2
10

max reftime allowed (m)
1

max read gas

max proofsize allowed
1000000

0.400s execution time, 9.99% of block weight

Call results ▾

```
getservice::getorderall (from: 0, to: 10): result<result<vec<logicimplstypesorder>, logicimplstypesfoodordererror>, inkprimitiveslangerror>
{ Ok: { Ok: [ { foodId: 1 restaurantId: 1 customerId: 1 courierId: 0 deliveryAddress: 123 main st status: FoodPrepared timestamp: 1,691,776,638,103 price: 1,000 eta: 200 } { foodId: 1 restaurantId: 1 customerId: 2 courierId: 0 deliveryAddress: 345 captain lane, CA status: OrderSubmitted timestamp: 1,691,777,304,084 price: 1,000 eta: 0 } ] } }
```

8/12/2023 3:16:39 am X

Read

Once the restaurant confirms an order, it generally calls a courier to let him deliver the food. Let's check the delivery status too.

call from account
ACCOUNT 1 (SUBWALLET-JS)

transferrable 15.3530 SBY
ZWexd44p3r... ▾

message to send
getService::getDeliveryAll (from: GetDeliveryAllInput1, to: GetDeliveryAllInput2):
Result<Result<Vec<LogicImplsTypesDelivery>, LogicImplsTypesFoodOrderError>, InkPrimitivesLangError>

from: GetDeliveryAllInput1
0

to: GetDeliveryAllInput2
10

max reftime allowed (m)
1

max read gas

max proofsize allowed
1000000

400s execution time, 9.99% of block weight

all results ▾

```
getservice::getdeliveryall (from: 0, to: 10): result<result<vec<logicimplstypesdelivery>, logicimplstypesfoodordererror>, inkprimitiveslangerror>
{ Ok: { Ok: [ { orderId: 1 restaurantId: 1 customerId: 1 courierId: 0 deliveryAddress: 123 main st status: Waiting timestamp: 1,691,777,574,052 } ] } }
```

8/12/2023 3:17:23 am X

X Read

8. Add a courier

The courier can be registered by calling `add_courier` message.

call from account
ACCOUNT 3 (SUBWALLET-JS)

transferrable 8.6964 SBY
ZvALqKqv5H... ▾

message to send
managerService::addCourier (courierAccount: AddCourierInput1, courierName: AddCourierInput2, courierAddress: AddCourierInput3, phoneNumber: AddCourierInput4): Result<Result<u64, LogicImplsTypesFoodOrderError>, InkPrimitivesLangError> ⚒

courierAccount: AddCourierInput1
ACCOUNT 3 (SUBWALLET-JS)

ZvALqKqv5H... ▾

courierName: AddCourierInput2
courier1

courierAddress: AddCourierInput3
234 main st

phoneNumber: AddCourierInput4
9876542123

max reftime allowed (m)
32488

max proofsize allowed
3407872

0.400s execution time, 9.99% of block weight

read contract only, no execution

→ Execute

contract to use
FOODORDER

ZkedUUC3...

call from account
ACCOUNT 1 (SUBWALLET-JS)

transferrable 9.9958 sby
ZPJU7ALjd... ▾

message to send
getService::getCourierAll (from: GetCourierAllInput1, to: GetCourierAllInput2):
Result<Result<Vec<LogicImplsTypesCourier>, LogicImplsTypesFoodOrderError, InkPrimitivesLangError>

from: GetCourierAllInput1
0

to: GetCourierAllInput2
10

max reftime allowed (m)
1

max read gas

max proofsize allowed
1000000

400s execution time, 9.99% of block weight

all results ▲

getservice::getcourierall (from: 0, to: 10): result<result<vec<logicimplstypescourier>, 8/11/2023 11:20:32 am
logicimplstypesfoodordererror, inkprimitiveslangerror>
{ Ok. [{courierAccount: ZvALQHQV5Hvigh4Z6oPN5f8uMmkY3TZvY3jqBgZsUuC courierName: courier1 courierAddress: 234 main st phoneNumber: 9876542123 }] }

  Read

9. Pickup a delivery

Once a courier arrives at the restaurant, then he picks up a delivery.
This is implemented in the `pickupDelivery` message.

contract to use
FOODORDER

call from account
ACCOUNT 3 (SUBWALLET-JS) transferrable 8,6940 SBY
ZvALqKqv5H...

message to send
courierService::pickupDelivery (deliveryId: PickupDeliveryInput1): Result<Result<u64, LogicImplsTypesFoodOrderError>, InkPrimitivesLangError>

deliveryId: PickupDeliveryInput1
1

max reftime allowed (m)
32488

max proofsize allowed
3407872

0.400s execution time, 9.99% of block weight

read contract only, no execution



Execute

call from account
ACCOUNT 3 (SUBWALLET-JS) transferrable 8,6923 SBY
ZvALqKqv5H...

message to send
getService::getDeliveryAll (from: GetDeliveryAllInput1, to: GetDeliveryAllInput2): Result<Result<Vec<LogicImplsTypesDelivery>, LogicImplsTypesFoodOrderError>, InkPrimitivesLangError>

from: GetDeliveryAllInput1
0

to: GetDeliveryAllInput2
10

max reftime allowed (m)
1

max proofsize allowed
1000000

0.400s execution time, 9.99% of block weight

Call results ▾

```
getservice::getdeliveryall (from: 0, to: 10): result<result<vec<logicimplstypesdelivery>, logicimplstypesfoodordererror>, inkprimitiveslangerror>
{ Ok: { Ok: { { orderId: 1 restaurantId: 1 customerId: 1 courierId: 1 deliveryAddress: 123 main st status: PickedUp time
stamp: 1,691,777,574,052 } } } }
```

8/12/2023 3:23:15 am



Read

10. Deliver an order

It means that the restaurant finally gives the food to a courier and the courier starts to carry it to its destination.

call a contract

contract to use
FOODORDER

call from account
ACCOUNT 1 (SUBWALLET-JS)

message to send
`restaurantService::deliverOrder (orderId: DeliverOrderInput1): Result<Result<u64, LogicImplsTypesFoodOrderError>, InkPrimitivesLangError>`

orderId: DeliverOrderInput1
1

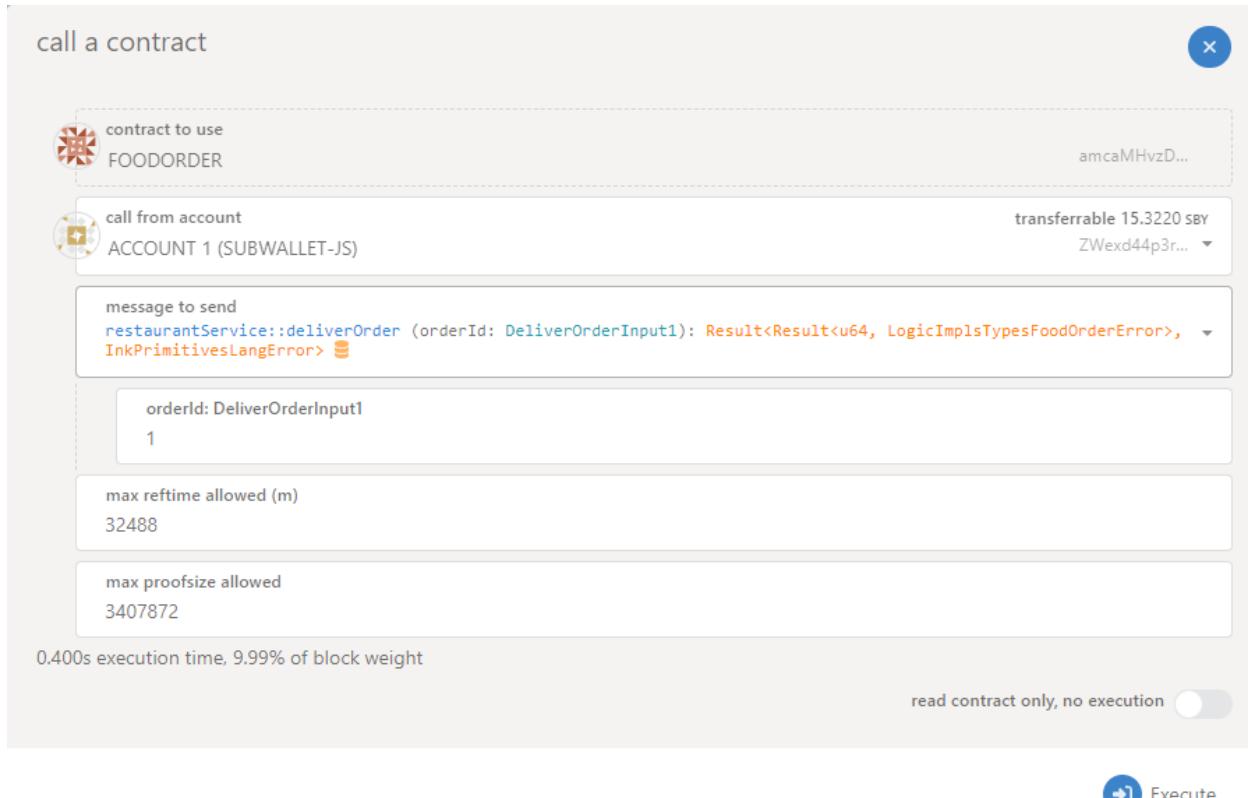
max reftime allowed (m)
32488

max proofsize allowed
3407872

0.400s execution time, 9.99% of block weight

read contract only, no execution

 Execute



call a contract

contract to use
FOODORDER

call from account
ACCOUNT 1 (SUBWALLET-JS) transferrable 15.3220 sby
ZWexd44p3r... ▾

message to send
getService::getOrderAll (from: GetOrderAllInput1, to: GetOrderAllInput2):
Result<Result<Vec<LogicImplsTypesOrder>, LogicImplsTypesFoodOrderError>, InkPrimitivesLangError>

from: GetOrderAllInput1
0

to: GetOrderAllInput2
10

max reftime allowed (m)
1

max proofsize allowed
1000000

max read gas

0.400s execution time, 9.99% of block weight

Call results ▲

getservice::getorderall (from: 0, to: 10): result<result<vec<logicimplstypesorder>, logicimplstypesfoodordererror>, inkprimitiveslangerror>
{Ok: {foodId: 1 restaurantId: 1 customerId: 1 courierId: 1 deliveryAddress: asfasdfsadf status: DeliveryAccepted timestamp: 1,691,783,334,608 price: 1,000 eta: 100 } } }
8/19/2023 4:32:06 am

✖

➡ Read

11. Accept a delivery.

Once the delivery is carried to the customer, then the customer will check the order and accept it.

contract to use
FOODORDER

call from account
ACCOUNT 1 (SUBWALLET-JS)

transferrable 9.9958 sby
ZPJU7ALjd... ▾

message to send
`customerService::acceptDelivery (deliveryId: AcceptDeliveryInput1): Result<Result<u64, LogicImplsTypesFoodOrderError>, InkPrimitivesLangError>` ▾

deliveryId: AcceptDeliveryInput1
1

max reftime allowed (m)
32488

max proofsize allowed
3407872

0.400s execution time, 9.99% of block weight

read contract only, no execution

 Execute

✓ contracts.call inblock

system.ExtrinsicSuccess
 balances.Withdraw
 balances.Transfer (x2)
 contracts.ContractEmitted
 contracts.Called
 balances.Deposit (x2)
 transactionPayment.TransactionFeePaid
 extrinsic event

call a contract

contract to use
FOODORDER

call from account
ACCOUNT 1 (SUBWALLET-JS)

message to send
getService::getOrderAll (from: GetOrderAllInput1, to: GetOrderAllInput2):
Result<Result<Vec<LogicImplsTypesOrder>, LogicImplsTypesFoodOrderError>, InkPrimitivesLangError>

from: GetOrderAllInput1
0

to: GetOrderAllInput2
10

max reftime allowed (m)
1

max proofsize allowed
1000000

max read gas

0.400s execution time, 9.99% of block weight

Call results 

```
getservice::getorderall (from: 0, to: 10): result<result<vec<logicimplstypesorder>, logicimplstypesfoodordererror>, inkprimitiveslangerror>
{ Ok: { Ok: [{ foodId: 1 restaurantId: 1 customerId: 1 courierId: 1 deliveryAddress: asfasdfsadf status: FoodDelivered timestamp: 1,691,783,334,608 price: 1,000 eta: 100 } ] } }
```

  Read