

# BlockchainFoodOrder Smart Contract Deployment and Interactions Guide

[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.

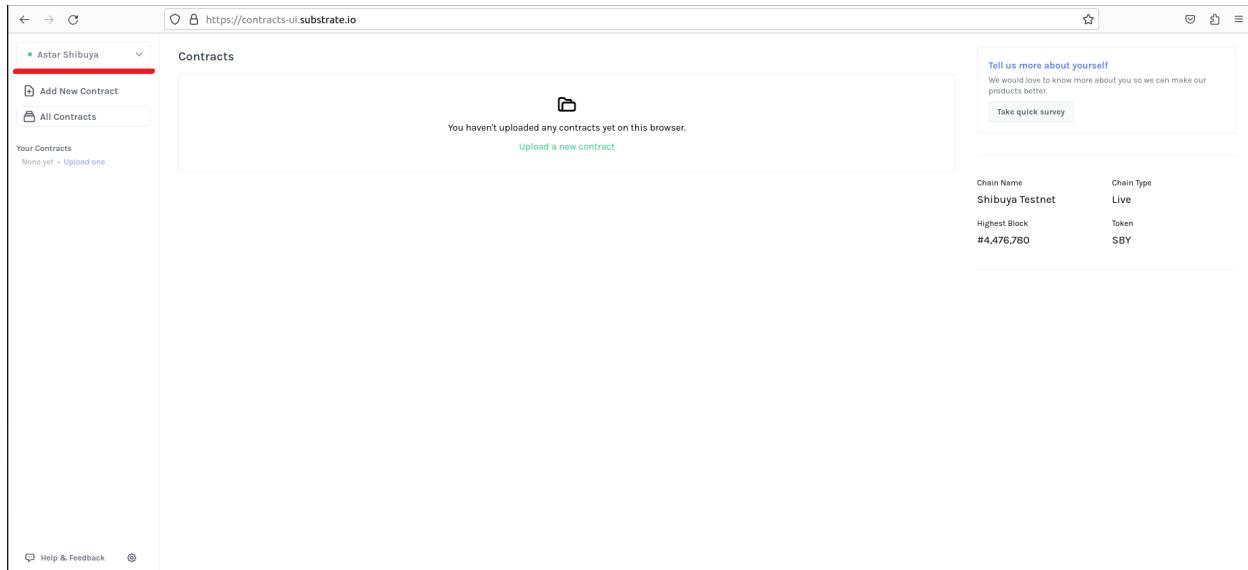
[Contracts-UI](#) is a web application for deploying Wasm smart contracts on Substrate chains. [Polkadot.js](#) is an effort to provide a collection of tools, utilities and libraries for interacting with the Polkadot network from JavaScript.

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. It is generally recommended to use the more user friendly Contracts UI over the PolkadotJS app for ink! deployments and interactions.

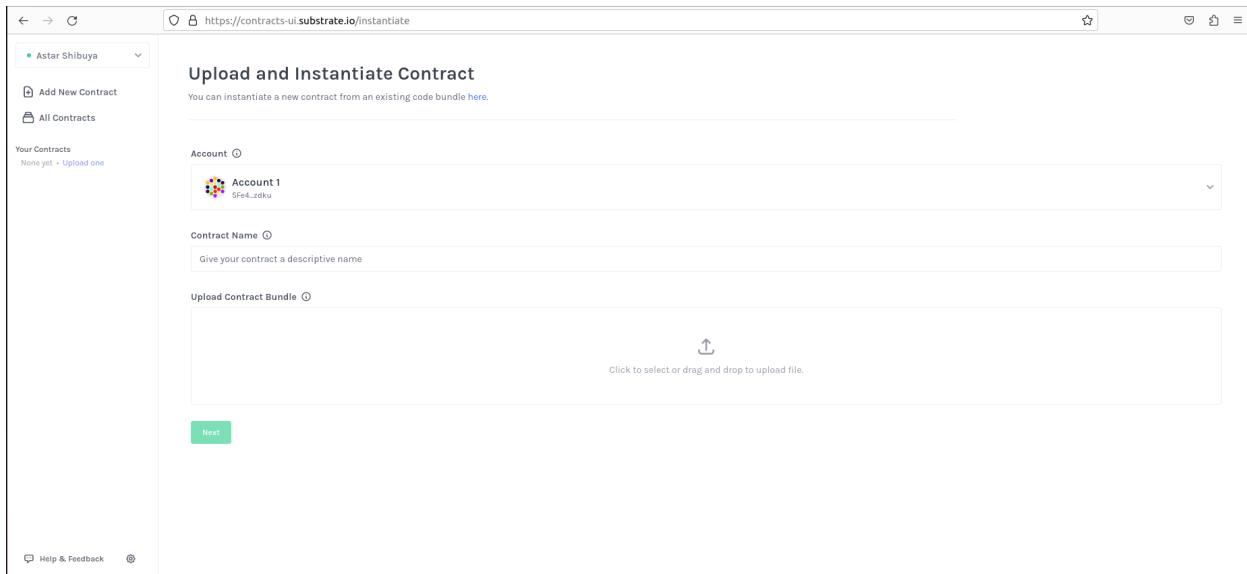
## Working with Contracts-UI

### Upload and instantiate

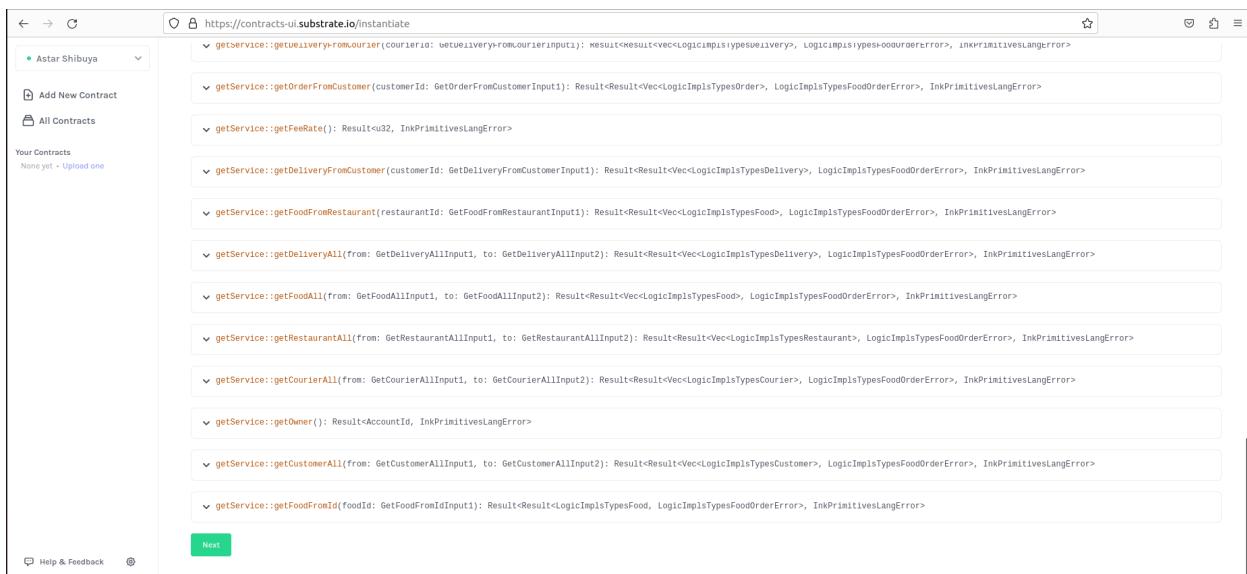
1. Change the network to Astar Shibuya.



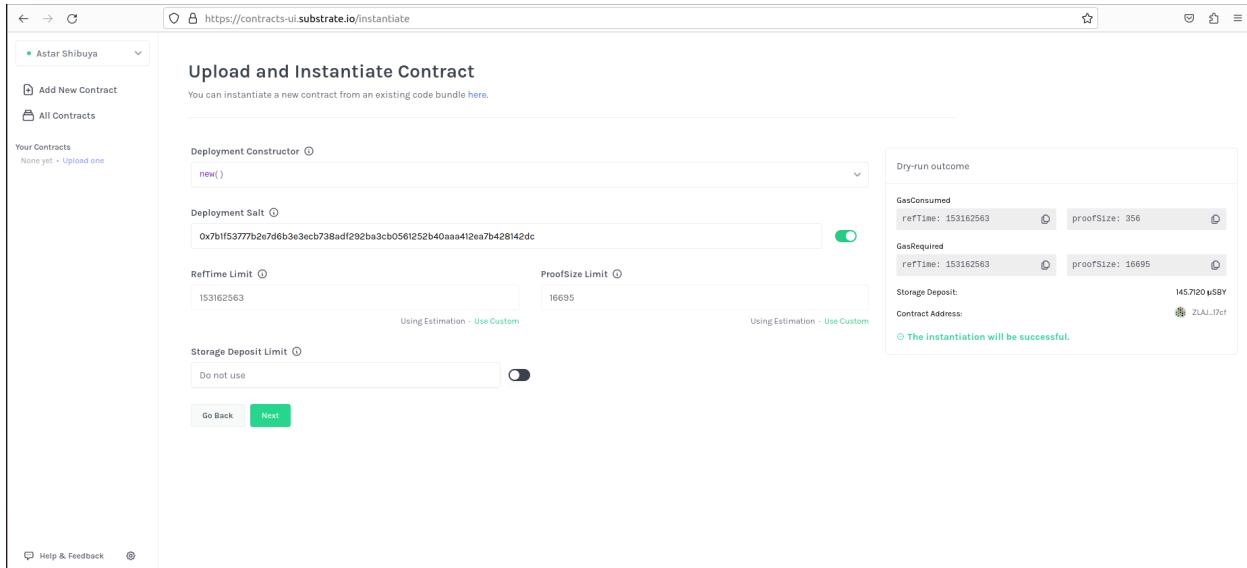
2. Click the “Upload a new contract” button.



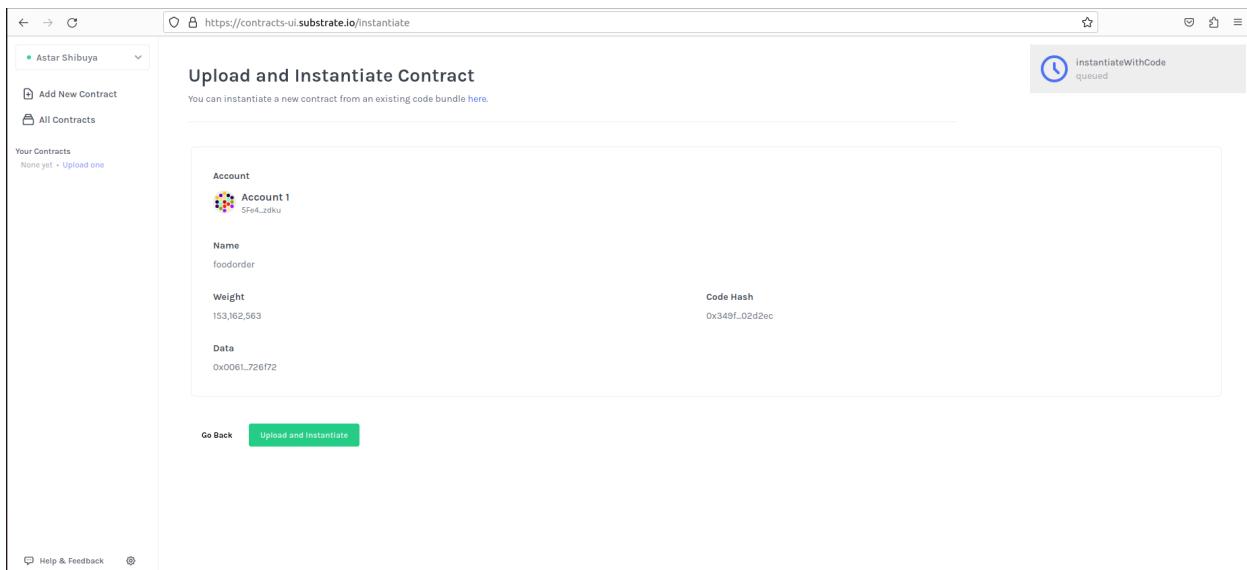
3. Drag and drop the “foodorder.contract” file and click the “Next” button.



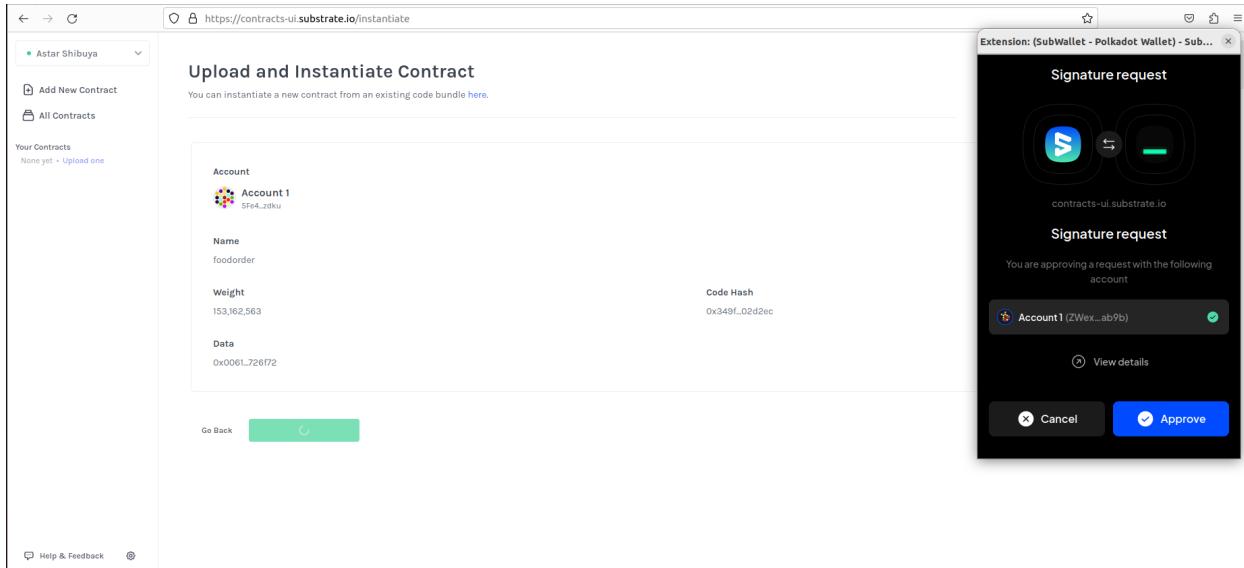
4. Click the “Next” button after checking the messages.



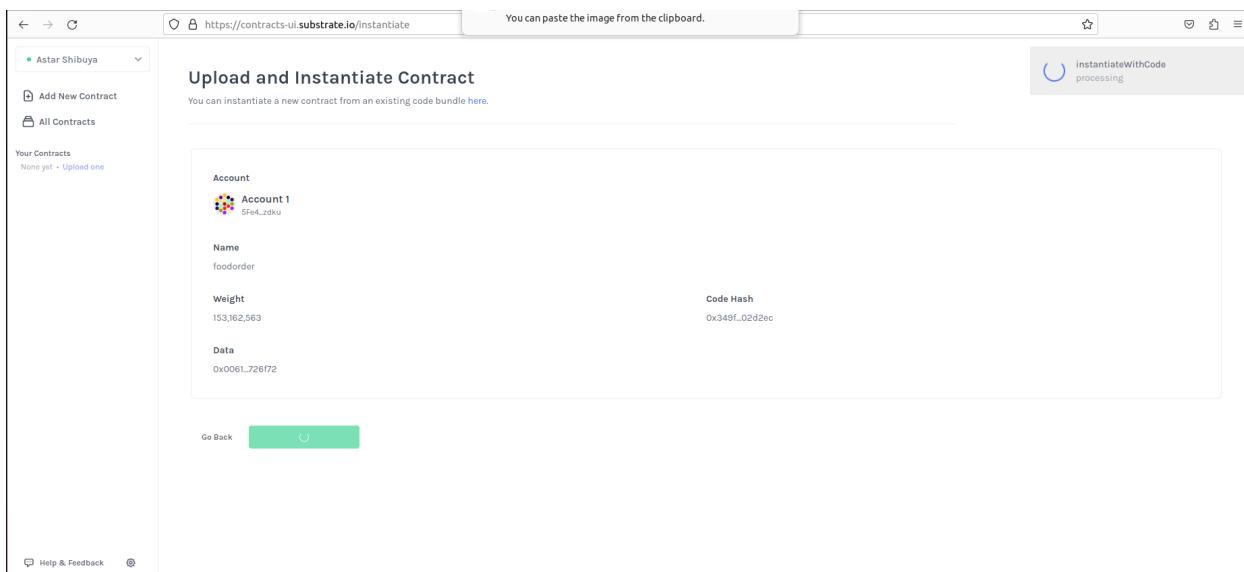
5. Click the “Next” button after checking the constructor message instantiates the contract.



6. Click the “Upload and Instantiate” button to finalize the uploading and instantiating the contract.



7. You can see the pop-up window and click the “Approve” button.

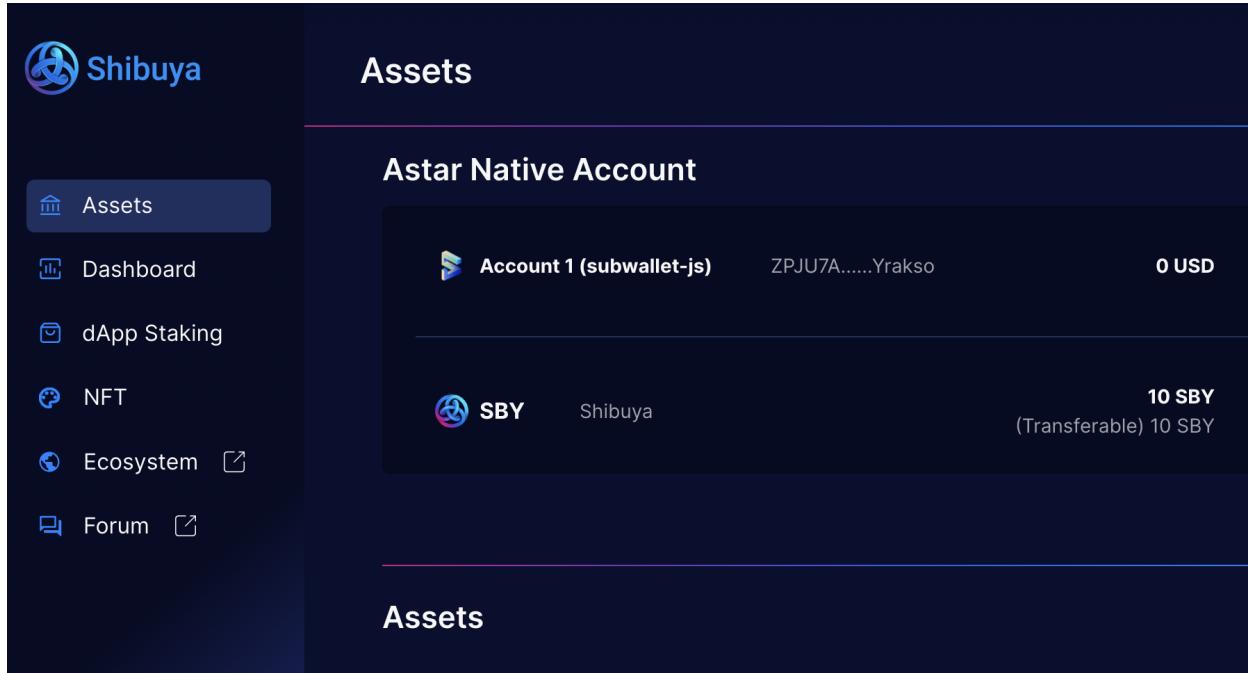


8. Now, the BlockchainFoodORder smart contract has successfully been deployed on Shibuya via Contracts-UI at address: YezJmtfEtFowEBto53VCpkLGHvFXtpe88frPW1q1z7Y2jUd, you can interact with the contract's messages.

# Interact with the smart contract

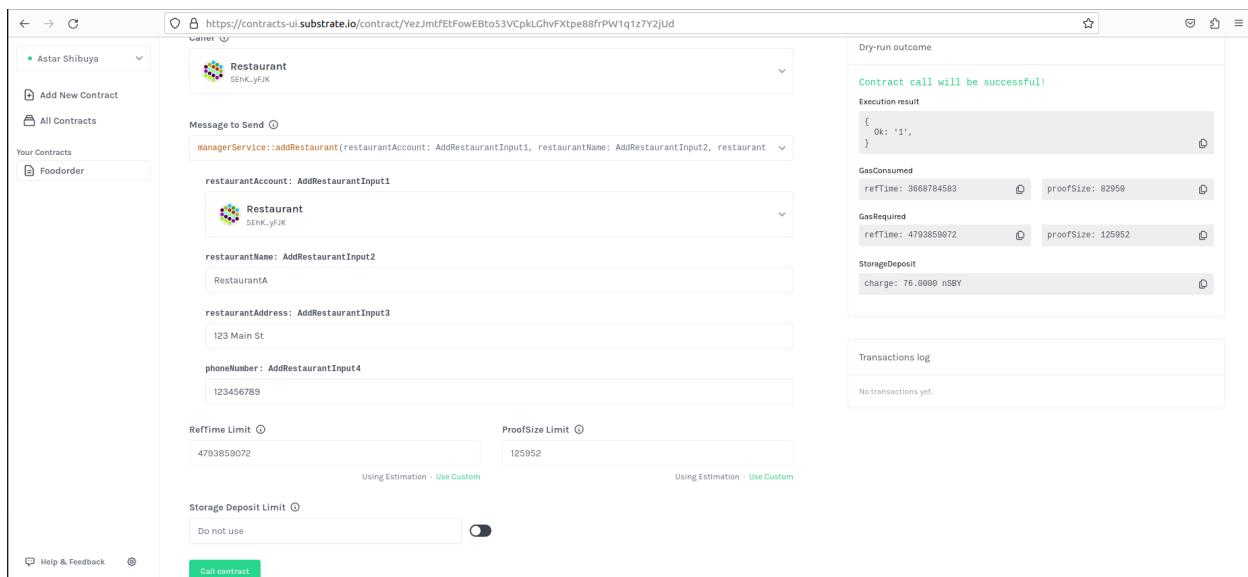
1. Get free SBY tokens from a faucet.

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



The screenshot shows the Shibuya web interface. On the left, a sidebar menu includes 'Assets' (which is selected and highlighted in blue), 'Dashboard', 'dApp Staking', 'NFT', 'Ecosystem', and 'Forum'. The main content area is titled 'Assets' and shows an 'Astar Native Account'. It displays an account named 'Account 1 (subwallet-js)' with the address 'ZPJU7A.....Yrakso' and a balance of '0 USD'. Below this, it shows an asset entry for 'SBY' with the amount '10 SBY' and the note '(Transferable) 10 SBY'. The bottom section is also titled 'Assets'.

2. Restaurant registers itself into the smart contract storage.



The screenshot shows the contracts-ui substrate interface. On the left, a sidebar menu includes 'Astar Shibuya' (selected), 'Add New Contract', 'All Contracts', and 'Foodorder'. The main content area shows a 'Restaurant' contract with a placeholder address '6ENL...yfJK'. Below it, there are input fields for 'Message to Send' with the code 'managerService::addRestaurant(restaurantAccount: AddRestaurantInput1, restaurantName: AddRestaurantInput2, restaurantAddress: AddRestaurantInput3, phoneNumber: AddRestaurantInput4)'. The inputs are filled with 'AddRestaurantInput1' (placeholder), 'RestaurantA' (placeholder), '123 Main St' (placeholder), and '123456789' (placeholder). On the right, the 'Dry-run outcome' section shows a successful contract call with the result '{ OK: '1' }'. It also displays 'GasConsumed' (refTime: 3668784583, proofSize: 82950), 'GasRequired' (refTime: 4793859072, proofSize: 125952), and 'StorageDeposit' (charge: 76.0000 nSBY). The 'Transactions log' section is empty, stating 'No transactions yet.'

Message to Send

```
managerService::addRestaurant(restaurantAccount: AddRestaurantInput1, restaurantName: AddRestaurantInput2, restaurantAddress: AddRestaurantInput3, phoneNumber: AddRestaurantInput4)
```

restaurantAccount: AddRestaurantInput1

restaurantName: AddRestaurantInput2

restaurantAddress: AddRestaurantInput3

phoneNumber: AddRestaurantInput4

RefTime Limit: 4793859072

ProofSize Limit: 125952

Storage Deposit Limit: Do not use

Dry-run outcome

Contract call will be successful

Execution result

```
{
  "Ok": "1"
}
```

GasConsumed: 3668784583

GasRequired: 4793859072

StorageDeposit: charge: 76.0000 nSBY

Transactions log: No transactions yet.

Signature request

You are approving a request with the following account

Restaurant (YzU...ZSD)

View details

Cancel Approve

Message to Send

```
managerService::addRestaurant(restaurantAccount: AddRestaurantInput1, restaurantName: AddRestaurantInput2, restaurantAddress: AddRestaurantInput3, phoneNumber: AddRestaurantInput4)
```

restaurantAccount: AddRestaurantInput1

restaurantName: AddRestaurantInput2

restaurantAddress: AddRestaurantInput3

phoneNumber: AddRestaurantInput4

RefTime Limit: 4793859072

ProofSize Limit: 125952

Storage Deposit Limit: Do not use

Dry-run outcome

Contract Reverted!

call success

Execution result

RestaurantAlreadyExist

GasConsumed: 3102209561

GasRequired: 4793859072

StorageDeposit: charge: 0

Transactions log

22/8/2023, 12:32:55 pm

ManagerService::add\_restaurant()

CONTRACT EVENTS

AddRestaurantEvent

restaurantId: "1"

restaurantName: "RestaurantA"

restaurantAddress: "123 Main St"

phoneNumber: "123456789"

GENERIC EVENTS

balances::Withdraw

### 3. The Courier registers itself into the smart contract.

https://contracts-ui.substrate.io/contract/YezJmtfElFowEBto53VCpkLChvFXtpe88frPW1q1z7Y2jUd

**Dry-run outcome**  
Contract call will be successful!

**Execution result**  
{  
Ok: '1',  
}  
GasConsumed  
refTime: 3674238680 proofSize: 82950  
GasRequired  
refTime: 4793859072 proofSize: 125952  
StorageDeposit  
charge: 77.0000 nSBY

**Transactions log**  
22/8/2023, 12:32:55 pm  
ManagerService::add\_restaurant()

**CONTRACT EVENTS**  
AddRestaurantEvent  
restaurantId: "1"  
restaurantName: "RestaurantA"  
restaurantAddress: "123 Main St"  
phoneNumber: "123456789"

https://contracts-ui.substrate.io/contract/YezJmtfElFowEBto53VCpkLChvFXtpe88frPW1q1z7Y2jUd

**Dry-run outcome**  
Contract call will be successful!

**Execution result**  
{  
Ok: '1',  
}  
GasConsumed  
refTime: 3674238680 proofSize: 82950  
GasRequired  
refTime: 4793859072 proofSize: 125952  
StorageDeposit  
charge: 77.0000 nSBY

**Transactions log**  
22/8/2023, 12:32:55 pm  
ManagerService::add\_restaurant()

**CONTRACT EVENTS**  
AddRestaurantEvent  
restaurantId: "1"  
restaurantName: "RestaurantA"  
restaurantAddress: "123 Main St"  
phoneNumber: "123456789"

**Extension: (SubWallet - Polkadot Wallet) - Sub...**  
**Signature request**  
You are approving a request with the following account  
Courier (WKPT\_3NEY)   
View details

#### 4. A Customer registers itself into the smart contract.

https://contracts-ui.substrate.io/contract/YezJmtfElFowEBto53VCpkLGhvFxtpe88frPW1q1z7Y2jUd

Caller: Customer (5eo2\_pSgP)

Message to Send

```
customerService::addCustomer(customerName: AddCustomerInput1, customerAddress: AddCustomerInput2, phoneNumber: AddCustomerInput3)
```

customerName: AddCustomerInput1  
CustomerA

customerAddress: AddCustomerInput2  
456 WestPoint St

phoneNumber: AddCustomerInput3  
987654321

ReffTime Limit: 4793859072

ProofSize Limit: 125952

Storage Deposit Limit: Do not use

Call contract

Dry-run outcome

Contract call will be successful!

Execution result

```
{ "Ok: '1'" }
```

GasConsumed

refTime: 3491983360 proofSize: 89149

GasRequired

refTime: 4793859072 proofSize: 125952

StorageDeposit

charge: 79.0000 nSBY

Transactions log

22/8/2023, 12:34:56 pm

ManagerService::add\_courier()

CONTRACT EVENTS

AddCourierEvent

courierId: "1"  
courierName: "CourierA"  
courierAddress: "234 Downtown St"  
phoneNumber: "654987321"

https://contracts-ui.substrate.io/contract/YezJmtfElFowEBto53VCpkLGhvFxtpe88frPW1q1z7Y2jUd

Caller: Customer (5eo2\_pSgP)

Message to Send

```
customerService::addCustomer(customerName: AddCustomerInput1, customerAddress: AddCustomerInput2, phoneNumber: AddCustomerInput3)
```

customerName: AddCustomerInput1  
CustomerA

customerAddress: AddCustomerInput2  
456 WestPoint St

phoneNumber: AddCustomerInput3  
987654321

ReffTime Limit: 4793859072

ProofSize Limit: 125952

Storage Deposit Limit: Do not use

Call contract

Dry-run outcome

Contract call will be successful!

Execution result

```
{ "Ok: '1'" }
```

GasConsumed

refTime: 3491983360 proofSize: 89149

GasRequired

refTime: 4793859072 proofSize: 125952

StorageDeposit

charge: 79.0000 nSBY

Transactions log

22/8/2023, 12:34:56 pm

ManagerService::add\_courier()

CONTRACT EVENTS

AddCourierEvent

courierId: "1"  
courierName: "CourierA"  
courierAddress: "234 Downtown St"  
phoneNumber: "654987321"

Extension: (SubWallet - Polkadot Wallet) - Sub...

Signature request

contracts-ui.substrate.io

Signature request

You are approving a request with the following account

Customer (Yfcm...QZgN)

View details

Cancel Approve

## 5. The Restaurant adds one food.

The screenshot shows the Substrate Contracts UI interface. On the left, the sidebar shows 'Astar Shibuya' selected, with options to 'Add New Contract', 'All Contracts', and 'Foodorder'. The main area is titled 'Caller' and shows a 'Restaurant' icon. Below it, a 'Message to Send' section contains the following message:

```
restaurantService::addFood(foodName: AddFoodInput1, description: AddFoodInput2, price: AddFoodInput3, eta: AddFoodInput4)
```

The message fields are filled with the following values:

- foodName: AddFoodInput1 → FoodA
- description: AddFoodInput2 → FoodA
- price: AddFoodInput3 → 50
- eta: AddFoodInput4 → 100

Below the message, there are sections for 'ReffTime Limit' (4793859072) and 'ProofSize Limit' (125952). A 'Storage Deposit Limit' section has a toggle switch set to 'Do not use'. At the bottom is a green 'Call contract' button.

The right side of the screen displays the 'Dry-run outcome' and 'Execution result' sections. The execution result shows a successful outcome with the value 'Ok: "1"'. Below this are sections for 'GasConsumed' (refTime: 3712073156, proofSize: 83578), 'GasRequired' (refTime: 4793859072, proofSize: 125952), and 'StorageDeposit' (charge: 63.0000 nSBY).

At the bottom right, a 'Transactions log' section shows the timestamp 22/8/2023, 12:38:55 pm and the event 'CustomerService::add\_customer()'.

## 6. Check the “getFoodFromId” message’s result.

The screenshot shows the Substrate Contracts UI interface. The sidebar is the same as the previous screenshot. The main area is titled 'Caller' and shows a 'Customer' icon. Below it, a 'Message to Send' section contains the following message:

```
getService::getFoodFromId(foodId: GetFoodFromIdInput1): Result<Result<LogicImplsTypesFood, LogicImplsTypesFoodOrder>>
```

The message field 'foodId' is filled with the value '1'.

The right side of the screen displays the 'Outcome' and 'Return value' sections. The return value is a JSON object:

```
{
  "Ok": {
    "Food": {
      "name": "FoodA",
      "description": "1",
      "price": "50",
      "eta": "100",
      "timestamp": "1,692,679,230,628"
    }
  }
}
```

Below this is a 'Transactions log' section showing the timestamp 22/8/2023, 12:42:43 pm and the event 'CustomerService::submit\_order()'.

## 7. The restaurant adds one more food.

Caller: Restaurant

Message to Send:

```
restaurantService::addFood(foodName: AddFoodInput1, description: AddFoodInput2, price: AddFoodInput3, eta: AddFoodInput4, storageDepositLimit: AddFoodInput5)
```

FoodName: AddFoodInput1

FoodB

description: AddFoodInput2

FoodB

price: AddFoodInput3

100

eta: AddFoodInput4

200

Storage Deposit Limit: Do not use

Reftime Limit: 4793859072

ProofSize Limit: 125952

CONTRACT EVENTS

SubmitOrderEvent

SubmitOrderEvent

foodId

restaurantId

customerId

timestamp

Signature request

You are approving a request with the following account

Restaurant (YzU...ZSDE)

Approve

## 8. Check the “getFoodAll” message’s result.

Caller: Customer

Message to Send:

```
getService::getFoodAll(from: GetFoodAllInput1, to: GetFoodAllInput2): Result<Result<Vec<LogicImplsTypesFood>, LogicImplsTypesError>
```

from: GetFoodAllInput1

1

to: GetFoodAllInput2

10

Outcome

Return value

```
{
  "ok": [
    {
      "FoodName": "FoodA",
      "restaurantId": "1",
      "description": "FoodA",
      "price": "50",
      "eta": "100",
      "timestamp": "1,692,679,230,620",
    },
    {
      "FoodName": "FoodB",
      "restaurantId": "1",
      "description": "FoodB",
      "price": "100",
      "eta": "200",
      "timestamp": "1,692,679,470,628",
    }
  ],
  "error": null
}
```

Transactions log

22/8/2023, 12:44:44 pm

RestaurantService::add\_food()

CONTRACT EVENTS

AddFoodEvent

FoodId

FoodName

FoodB

RestaurantId

Timestamp

## 9. The customer submits an order.

The screenshot shows the Substrate Contracts UI for the 'Foodorder' contract. On the left, the 'Foodorder' contract is selected. The main area shows a 'Message to Send' for the `customerService::submitOrder` method. The message includes parameters: `foodId: SubmitOrderInput1` (value: 1), `deliveryAddress: SubmitOrderInput2` (value: 324 Main St), and `refTimeLimit: 4793859072`. The 'ProofSize Limit' is set to 125952. The 'Storage Deposit Limit' is set to 0.000000000000000005. A 'Call contract' button is present. On the right, the 'Dry-run outcome' section shows a successful execution result with an empty object. It also displays gas consumption (4051068123), gas required (4793859072), and storage deposit (charge: 98.0000 nSBY). The 'Transactions log' shows the event `RestaurantService::add_food()` with details: foodId: "1", foodName: "FoodA", restaurantId: "1", description: "FoodA".

## 10. Check the order status.

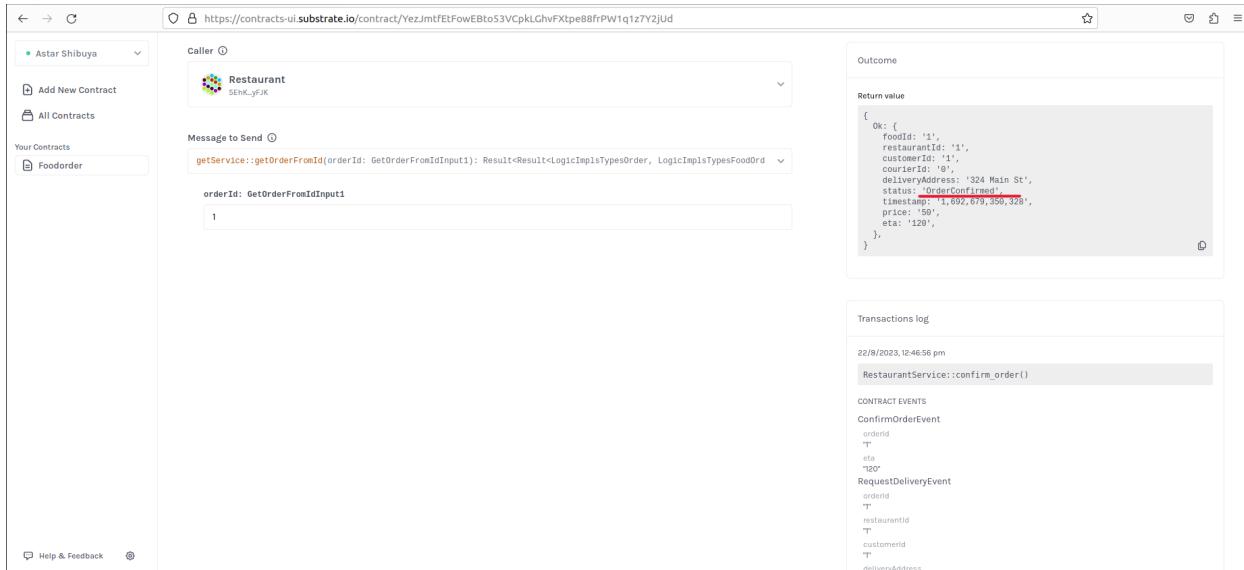
The screenshot shows the Substrate Contracts UI for the 'Foodorder' contract. The 'Foodorder' contract is selected on the left. The main area shows a 'Message to Send' for the `getService::getOrderFromId` method. The message includes a parameter: `orderId: GetOrderFromIdInput1` (value: 1). On the right, the 'Outcome' section shows the return value, which is an object with an 'Ok' key. The 'Ok' value is an object containing: foodId: '1', restaurantId: '1', customerId: '1', courierId: '1', deliveryAddress: '324 Main St', status: 'OrderSubmitted', timestamp: '1,692,679,398,328', price: '50', eta: '9'. The 'Transactions log' shows the event `RestaurantService::add_food()` with details: foodId: "2", foodName: "FoodB", restaurantId: "1", description: "FoodB", price: "100.0000 aSBY", eta: "200".

The screenshot shows the Astar Shibuya substrate UI. On the left, the sidebar shows 'Astar Shibuya' selected, with options to 'Add New Contract', 'All Contracts', and 'Your Contracts' (Foodorder). The main area has a 'Caller' section showing 'Customer' (5eo2...psg8). Below it is a 'Message to Send' section for the 'getOrderAll' method. The 'from' field is 'GetOrderAllInput1' with value '1'. The 'to' field is 'GetOrderAllInput2' with value '10'. To the right, there are two panels: 'Outcome' showing the JSON return value (an array of order objects) and 'Transactions log' showing the timestamp '22/8/2023, 12:44:44 pm' and the event 'RestaurantService::add\_food()'.

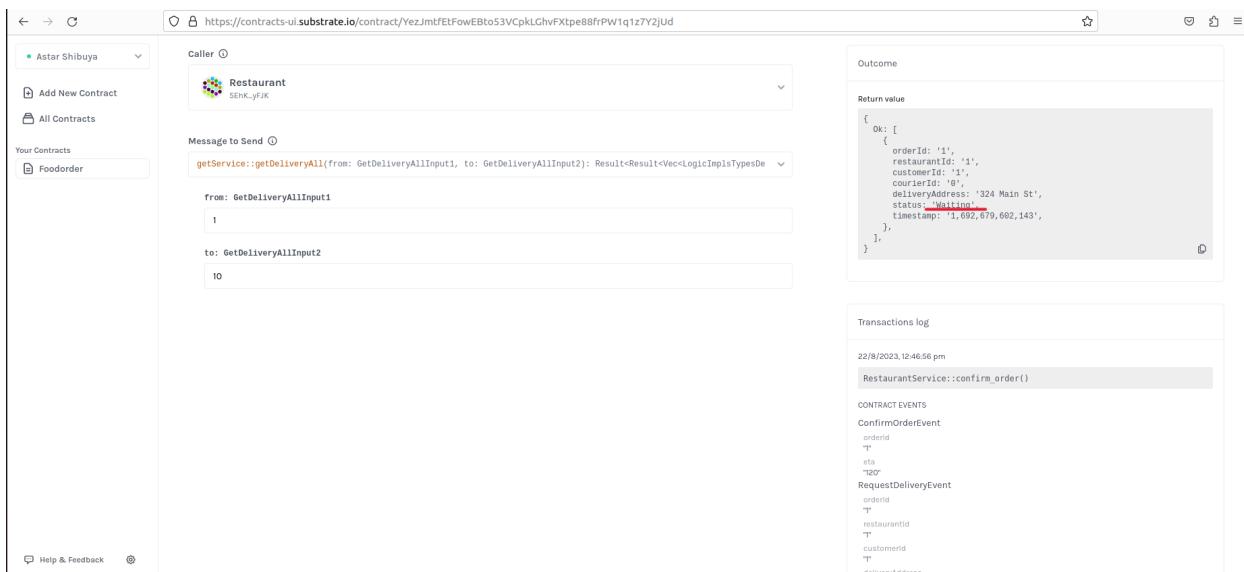
11. The restaurant confirms an order before starting the cook and calls a courier to deliver the order after eta deadline.

The screenshot shows the Astar Shibuya substrate UI. On the left, the sidebar shows 'Astar Shibuya' selected, with options to 'Add New Contract', 'All Contracts', and 'Your Contracts' (Foodorder). The main area has a 'Caller' section showing 'Restaurant' (6HK...yJk). Below it is a 'Message to Send' section for the 'confirmOrder' method. The 'orderId' field is 'ConfirmOrderInput1' with value '1'. The 'eta' field is 'ConfirmOrderInput2' with value '120'. Below the message fields are 'RefTime Limit' (4793859072) and 'ProofSize Limit' (125552). At the bottom is a 'Call contract' button. To the right, there are two panels: 'Dry-run outcome' showing 'Contract call will be successful!', 'Execution result' showing 'OK: '1'', 'GasConsumed' (refTime: 4376092637, proofSize: 89267), 'GasRequired' (refTime: 4793859072, proofSize: 125952), and 'StorageDeposit' (charge: 74.0000 nSBY). The 'Transactions log' panel shows the timestamp '22/8/2023, 12:44:44 pm' and the event 'RestaurantService::add\_food()'.

12. Check the order status and delivery status.



The screenshot shows the Substrate UI interface for a blockchain-based food delivery system. The left sidebar shows the user is connected to the 'Astar Shibuya' network and has a 'Foodorder' contract selected. The main area shows a transaction being sent from a 'Restaurant' account (SEhK\_yfJK) to the 'Foodorder' contract. The message being sent is the result of a 'getOrderFromId' call, with the order ID set to '1'. The 'Outcome' panel on the right shows the JSON response of the call, which includes fields like orderId, restaurantId, customerId, courierId, deliveryAddress, status, timestamp, price, and eta. The 'Transactions log' panel at the bottom shows the transaction details, including the event 'RestaurantService::confirm\_order()' and the event 'RequestDeliveryEvent' with the same parameters.



This screenshot shows a similar transaction process, but for a delivery confirmation. The message being sent is the result of a 'getDeliveryAll' call, with the 'from' field set to 'GetDeliveryAllInput1' and the 'to' field set to 'GetDeliveryAllInput2'. The 'Outcome' panel shows the JSON response, which includes a list of order details. The 'Transactions log' panel shows the transaction details, including the event 'RestaurantService::confirm\_order()' and the event 'RequestDeliveryEvent' with the same parameters.

13. The restaurant finishes the cooking of the submitted order.

Message to Send

```
restaurantService::finishCook(orderId: FinishCookInput1): Result<Result<u64, LogicImplsTypesFoodOrderError>, InkPrim1
```

orderId: FinishCookInput1

1

Reffime Limit

4793859072

ProofSize Limit

125952

Storage Deposit Limit

Do not use

Call contract

Dry-run outcome

Contract call will be successful!

```
{
  Ok: '1',
  }
}
```

GasConsumed

refTime: 3986268386 proofSize: 88178

GasRequired

refTime: 4793859072 proofSize: 125952

StorageDeposit

charge: 0

Transactions log

22/8/2023, 12:46:56 pm

RestaurantService::confirm\_order()

CONTRACT EVENTS

ConfirmOrderEvent

orderID: "T"  
eta: "120"

RequestDeliveryEvent

orderID: "T"  
restaurantID: "1"

#### 14. Check the order status.

Message to Send

```
getService::getOrderFromId(orderId: GetOrderFromIdInput1): Result<Result<LogicImplsTypesOrder, LogicImplsTypesFoodOrderError>, InkPrim1
```

orderId: GetOrderFromIdInput1

1

Outcome

Return value

```
{
  Ok: {
    orderId: '1',
    restaurantId: '1',
    customerId: '1',
    courierId: '9',
    deliveryAddress: '324 Main St',
    status: 'Cooked',
    timestamp: '1,692,079,350,328',
    price: '50',
    eta: '120',
  },
}
```

Transactions log

22/8/2023, 12:49:19 pm

RestaurantService::finish\_cook()

CONTRACT EVENTS

FinishCookEvent

orderId: "T"

GENERAL EVENTS

balances::Withdraw  
balances::Transfer  
balances::Called  
balances::Deposit  
balances::Deposit  
balances::Deposit  
balances::Deposit  
balances::Deposit

#### 15. The courier picks up the delivery after food is prepared.

The screenshot shows the Substrate Contracts UI interface. On the left, the sidebar shows 'Astar Shibuya' selected, with options to 'Add New Contract', 'All Contracts', and 'Your Contracts' (Foodorder). The main area shows a 'Caller' section with 'Courier' selected. Below it is a 'Message to Send' section with the message: `courierService::pickupDelivery(deliveryId: PickupDeliveryInput1): Result<Result<u64, LogicImplsTypesFoodOrderError>,`. The message input field contains `deliveryId: PickupDeliveryInput1`. Below this are 'ReffTime Limit' (4793859072), 'ProofSize Limit' (125952), and 'Storage Deposit Limit' (disabled). A green 'Call contract' button is at the bottom. On the right, the 'Dry-run outcome' section shows a green message: 'Contract call will be successful!'. It details the execution result as an object with an 'Ok' key containing the value '1'. It also shows gas consumption: refTime: 4836525131, proofSize: 85761, and storage deposit: charge: 10.0000 nSBY. The 'Transactions log' section shows a log entry for `RestaurantService::finish_cook()` at 22/8/2023, 12:49:19 pm. The log also includes generic events for balances withdraw, balances transfer, contracts called, and balances deposit.

## 16. Check the delivery status.

The screenshot shows the Substrate Contracts UI interface. The sidebar is the same as the previous screenshot. The main area shows a 'Caller' section with 'Restaurant' selected. Below it is a 'Message to Send' section with the message: `getService::getDeliveryFromId(deliveryId: GetDeliveryFromIdInput1): Result<Result<LogicImplsTypesDelivery, LogicImpls`. The message input field contains `deliveryId: GetDeliveryFromIdInput1`. A green 'Call contract' button is at the bottom. On the right, the 'Outcome' section shows a green message: 'Return value'. It details the result as an object with an 'Ok' key containing an object with fields: orderId: '1', restaurantId: '1', customerId: '1', courierId: '1', deliveryAddress: '324 Main St', status: 'PickedUp', and timestamp: '1,692,679,602,143'. The 'Transactions log' section shows a log entry for `CourierService::pickup_delivery()` at 22/8/2023, 12:51:31 pm. The log also includes generic events for balances withdraw, contracts called, balances transfer, balances deposit, and balances deposit.

## 17. The restaurant change the order status to “FoodDelivered”.

Caller: Restaurant

Message to Send: `restaurantService::deliverOrder(orderId: DeliverOrderInput1): Result<Result<u64, LogicImplsTypesFoodOrderError>, InkP`

orderId: `DeliverOrderInput1`

RefTime Limit: 4793859072

ProofSize Limit: 125952

Storage Deposit Limit: Do not use

Dry-run outcome: `call processing`  
Contract call will be successful!

Execution result:

```
{
  Ok: '1',
}

```

GasConsumed: refTime: 3848595929, proofSize: 86937

GasRequired: refTime: 4793859072, proofSize: 125952

StorageDeposit: charge: 0

Transactions log:

22/8/2023, 12:51:31 pm  
CourierService::pickup\_delivery()

CONTRACT EVENTS  
PickUpDeliveryEvent  
deliveryId: "T"  
courierId: "T"

GENERIC EVENTS  
balances::Withdraw  
contracts::Called

## 18. Check the order status.

Caller: Restaurant

Message to Send: `getService::getOrderFromId(orderId: GetOrderFromIdInput1): Result<Result<LogicImplsTypesOrder, LogicImplsTypesFoodOrderError>, InkP`

orderId: `GetOrderFromIdInput1`

Return value:

```
{
  Ok: {
    foodId: '1',
    restaurantId: '1',
    customerId: '1',
    courierId: '1',
    deliveryAddress: '324 Main St',
    status: 'FoodDelivered',
    timestamp: '1,692,679,356,328',
    price: '50$',
    eta: '120',
  }
}
```

Transactions log:

22/8/2023, 12:53:08 pm  
RestaurantService::deliver\_order()

CONTRACT EVENTS  
DeliverFoodEvent  
orderID: "T"  
restaurantId: "T"  
customerId: "T"  
courierId: "T"

GENERIC EVENTS  
balances::Withdraw  
contracts::Called

## 19. The customer accepts the delivery.

## 20. Check the order status.

## 21. Check the delivery status.

# Capturing smart contract upgrades

## 1. Instantiate FoodOrder\_V1 contract.

## 2. Create a courier in FoodOrder\_V1 contract.

Local Node

Add New Contract

All Contracts

Your Contracts

- Foodorder\_V1
- Foodorder\_V2

Foodorder\_V1

You instantiated this contract 5CFB...LpTk from Foodorder on 13 Sep

Metadata Interact

Caller

Message to Send

courierServiceImpl::createCourier(courierName: CreateCourierInput1, courierAddress: CreateCourierInput2, phoneNumber: CreateCourierInput3)

courierName: CreateCourierInput1

Courier-Alice

courierAddress: CreateCourierInput2

101 Main St

phoneNumber: CreateCourierInput3

1234567895

Reftime Limit

3947587584

ProofSize Limit

629760

Using Estimation - Use Custom

Storage Deposit Limit

Do not use

Call contract

call success

balances:Withdraw contracts:Called balances:Transfer transactionPayment:TransactionFeePaid system:ExtrinsicSuccess

Dry-run outcome

Contract Reverted!

Execution result

AlreadyExist

GasConsumed

refTime: 2862523870 proofSize: 434136

GasRequired

refTime: 3947587584 proofSize: 629760

StorageDeposit

charge: 0

Transactions log

13/9/2023, 11:34:07 am

CourierServiceImpl::create\_courier()

GENERIC EVENTS

balances:Withdraw contracts:Called balances:Transfer transactionPayment:TransactionFeePaid system:ExtrinsicSuccess

### 3. Create a customer in FoodOrder\_V1 contract.

Local Node

Add New Contract

All Contracts

Your Contracts

- Foodorder\_V1
- Foodorder\_V2

Foodorder\_V1

You instantiated this contract 5CFB...LpTk from Foodorder on 13 Sep

Metadata Interact

Caller

Message to Send

customerServiceImpl::createCustomer(customerName: CreateCustomerInput1, customerAddress: CreateCustomerInput2, phoneNumber: CreateCustomerInput3)

customerName: CreateCustomerInput1

Customer-Bob

customerAddress: CreateCustomerInput2

342 Harkward St

phoneNumber: CreateCustomerInput3

654215489

Reftime Limit

3947587584

ProofSize Limit

629760

Using Estimation - Use Custom

Storage Deposit Limit

Do not use

Call contract

call success

balances:Withdraw contracts:Called balances:Transfer transactionPayment:TransactionFeePaid system:ExtrinsicSuccess

Dry-run outcome

Contract call will be successful!

Execution result

{ Ok: '1', }

GasConsumed

refTime: 3268196383 proofSize: 434056

GasRequired

refTime: 3947587584 proofSize: 629760

StorageDeposit

charge: 2.0111 Unit

Transactions log

13/9/2023, 11:34:45 am

CustomerServiceImpl::create\_customer()

GENERIC EVENTS

balances:Withdraw contracts:Called balances:Transfer transactionPayment:TransactionFeePaid system:ExtrinsicSuccess

### 4. Create a restaurant in FoodOrder\_V1 contract.

The screenshot shows the Foodorder\_V1 contract interface. The left sidebar lists 'Local Node' and 'Your Contracts' with 'Foodorder\_V1' selected. The main area shows a 'Message to Send' for the `restaurantServiceImpl::createRestaurant` method. The parameters are set as follows:

- `restaurantName: CreateRestaurantInput1`: Restaurant-Charlie
- `restaurantAddress: CreateRestaurantInput2`: 150 Wall St
- `phoneNumber: CreateRestaurantInput3`: 987654215

Below the message, 'RefTime Limit' is set to 3947587584 and 'ProofSize Limit' is set to 629760. A 'Storage Deposit Limit' slider is set to 'Do not use'. A 'Call contract' button is at the bottom. The right side shows a 'call SUCCESS' result with a green checkmark. The 'Dry-run outcome' section shows 'Contract Reverted!' with an error message 'Execution result: AlreadyExist'. The 'Transactions log' section shows the transaction details for the `RestaurantServiceImpl::create_restaurant()` call.

## 5. Create a food in FoodOrder\_V1 contract.

The screenshot shows the Foodorder\_V1 contract interface. The left sidebar lists 'Local Node' and 'Your Contracts' with 'Foodorder\_V1' selected. The main area shows a 'Message to Send' for the `restaurantServiceImpl::createFood` method. The parameters are set as follows:

- `foodName: CreateFoodInput1`: Food A
- `foodDescription: CreateFoodInput2`: FoodA description
- `foodPrice: CreateFoodInput3`: 100
- `foodEta: CreateFoodInput4`: 20

Below the message, 'RefTime Limit' is set to 3947587584 and 'ProofSize Limit' is set to 629760. A 'Storage Deposit Limit' slider is set to 'Do not use'. A 'Call contract' button is at the bottom. The right side shows a 'call SUCCESS' result with a green checkmark. The 'Dry-run outcome' section shows 'Contract call will be successful!' with an error message 'Execution result: { Ok: '2' }'. The 'Transactions log' section shows the transaction details for the `RestaurantServiceImpl::create_food()` call.

## 6. Submit an order in FoodOrder\_V1 contract.

Foodorder\_V1

You instantiated this contract `5CFB...LpTe` from `Foodorder` on 13 Sep

Caller: bob

Message to Send:

`customerServiceImpl::submitOrder(foodId: SubmitOrderInput1, deliveryAddress: SubmitOrderInput2): Result<Result<u64, LogicalError>`

foodId: SubmitOrderInput1  
1

deliveryAddress: SubmitOrderInput2  
343 Main St

RefTime Limit: 3947587584

ProofSize Limit: 629760

Storage Deposit Limit: Do not use

Value: 0.0000000000000001

Unit

Using Estimation - Use Custom

Using Estimation - Use Custom

Call contract

Help & Feedback

Local Node

Add New Contract

All Contracts

Your Contracts

Foodorder\_V1

Foodorder\_V2

Metadata

Interact

call

SUCCESS

balances:Withdraw

balances:Transfer (x2)

contracts:ContractSubmitted

contracts:Called

transactionPayment:TransactionFeePaid

system:ExtrinsicSuccess

Dry-run outcome

Contract call will be successful!

Execution result

Ok: '1',

GasConsumed

refTime: 3744100138

proofSize: 444647

GasRequired

refTime: 3947587584

proofSize: 629760

StorageDeposit

charge: 1.0093 Unit

Transactions log

13/9/2023, 11:39:48 am

`CustomerServiceImpl::submit_order()`

CONTRACT EVENTS

SubmitOrderEvent

orderid: '1'

customerAccount: 'SFHneW46xGxgs5mUiveU4ab7yG8zmnstUpZC92UhjJM684tY'

## 7. Instantiate FoodOrder\_V2 contract.

Foodorder\_V2

You instantiated this contract `5HGP...JY0m` from `Foodorder` on 13 Sep

Caller: alice

Message to Send:

`courierServiceImpl::readCourierAll(from: ReadCourierAllInput1, to: ReadCourierAllInput2): Result<@Result<Vec<LogicImpls>, LogicalError>`

from: ReadCourierAllInput1  
0

to: ReadCourierAllInput2  
0

Outcome

Return value

InvalidParameters

Transactions log

No transactions yet.

Help & Feedback

Local Node

Add New Contract

All Contracts

Your Contracts

Foodorder\_V1

Foodorder\_V2

Metadata

Interact

Reinstantiate

https://contracts-ui.substrate.io/instantiate/0x752af90e7e2b1a5f19a54fb2f64c37390ff5a3ecb6db99969744c0ab34fb2e

## 8. Copy code\_hash of FoodOrder\_V2 contract.

Local Node

Add New Contract

All Contracts

Your Contracts

- Foodorder\_V1
- Foodorder\_V2

Instantiate Contract from Code Hash

You can upload and instantiate new contract code [here](#).

Account

alice

Contract Name

foodorder

On-Chain Code

foodorder\_V2 Copied to clipboard

Code hash: 0x752a\_4fba2e

Metadata

Contract Hash: 0x204cd920bd133169e6595b250c7c0c1a8ff0da732a3e693aed88c3cf0f3bf4a

Language: ink! 4.3.0

Contract version: 1.0.0

Compiler: rustc 1.72.0

Authors: Alan Boyd <alan.cameron.boyd@gmail.com>

Method List (partial):

- courierServiceImpl::readCourierAll (from: ReadCourierAllInput1, to: ReadCourierAllInput2): Result<Result<Vec<LogicImplsDataCourier>, LogicImplsDataFoodOrderError>, InkPrimitivesLangError>
- courierServiceImpl::readCourierFromId (courierId: ReadCourierFromIdInput1): Result<Result<LogicImplsDataCourier, LogicImplsDataFoodOrderError>, InkPrimitivesLangError>
- courierServiceImpl::readCourier (): Result<Result<LogicImplsDataCourier, LogicImplsDataFoodOrderError>, InkPrimitivesLangError>

Help & Feedback

## 9. Confirm an order before upgrading the contract for comparison.

Local Node

Add New Contract

All Contracts

Your Contracts

- Foodorder\_V1
- Foodorder\_V2

Foodorder\_V1

You instantiated this contract [5CFB...LpTk](#) from [foodorder](#) on 13 Sep

Metadata

Caller

charlie

Message to Send

restaurantServiceImpl::confirmOrder(orderId: ConfirmOrderInput1, eta: ConfirmOrderInput2): Result<Result<u64, LogicImplsDataFoodOrderError>, InkPrimitivesLangError>

orderId: ConfirmOrderInput1

1

eta: ConfirmOrderInput2

50

RefTime Limit

4119537840

ProofSize Limit

629760

Storage Deposit Limit

Do not use

Call contract

call

SUCCESS

balances::Withdraw

contracts::ContractEmitted (x2)

contracts::Called

transactionPayment::TransactionFeePaid

system::ExtrinsicSuccess

Dry-run outcome

Contract call will be successful!

Execution result

```
{
  "Ok": "1",
}
```

GasConsumed

refTime: 4119537840

proofSize: 453676

GasRequired

refTime: 4119537840

proofSize: 629760

StorageDeposit

charge: 0

Transactions log

13/9/2023, 11:43:29 am

RestaurantServiceImpl::confirm\_order()

CONTRACT EVENTS

GENERIC EVENTS

balances::New

contracts::Called

transactionPayment::TransactionFeePaid

system::ExtrinsicSuccess

Help & Feedback

## 10. Set code\_hash to FoodOrder\_V1.

Local Node

Foodorder\_V1

You instantiated this contract `5CFB...LpTk` from `Foodorder` on 13 Sep

Metadata

Caller: alice

Message to Send:

```
upgradeable::setCodeHash(newCodeHash: SetCodeHashInput1): Result<Result<Null, OpenbrushContractsErrorsUpgradeableUpgra>
```

newCodeHash: SetCodeHashInput1

0x 752af90e7e2b1a5f19a54fb2f64c37390ff5a3eccb6db89969744c0b34fb2e

RefTime Limit: 7012235366

ProofSize Limit: 1075377

Storage Deposit Limit: Do not use

Call contract

call

SUCCESS

Dry-run outcome

Contract call will be successful!

Execution result

Ok

GasConsumed

refTime: 5951414718 proofSize: 885157

GasRequired

refTime: 7012235366 proofSize: 1075377

StorageDeposit

charge: 0

Transactions log

13/9/2023, 11:41:11 am

Upgradeable::set\_code\_hash()

GENERIC EVENTS

balances::Withdraw

contracts::ContractCodeUpdated

contracts::Called

transactionPayment::TransactionFeePaid

system::ExtrinsicSuccess

## 11. Update metadata to a new one.

Local Node

Foodorder\_V1

You instantiated this contract `5CFB...LpTk` from `Foodorder` on 13 Sep

Metadata

new()

No documentation provided

courierServiceImpl::pickupDelivery(deliveryId: PickupDeliveryInput1): Result<Result<u64, LogicImplsDataFoodOrderError>, InkPrimitivesLangError>

Function that a courier picks up food at a restaurant for delivery

courierServiceImpl::readCourierFromId(courierId: ReadCourierFromIdInput1): Result<Result<LogicImplsDataCourier, LogicImplsDataFoodOrderError>, InkPrimitivesLangError>

Function that get a courier from given id Use read\_item...from...id procedure macro for Courier

courierServiceImpl::readCourier(): Result<Result<LogicImplsDataCourier, LogicImplsDataFoodOrderError>, InkPrimitivesLangError>

Function to get a courier Use read\_item procedure macro for Courier

courierServiceImpl::createCourier(courierName: CreateCourierInput1, courierAddress: CreateCourierInput2, phoneNumber: CreateCourierInput3): Result<Result<u64, LogicImplsDataFoodOrderError>, InkPrimitivesLangError>

Function to create a courier Use create\_item procedure macro for Courier

courierServiceImpl::readCourierAll(from: ReadCourierAllInput1, to: ReadCourierAllInput2): Result<Result<Vec<LogicImplsDataCourier>, LogicImplsDataFoodOrderError>, InkPrimitivesLangError>

Function to readCourierAll Use readCourierAllInput1, to: ReadCourierAllInput2 procedure macro for Courier

Reinstantiate

Function to create a food

```
^ restaurantServiceImpl::confirmOrder(orderId: ConfirmOrderInput1, eta: ConfirmOrderInput2): Result<Result<u64, LogicImplDataFoodOrderError>, InkPrimitivesLangError> ⓘ
```

Function that a restaurant confirms an order

```
^ ownable::transferOwnership(newOwner: TransferOwnershipInput1): Result<Result<Null, OpenbrushContractsErrorsOwnableOwnableError>, InkPrimitivesLangError> ⓘ
```

No documentation provided

```
^ ownable::owner(): Result<Option<AccountId>, InkPrimitivesLangError>
```

No documentation provided

```
^ ownable::renounceOwnership(): Result<Result<Null, OpenbrushContractsErrorsOwnableOwnableError>, InkPrimitivesLangError> ⓘ
```

No documentation provided

```
^ upgradeable::setCodeHash(newCodeHash: SetCodeHashInput1): Result<Result<Null, OpenbrushContractsErrorsUpgradeableUpgradeableError>, InkPrimitivesLangError> ⓘ
```

No documentation provided

Update Metadata ⓘ

foodorder.json (108.09kb) X

Valid metadata file!

Update metadata

Help & Feedback ⓘ

## 12. Confirm an order after upgrade.

Caller ⓘ

charlie  
5FLS...SSSY

Message to Send ⓘ

```
restaurantServiceImpl::confirmOrder(orderId: ConfirmOrderInput1, eta: ConfirmOrderInput2): Result<Result<u64, LogicImplDataFoodOrderError>, InkPrimitivesLangError> ⓘ
```

orderId: ConfirmOrderInput1  
1

eta: ConfirmOrderInput2  
30

Reftime Limit ⓘ

4119537840

ProofSize Limit ⓘ

620760

Storage Deposit Limit ⓘ

Do not use

Call contract

Dry-run outcome

Contract call will be successful!

Execution result

```
{
  Ok: '1',
}
```

GasConsumed

refTime: 3935199728 proofSize: 453676

GasRequired

refTime: 4119537840 proofSize: 629760

StorageDeposit

charge: 0

Transactions log

13/9/2023, 11:45:01 am

RestaurantServiceImpl::confirm\_order()

CONTRACT EVENTS

RandomCreatedEvent  
rand  
"75"  
DeclineOrderEvent  
orderid  
"1"

GENERIC EVENTS

balances::Withdraw  
contracts::Called  
transactionPayment::TransactionFeePaid  
system::ExtrinsicSuccess

Help & Feedback ⓘ

## 13. Read food to check the old data.

The screenshot shows the Foodorder\_V1 interface. The left sidebar lists 'Local Node', 'Add New Contract', 'All Contracts', and 'Your Contracts' (Foodorder\_V1, Foodorder\_V2). The main area is titled 'Foodorder\_V1' and shows a message sent to 'charlie' with the message text: 'restaurantServiceImpl::readFood(foodId: ReadFoodInput1); Result<Result<LogicImplsDataFood, LogicImplsDataFoodOrderError>'. The 'Caller' section shows 'charlie' (SFLS..559Y). The 'Outcome' section shows a JSON return value: 
 

```
{
      "Ok": {
        "FoodId": "1",
        "FoodName": "Food A",
        "RestaurantId": "1",
        "FoodDescription": "FoodA description",
        "FoodPrice": "100",
        "FoodEta": "20"
      }
    }
```

 The 'Transactions log' section shows two entries: 'ManagerServiceImpl::get\_order()' at 13/9/2023, 11:45:33 am and 'RestaurantServiceImpl::confirm\_order()' at 13/9/2023, 11:45:01 am. The 'Contract Events' section shows 'RandomCreatedEvent'.

## 14. Create a food (this is a new data).

The screenshot shows the Foodorder\_V1 interface. The left sidebar lists 'Local Node', 'Add New Contract', 'All Contracts', and 'Your Contracts' (Foodorder\_V1, Foodorder\_V2). The main area is titled 'Foodorder\_V1' and shows a message sent to 'charlie' with the message text: 'restaurantServiceImpl::createFood(foodName: CreateFoodInput1, foodDescription: CreateFoodInput2, foodPrice: CreateFoodInput3, foodEta: CreateFoodInput4)'. The 'Caller' section shows 'charlie' (SFLS..559Y). The 'Dry-run outcome' section shows a JSON execution result: 
 

```
{
      "Ok": "3"
    }
```

 The 'Transactions log' section shows two entries: 'RestaurantServiceImpl::create\_food()' at 13/9/2023, 11:53:43 am and 'ManagerServiceImpl::get\_order()' at 13/9/2023, 11:45:33 am. The 'Contract Events' section shows 'RandomCreatedEvent'.

## 15. Check both old data and new data by reading all foods.

Local Node

Add New Contract

All Contracts

Your Contracts

Foodorder\_V1

Foodorder\_V2

Metadata

Interact

Caller

charlie

Message to Send

restaurantServiceImpl::readFoodAll(from: ReadFoodAllInput1, to: ReadFoodAllInput2): Result<Result<Vec<LogicImplsDataFc

from: ReadFoodAllInput1

0

to: ReadFoodAllInput2

10

Outcome

Return value

```
{
  Ok: [
    {
      foodId: '1',
      foodName: 'Food A',
      restaurantId: '1',
      foodDescription: 'FoodA description',
      foodPrice: '100',
      foodEta: '20',
    },
    {
      foodId: '2',
      foodName: 'Food B',
      restaurantId: '1',
      foodDescription: 'FoodB description',
      foodPrice: '200',
      foodEta: '30',
    }
  ],
}
```

Transactions log

13/9/2023, 11:53:43 am

RestaurantServiceImpl::create\_food()

13/9/2023, 11:45:33 am

ManagerServiceImpl::get\_order()

Help & Feedback

# Working with Polkadot.js UI

## 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/>.

## Interacting with the smart contract

Click the button “Add an existing contract”

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>

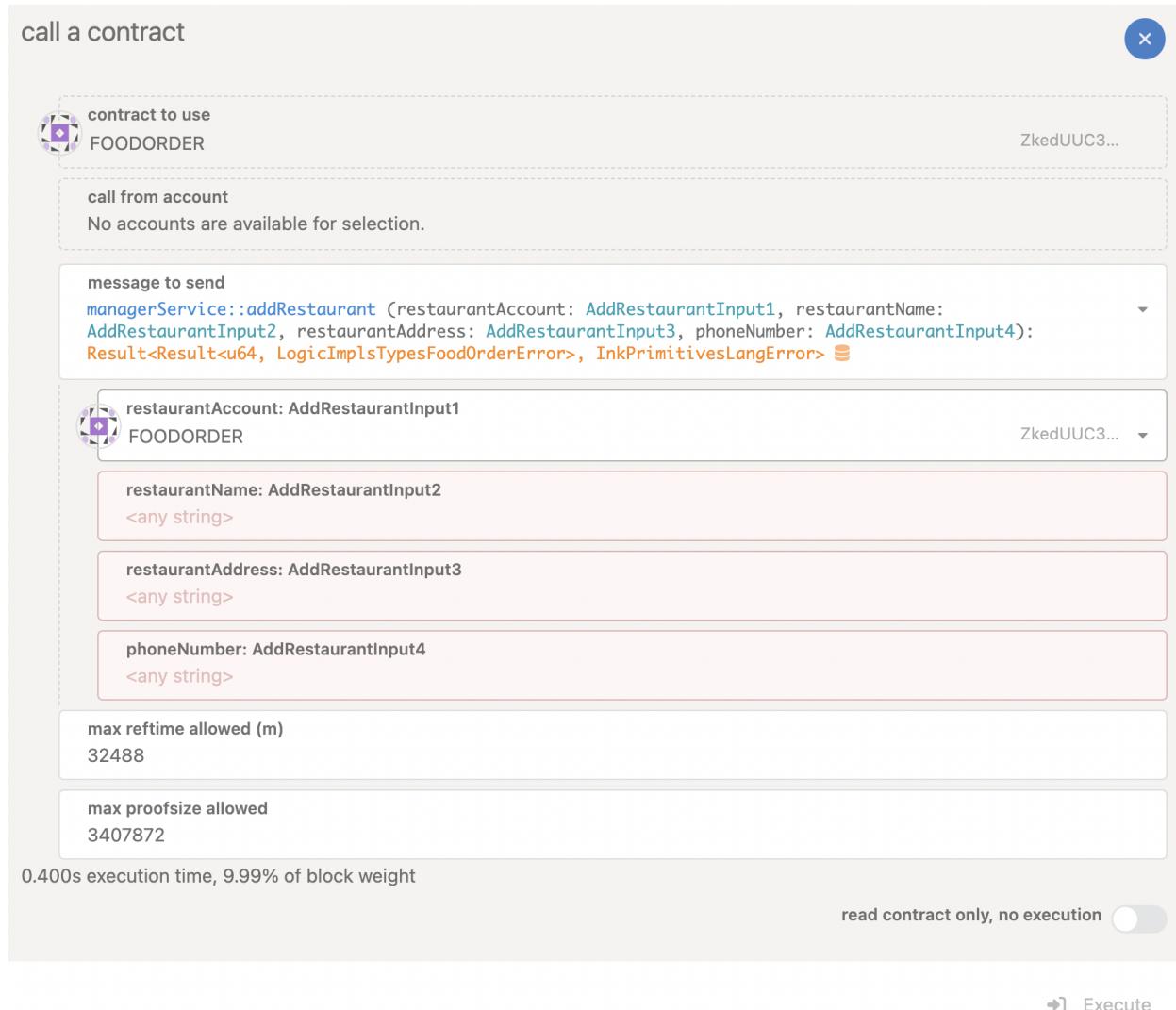
max reftime allowed (m)  
32488

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 '0 USD'. Below that is 'SBY Shibuya' with '10 SBY (Transferable) 10 SBY'. At the bottom, another 'Assets' section is visible.

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

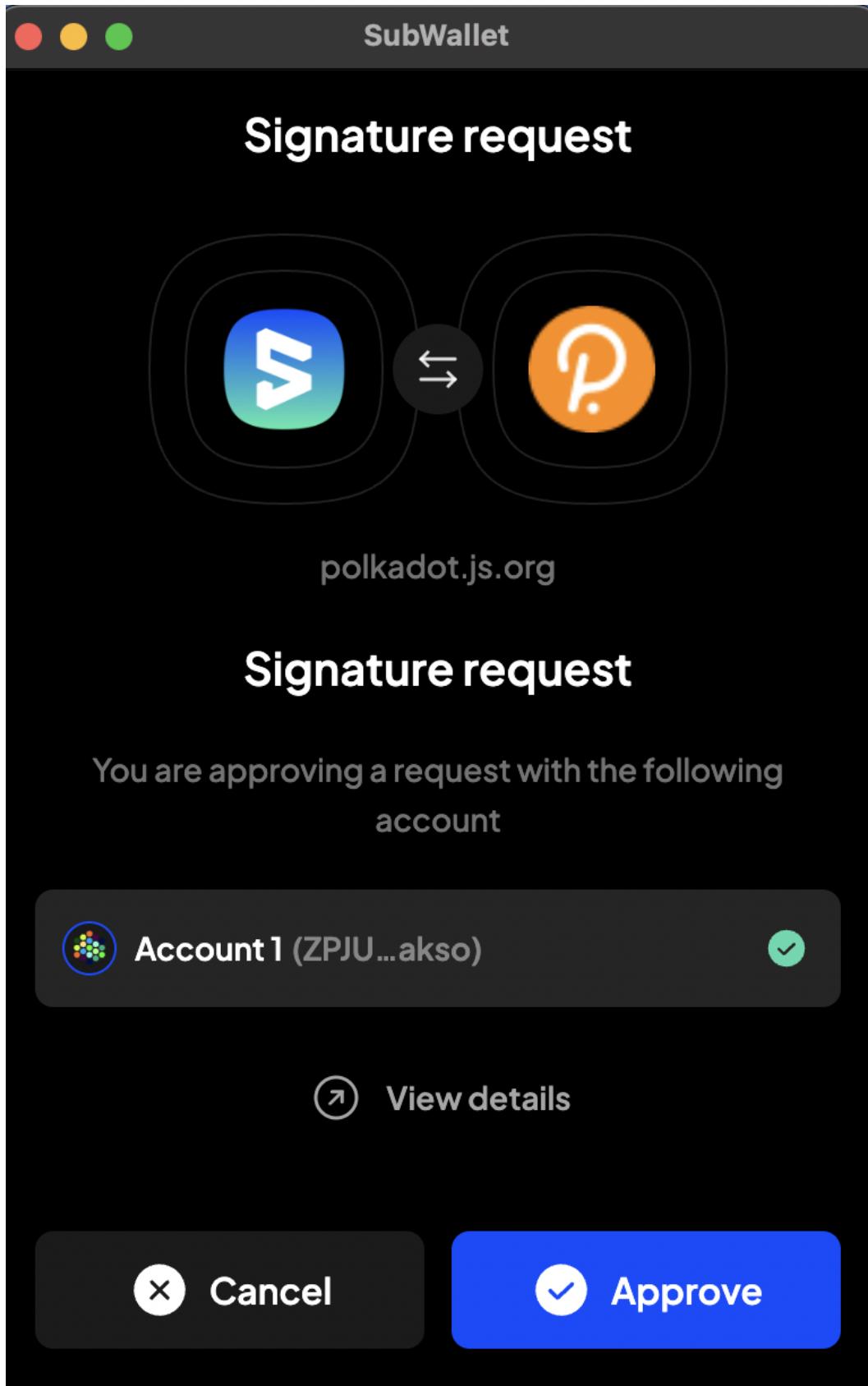
The sending account that will be used to send this transaction. Any applicable fees will be paid by this account.

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>, 8/11/2023 11:00:28 am
{ok {customerAccount: Yc1n8PZ2ERypdRUMWcAjiJ2Qw6Xqm8vbypBt3D14vaQZgN customerName:
customer1 customerAddress: 4534 main st phoneNumber: 654321987 } { customerAccount: ZPJU7ALjdVXfyp
wTAVde77m8tcUTLyXv2PJ7Nqtk1Yrakso customerName: Customer2 customerAddress: 789 Broadway Way ph
oneNumber: 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

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

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.

sending from my account  
ACCOUNT 1 (SUBWALLET-JS) ZPJU7ALjd...

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

do not include a tip for the block author

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

call hash  
0x4579f1b008eeab3b126fadccb97c3487c3eba4b94678be9d871148e5738faef 

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

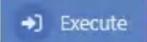
eta: ConfirmOrderInput2  
200

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> ⚒

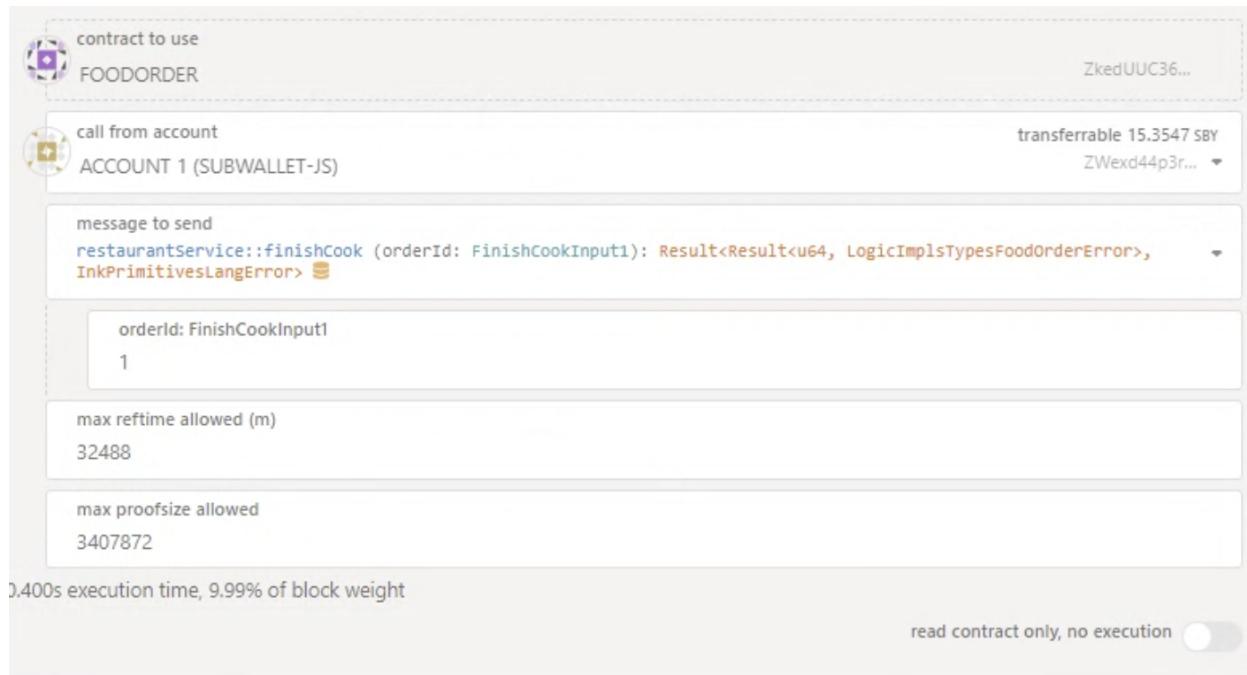
orderId: FinishCookInput1  
1

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.

 Execute

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 ✖

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

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>` X

 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

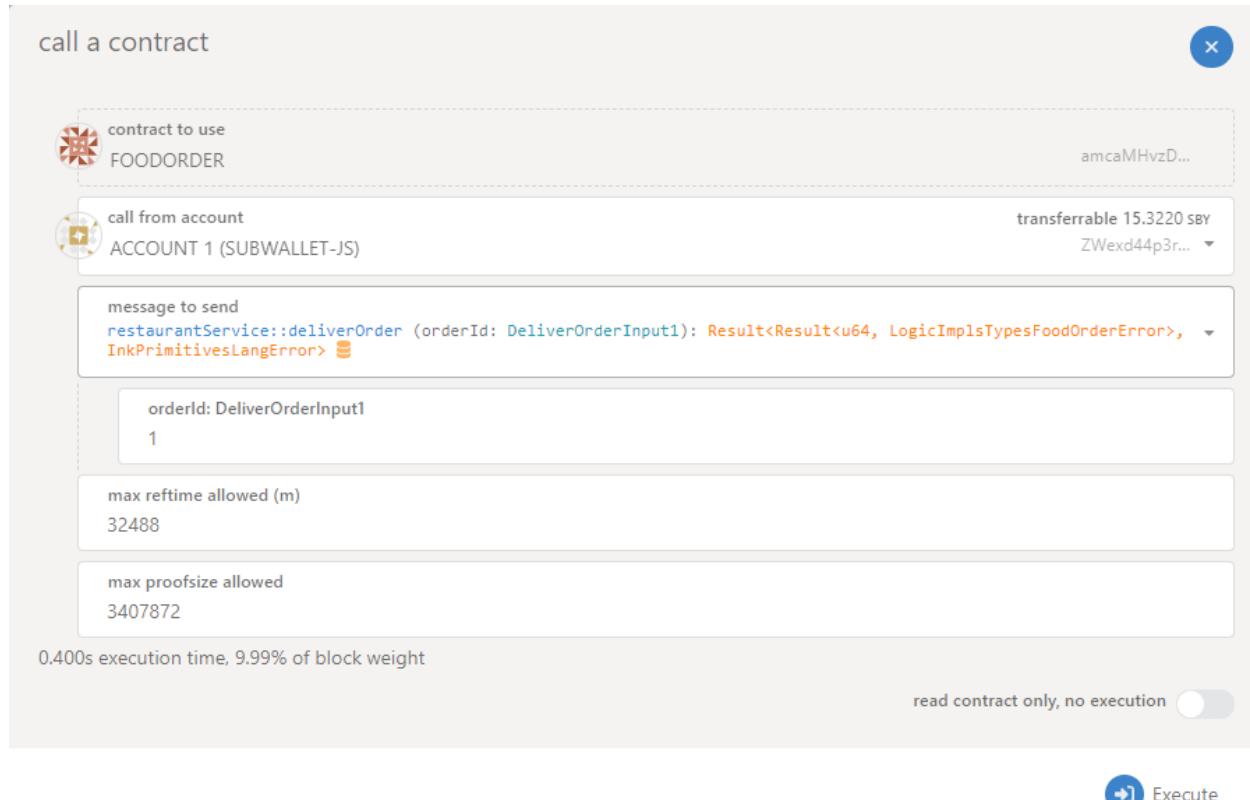
max reftime allowed (m)  
32488

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