Complex Cross Contract Call

This example presents 3 instances of complex cross-contract calls. Particularly, it shows:

1. How to batch multiple function calls to a same contract.
2. How to call multiple contracts in parallel, each returning a different type.
3. Different ways of handling the responses in the callback.

Simple Cross-Contract Calls Check the tutorial on how to usesimple cross-contract calls

Obtaining the Cross Contract Call Example

You have two options to start the Donation Example:

1. You can use the app throughGithub Codespaces
2. , which will open a web-based interactive environment.
3. Clone the repository locally and use it from your computer.

Codespaces Clone locally https://github.com/near-examples/cross-contract-calls

Structure of the Example

The smart contract is available in two flavors: Rust and JavaScript

- JavaScript
- Rust

┌── sandbox-ts # sandbox testing │ ├── external-contracts │ │ ├── counter.wasm │ │ ├── guest-book.wasm │ │ └── hello-near.wasm │ └── main.ava.ts ├── src # contract's code │ ├── internal │ │ ├── batch_actions.ts │ │ ├── constants.ts │ │ ├── multiple_contracts.ts │ │ ├── similar_contracts.ts │ │ └── utils.ts │ └── contract.ts ├── package.json ├── README.md └── tsconfig.json ┌── tests # sandbox testing │ ├── external-contracts │ │ ├── counter.wasm │ │ ├── guest-book.wasm │ │ └── hello-near.wasm │ └── main.ava.ts ├── src # contract's code │ ├── batch_actions.rs │ ├── lib.rs │ ├── multiple_contracts.rs │ └── similar_contracts.rs ├── Cargo.toml # package manager ├── README.md └── rust-toolchain.toml

Smart Contract

Batch Actions

You can aggregate multiple actions directed towards one same contract into a batched transaction. Methods called this way are executed sequentially, with the added benefit that, if one fails then theyall get reverted .

- JavaScript
- Rust
- contract.ts
- batch_actions.ts

contract-advanced-ts/src/contract.ts loading ... See full example on GitHubcontract-advanced-ts/src/internal/batch_actions.ts loading ... See full example on GitHub contract-advanced-rs/src/batch_actions.rs loading ... See full example on GitHub

Getting the Last Response

In this case, the callback has access to the value returned by thelast action from the chain.

- JavaScript
- Rust
- contract.ts
- batch_actions.ts
- utils.ts

contract-advanced-ts/src/contract.ts loading ... See full example on GitHubcontract-advanced-ts/src/internal/batch_actions.ts loading ... See full example on GitHub contract-advanced-ts/src/internal/utils.ts loading ... See full example on GitHubcontract-advanced-rs/src/batch_actions.rs loading ... See full example on GitHub

Calling Multiple Contracts

A contract can call multiple other contracts. This creates multiple transactions that execute all in parallel. If one of them fails the restARE NOT REVERTED .

- JavaScript
- Rust

- contract.ts
- multiple_contracts.ts

contract-advanced-ts/src/contract.ts loading ... See full example on GitHub contract-advanced-ts/src/internal/multiple_contracts.ts loading ... See full example on GitHub contract-advanced-rs/src/multiple_contracts.rs loading ... See full example on GitHub

Getting All Responses

In this case, the callback has access to anarray of responses , which have either the value returned by each call, or an error message.

- JavaScript
- Rust
- contract.ts
- multiple_contracts.ts
- utils.ts

contract-advanced-ts/src/contract.ts loading ... See full example on GitHub contract-advanced-ts/src/internal/multiple_contracts.ts loading ... See full example on GitHub contract-advanced-ts/src/internal/utils.ts loading ... See full example on GitHub contract-advanced-rs/src/multiple_contracts.rs loading ... See full example on GitHub

Multiple Calls - Same Result Type

This example is a particular case of the previous one (Calling Multiple Contracts). It simply showcases a different way to check the results by directly accessing thepromise_result array.

In this case, we call multiple contracts that will return the same type:

- JavaScript
- Rust
- contract.ts
- similar_contracts.ts

contract-advanced-ts/src/contract.ts loading ... See full example on GitHub contract-advanced-ts/src/internal/similar_contracts.ts loading ... See full example on GitHub contract-advanced-rs/src/similar_contracts.rs loading ... See full example on GitHub

Getting All Responses

In this case, the callback again has access to anarray of responses , which we can iterate checking the results.

- JavaScript
- Rust
- contract.ts
- similar_contracts.ts
- utils.ts

contract-advanced-ts/src/contract.ts loading ... See full example on GitHub contract-advanced-ts/src/internal/similar_contracts.ts loading ... See full example on GitHub contract-advanced-ts/src/internal/utils.ts loading ... See full example on GitHub contract-advanced-rs/src/similar_contracts.rs loading ... See full example on GitHub

Testing the Contract

The contract readily includes a set of unit and sandbox testing to validate its functionality. To execute the tests, run the following commands:

- JavaScript
- Rust

cd contract-advanced-ts yarn yarn test cd contract-advanced-rs cargo test tip Theintegration tests use a sandbox to create NEAR users and simulate interactions with the contract.

Deploying the Contract to the NEAR network

In order to deploy the contract you will need tocreate a NEAR account .

- JavaScript
- Rust

Optional - create an account

near create-account --useFaucet

Deploy the contract

cd contract-advanced-ts yarn build near deploy ./build/cross_contract.wasm --initFunction init --initArgs '{"hello_account":"hello.near-example.testnet","guestbook_account":"guestbook_account.near-example.testnet","counter_account":"counter_account.near-example.testnet"}'

Optional - create an account

near create-account --useFaucet

Deploy the contract

cd contract-advanced-rs cargo near build

During deploying pass {"hello_account":"hello.near-example.testnet","guestbook_account":"guestbook_account.near-example.testnet","counter_account":"counter_account.near-example.testnet"} as init arguments

cargo near deploy

CLI: Interacting with the Contract

To interact with the contract through the console, you can use the following commands:

Execute contracts sequentially

Replace with your account ID

near call batch_actions --accountId --gas 300000000000000

Execute contracts in parallel

Replace with your account ID

near call multiple_contracts --accountId --gas 300000000000000

Execute multiple instances of the same contract in parallel

Replace with your account ID

near call similar_contracts --accountId --gas 300000000000000 Note If the contract exceeds the execution time, additional gas must be provided. For further detailsclick here . Edit this page Last updatedonMar 15, 2024 bygarikbesson Was this page helpful? Yes No

Previous Factory Next Self Upgrade & State Migration