

Cross Contract Call

This example performs the simplest cross-contract call possible: it calls out [Hello NEAR](#) example to set and retrieve a greeting. It is one of the simplest examples on making a cross-contract call, and the perfect gateway to the world of interoperative contracts.

Advanced Cross-Contract Calls Check the tutorial on how to perform cross-contract calls [in batches and in parallel](#)

Obtaining the Cross Contract Call Example

You have two options to start the project:

1. You can use the app through [Github 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 | └── hello-near | | └── hello-near.wasm | └── main.ava.ts |── src # contract's code | └── contract.ts |── package.json |── README.md |── tsconfig.json =====

└── tests # sandbox testing | └── hello-near | | └── hello-near.wasm | └── tests.rs |── src # contract's code | └── external.rs | └── lib.rs |── Cargo.toml # package manager |── README.md |── rust-toolchain.toml

Smart Contract

Contract

The contract exposes methods to query the greeting and change it. These methods do nothing but calling `get_greeting` and `set_greeting` in the `hello-near` example.

- JavaScript
- Rust

contract-simple-ts/src/contract.ts loading ... [See full example on GitHub](#) * lib.rs * external.rs

contract-simple-rs/src/lib.rs loading ... [See full example on GitHub](#) contract-simple-rs/src/external.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-simple-ts yarn yarn test cd contract-simple-rs cargo test` tip The integration tests use a sandbox to create NEAR users and simulate interactions with the contract. In this project in particular, the integration tests first deploy the `hello-near` contract. Then, they test that the cross-contract call correctly sets and retrieves the message. You will find the integration tests in `sandbox-ts/` for the JavaScript version and `intests/` for the Rust version.

- JavaScript
- Rust

contract-simple-ts/sandbox-ts/src/main.ava.ts loading ... [See full example on GitHub](#) contract-simple-rs/tests/tests.rs loading ... [See full example on GitHub](#)

Deploying the Contract to the NEAR network

In order to deploy the contract you will need to [create a NEAR account](#).

• JavaScript
• Rust

Optional - create an account

```
near create-account --useFaucet
```

Deploy the contract

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

Optional - create an account

```
near create-account --useFaucet
```

Deploy the contract

```
cd contract-simple-rs
```

```
cargo near build
```

During deploying pass {"hello_account":"hello.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:

Get message from the hello-near contract

Replace with your account ID

```
near call query_greeting --accountId
```

Set a new message for the hello-near contract

Replace with your account ID

```
near call change_greeting '{"new_greeting":"XCC Hi"}' --accountId
```

Moving Forward

A nice way to learn is by trying to expand a contract. Modify the cross contract example to use the [quest-book](#) contract!. In this way, you can try to make a cross-contract call that attaches money. Remember to correctly [handle the callback](#), and to return the money to the user in case of error.

Advanced Cross Contract Calls

Your contract can perform multiple cross-contract calls in simultaneous, creating promises that execute in parallel, or as a batch transaction. Check the [advanced cross contract calls tutorial](#) to learn more. [Edit this page](#) Last updated on Mar 15, 2024 by garikbesson Was this page helpful? Yes No

[Previous Donation](#) [Next Coin Flip](#)