Rust SDK

The Sui Rust SDK crate is in the crates\sui-sdk directory of the Sui repository.

This crate provides the Sui Rust SDK, containing APIs to interact with the Sui network. Auto-generated documentation for this crate is here .

Add the sui-sdk dependency as following:

The main building block for the Sui Rust SDK is the SuiClientBuilder , which provides a simple and straightforward way of connecting to a Sui network and having access to the different available APIs.

In the following example, the application connects to the Sui testnet and devnet networks and prints out their respective RPC API versions.

GitHub Pages hosts the generated documentation for all Rust crates in the Sui repository.

You can also build the documentation locally. To do so,

Clone the sui repo locally. Open a Terminal or Console and go to the sui/crates/sui-sdk directory.

Run cargo doc to build the documentation into the sui/target directory. Take note of location of the generated file from the last line of the output, for example Generated /Users/foo/sui/target/doc/sui_sdk/index.html .

Use a web browser, like Chrome, to open the .../target/doc/sui_sdk/index.html file at the location your console reported in the previous step.

The examples folder provides both basic and advanced examples.

There are serveral files ending in _api.rs which provide code examples of the corresponding APIs and their methods. These showcase how to use the Sui Rust SDK, and can be run against the Sui testnet. Below are instructions on the prerequisites and how to run these examples.

Unless otherwise specified, most of these examples assume Rust and cargo are installed, and that there is an available internet connection. The examples connect to the Sui testnet (https://fullnode.testnet.sui.io:443) and execute different APIs using the active address from the local wallet. If there is no local wallet, it will create one, generate two addresses, set one of them to be active, and it will request 1 SUI from the testnet faucet for the active address.

In the root folder of the sui repository (or in the sui-sdk crate folder), you can individually run examples using the command cargo run --example filename (without .rs extension). For example:

The SuiClientBuilder struct provides a connection to the JSON-RPC server that you use for all read-only operations. The default URLs to connect to the Sui network are:

For all available servers, see here .

For running a local Sui network, please follow this guide for installing Sui and this guide for starting the local Sui network.

See the programmable transactions example .

Prepare the environment

Publish the move contract

Create a new tic-tac-toe game

Making a move

Run the following command in the tic-tac-toe/cli directory to make a move in an existing game, as the active address in the CLI, replacing the game ID and address accordingly:

Getting started

Add the sui-sdk dependency as following:

The main building block for the Sui Rust SDK is the SuiClientBuilder , which provides a simple and straightforward way of connecting to a Sui network and having access to the different available APIs.

In the following example, the application connects to the Sui testnet and devnet networks and prints out their respective RPC API versions.

GitHub Pages hosts the generated documentation for all Rust crates in the Sui repository.

You can also build the documentation locally. To do so,

Clone the sui repo locally. Open a Terminal or Console and go to the sui/crates/sui-sdk directory.

Run cargo doc to build the documentation into the sui/target directory. Take note of location of the generated file from the last line of the output, for example Generated /Users/foo/sui/target/doc/sui_sdk/index.html .

Use a web browser, like Chrome, to open the .../target/doc/sui_sdk/index.html file at the location your console reported in the previous step.

The examples folder provides both basic and advanced examples.

There are serveral files ending in _api.rs which provide code examples of the corresponding APIs and their methods. These showcase how to use the Sui Rust SDK, and can be run against the Sui testnet. Below are instructions on the prerequisites and how to run these examples.

Unless otherwise specified, most of these examples assume Rust and cargo are installed, and that there is an available internet connection. The examples connect to the Sui testnet (https://fullnode.testnet.sui.io:443) and execute different APIs using the active address from the local wallet. If there is no local wallet, it will create one, generate two addresses, set one of them to be active, and it will request 1 SUI from the testnet faucet for the active address.

In the root folder of the sui repository (or in the sui-sdk crate folder), you can individually run examples using the command cargo run --example filename (without .rs extension). For example:

The SuiClientBuilder struct provides a connection to the JSON-RPC server that you use for all read-only operations. The default URLs to connect to the Sui network are:

For all available servers, see here .

For running a local Sui network, please follow this guide for installing Sui and this guide for starting the local Sui network.

See the programmable transactions example .

Prepare the environment

Publish the move contract

Create a new tic-tac-toe game

Making a move

Run the following command in the tic-tac-toe/cli directory to make a move in an existing game, as the active address in the CLI, replacing the game ID and address accordingly:

SPDX-License-Identifier: Apache-2.0

Documentation for sui-sdk crate

GitHub Pages hosts the generated documentation for all Rust crates in the Sui repository.

You can also build the documentation locally. To do so,

Clone the sui repo locally. Open a Terminal or Console and go to the sui/crates/sui-sdk directory.

Run cargo doc to build the documentation into the sui/target directory. Take note of location of the generated file from the last line of

the output, for example Generated /Users/foo/sui/target/doc/sui_sdk/index.html .

Use a web browser, like Chrome, to open the .../target/doc/sui_sdk/index.html file at the location your console reported in the previous step.

The examples folder provides both basic and advanced examples.

There are serveral files ending in _api.rs which provide code examples of the corresponding APIs and their methods. These showcase how to use the Sui Rust SDK, and can be run against the Sui testnet. Below are instructions on the prerequisites and how to run these examples.

Unless otherwise specified, most of these examples assume Rust and cargo are installed, and that there is an available internet connection. The examples connect to the Sui testnet (https://fullnode.testnet.sui.io:443) and execute different APIs using the active address from the local wallet. If there is no local wallet, it will create one, generate two addresses, set one of them to be active, and it will request 1 SUI from the testnet faucet for the active address.

In the root folder of the sui repository (or in the sui-sdk crate folder), you can individually run examples using the command cargo run --example filename (without .rs extension). For example:

The SuiClientBuilder struct provides a connection to the JSON-RPC server that you use for all read-only operations. The default URLs to connect to the Sui network are:

For all available servers, see here .

For running a local Sui network, please follow this guide for installing Sui and this guide for starting the local Sui network.

See the programmable transactions example .

Prepare the environment

Publish the move contract

Create a new tic-tac-toe game

Making a move

Run the following command in the tic-tac-toe/cli directory to make a move in an existing game, as the active address in the CLI, replacing the game ID and address accordingly:

SPDX-License-Identifier: Apache-2.0

Rust SDK examples

The examples folder provides both basic and advanced examples.

There are serveral files ending in _api.rs which provide code examples of the corresponding APIs and their methods. These showcase how to use the Sui Rust SDK, and can be run against the Sui testnet. Below are instructions on the prerequisites and how to run these examples.

Unless otherwise specified, most of these examples assume Rust and cargo are installed, and that there is an available internet connection. The examples connect to the Sui testnet (https://fullnode.testnet.sui.io:443) and execute different APIs using the active address from the local wallet. If there is no local wallet, it will create one, generate two addresses, set one of them to be active, and it will request 1 SUI from the testnet faucet for the active address.

In the root folder of the sui repository (or in the sui-sdk crate folder), you can individually run examples using the command cargo run --example filename (without .rs extension). For example:

The SuiClientBuilder struct provides a connection to the JSON-RPC server that you use for all read-only operations. The default URLs to connect to the Sui network are:

For all available servers, see here .

For running a local Sui network, please follow this guide for installing Sui and this guide for starting the local Sui network.

See the programmable transactions example .

Prepare the environment

Publish the move contract

Create a new tic-tac-toe game

Making a move

Run the following command in the tic-tac-toe/cli directory to make a move in an existing game, as the active address in the CLI, replacing the game ID and address accordingly:

SPDX-License-Identifier: Apache-2.0

Advanced examples

See the programmable transactions example .

Prepare the environment

Publish the move contract

Create a new tic-tac-toe game

Making a move

Run the following command in the tic-tac-toe/cli directory to make a move in an existing game, as the active address in the CLI, replacing the game ID and address accordingly:

SPDX-License-Identifier: Apache-2.0

Games examples

Prepare the environment

Publish the move contract

Create a new tic-tac-toe game

Making a move

Run the following command in the tic-tac-toe/cli directory to make a move in an existing game, as the active address in the CLI, replacing the game ID and address accordingly:

SPDX-License-Identifier: Apache-2.0

License

SPDX-License-Identifier: Apache-2.0