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.

Getting started

Add the `sui-sdk` dependency as following:

```
sui_sdk = { git = "https://github.com/mystenlabs/sui", package = "sui-sdk"}
tokio = { version = "1.2", features = ["full"] }
anyhow = "1.0"
```

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.

```
use sui_sdk::SuiClientBuilder;

#[tokio::main]
async fn main() -> Result<(), anyhow::Error> {
    // Sui testnet -- https://fullnode.testnet.sui.io:443
    let sui_testnet = SuiClientBuilder::default().build_testnet().await?;
    println!("Sui testnet version: {}", sui_testnet.api_version());

     // Sui devnet -- https://fullnode.devnet.sui.io:443
    let sui_devnet = SuiClientBuilder::default().build_devnet().await?;
    println!("Sui devnet version: {}", sui_devnet.api_version());

    // Sui mainnet -- https://fullnode.mainnet.sui.io:443
    let sui_mainnet = SuiClientBuilder::default().build_mainnet().await?;
    println!("Sui mainnet version: {}", sui_mainnet.api_version());

    Ok(())
}
```

Documentation for sui-sdk crate

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

Building documentation locally

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

1. Clone the `sui` repo locally. Open a Terminal or Console and go to the `sui/crates/sui-sdk` directory.
2. 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`.
3. 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.

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.

Prerequisites

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.

Running the existing examples

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:

- `cargo run --example sui_client` -- this one requires a local Sui network running (see [here](#Connecting to Sui Network)). If you do not have a local Sui network running, please skip this example.
- `cargo run --example coin_read_api`
- `cargo run --example event_api` -- note that this will subscribe to a stream and thus the program will not terminate unless forced (Ctrl+C)
- `cargo run --example governance_api`
- `cargo run --example read_api`
- `cargo run --example programmable_transactions_api`

- `cargo run --example sign_tx_guide`

Basic Examples

Connecting to Sui Network

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:

- Local: http://127.0.0.1:9000
- Devnet: https://fullnode.devnet.sui.io:443
- Testnet: https://fullnode.testnet.sui.io:443
- Mainnet: https://fullnode.mainnet.sui.io:443

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.

```
use sui_sdk::SuiClientBuilder;

#[tokio::main]
async fn main() -> Result<(), anyhow::Error> {
    let sui = SuiClientBuilder::default()
        .build("http://127.0.0.1:9000") // local network address
        .await?;
    println!("Sui local network version: {}", sui.api_version());

    // local Sui network, like the above one but using the dedicated function
    let sui_local = SuiClientBuilder::default().build_localnet().await?;
    println!("Sui local network version: {}", sui_local.api_version());

    // Sui devnet -- https://fullnode.devnet.sui.io:443
    let sui_devnet = SuiClientBuilder::default().build_devnet().await?;
    println!("Sui devnet version: {}", sui_devnet.api_version());

    // Sui testnet -- https://fullnode.testnet.sui.io:443
    let sui_testnet = SuiClientBuilder::default().build_testnet().await?;
    println!("Sui testnet version: {}", sui_testnet.api_version());

    Ok(())
}
```

Read the total coin balance for each coin type owned by this address

```
use std::str::FromStr;
use sui_sdk::types::base_types::SuiAddress;
use sui_sdk::{ SuiClientBuilder};
```

```rust
#[tokio::main]
async fn main() -> Result<(), anyhow::Error> {

    let sui_local = SuiClientBuilder::default().build_localnet().await?;
    println!("Sui local network version: {}", sui_local.api_version());

    let active_address = SuiAddress::from_str("<YOUR SUI ADDRESS>")?; // change
to your Sui address

    let total_balance = sui_local
        .coin_read_api()
        .get_all_balances(active_address)
        .await?;
    println!("The balances for all coins owned by address: {active_address} are
{}", total_balance);
    Ok(())
}
```

Advanced examples

See the programmable transactions example.

Games examples

Tic Tac Toe quick start

1. Prepare the environment

    i. Install `sui` binary following the Sui installation docs.

    ii. Connect to Sui Devnet.

    iii. Make sure you have two addresses with gas by using the `new-address` command to create new addresses:

    ```
    sui client new-address ed25519
    ```

    You must specify the key scheme, one of `ed25519` or `secp256k1` or `secp256r1`. You can skip this step if you are going to play with a friend. :)

    iv. Request Sui tokens for all addresses that will be used to join the game.

2. Publish the move contract

    i. Download the Sui source code.

    ii. Publish the `tic-tac-toe` package using the Sui client:

    ```
    sui client publish --path /path-to-sui-source-code/examples/tic-tac-
    toe/move
    ```

    iii. Record the package object ID.

3. Create a new tic-tac-toe game

   i. Run the following command in the `tic-tac-toe/cli` directory to start a new game, replacing the game package objects ID with the one you recorded:

   ```
   cargo run -- new --package-id <<tic-tac-toe package object ID>> <<player O address>>
   ```

   This will create a game between the active address in the keystore, and the specified Player O.

   ii. Copy the game ID and pass it to your friend to join the game.

4. 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:

   ```
   cargo run -- move --package-id <<tic-tac-toe package object ID>> --row $R --col $C <<game ID>>
   ```

License