Sui Client CLI

The Sui CLI client command provides command-level access to interact with the Sui network. Typical uses for sui client include publishing Move smart contracts, getting the information of an object, executing transactions, or managing addresses.

Before you can use the Sui CLI, you must install it. To check if the CLI exists on your system, open a terminal or console and type the following command:

If the terminal or console responds with a version number, you already have the Sui CLI installed.

If the command is not found, follow the instructions in Install Sui to get the Sui CLI on your system.

The following list itemizes all the available subcommands for the sui client command.

Append the --json flag to commands to format responses in JSON instead of the more human-friendly default Sui CLI output. This can be useful for extremely large datasets, for example, as those results can have a troublesome display on smaller screens. In these cases, the --json flag is useful.

The following examples demonstrate some of the most often used commands.

Use the sui client envs command to find the network environments set up in the CLI. The information for these environments is also stored in the client.yaml file in the Sui configuration directory (~/.sui/sui_config).

Use client new-env to add details for a new network environment. This example creates an environment pointer to Sui Mainnet. Setting the alias value makes referencing the environment less prone to typographical errors. After running this command, Sui updates your client.yaml file in ~/.sui/sui_config with the new information.

Use the sui client switch command to change the current network. This example switches the current network to mainnet .

If you run sui client envs after this command, you see the asterisk in the active column on the mainnet row of the table.

Use the sui client active-address command to reveal the current address. The CLI uses the current active address to execute address-specific CLI commands (like sui client objects) when you don't provide them with a Sui address value.

If you use one of the standard public RPCs (for example, fullnode.devnet.sui.io:443), you can use the faucet command to request gas coins. If you use a custom faucet service, then pass in the URL to the faucet using the --url option. The faucet command works for a local network, as well. If you start your network with a custom faucet port, include the --url option.

Use sui client objects to list summary information about the objects the current active address owns. You can provide a Sui address value to the command to list objects for a particular address. This example lists objects for the current active address.

Use sui client object to list object information for the ID you provide. This example displays the information for a Coin object.

Use the sui client dynamic-field command to list the details of the dynamic field with the ID you provide.

In this example, let's see how to transfer SUI or transfer an object from one address to another. First of all, there two main commands for sending SUI or transferring objects: pay and transfer . Both pay and transfer have a few sister commands: pay-sui , pay-all-sui , transfer-sui .

The differences between these commands are:

Assume you have two addresses:

Address hungry-spodumene has a few coins:

You want to send 0.5 SUI to eloquent-amber . Given that you have a few gas coins, you can use pay . If only one gas coin exists, then you need to use transfer-sui or pay-sui , or you would need to split the coin first to have another coin to use for paying gas. In this case, let's use the pay-sui command as you do not need to provide a separate gas coin to be used for the gas fees. In the command below, you set the recipient to be eloquent-amber , which coin to use to transfer SUI from, and the amount of SUI to transfer.

Beginning with the Sui v1.24.1 release , the --gas-budget option is no longer required for CLI commands.

The result is:

Check the gas for the sender and the recipient's coins using sui client gas

. The sender now has 199.49 SUI for that gas coin that was used. 0.5 SUI was transferred, and the remaining 0.01 SUI paid the gas fees.

If you want to transfer the whole object, you can use sui client pay-all-sui or sui client transfer-sui (without passing the amount):

or

Then check the gas for eloquent-amber again:

Use the sui client replay-transaction --tx-digest to re-execute a transaction locally and show the transaction effects. This command will fetch the transaction dependencies from the Full node specified in the client env. For transactions that happened quite far in the past, it is advised to set the client Full node to one that has non-pruned chain data for that transaction. This will also verify that the resulting effects from the locally executed transaction match the effects of the transaction stored on-chain.

You can add additional flags --gas-info and --ptb-info to this command to see more information about the transaction.

Use sui client replay-batch --path to replay several transactions listed in a newline-separated file. This will verify that all transactions local execution results match the effects on-chain.

Use the sui client profile-transaction --tx-digest command to re-execute a transaction locally and produce a gas profile. Similar to the replay command, this command fetches the transaction dependencies from the Full node specified in the client environment that are needed to execute the transaction. During the local execution of the transaction, this command records all the Move function invocations and the gas cost breakdown for each invocation.

To enable the profiler, you must either install or build the Sui Client binary locally with the --features tracing flag if it has not already been built with that feature enabled.

The command outputs a profile to the current working directory in the format gas_profile_{tx_digest}{unix_timestamp}.json . You can include the optional flag --profile-output to write the profile to /PATH/OUTPUT{tx_digest}_{unix_timestamp}.json instead. Use speedscope to inspect the profile.

To install speedscope run

To open a profile in speedscope run

When looking at the profile in speedscope, there are three different views available from the top left menu: Timer Order , Left Heavy , and Sandwich . In each view, each bar's vertical width corresponds to the percentage of gas consumption incurred by the function, and you can hover your mouse over a bar or click a bar to see the computation units accrued by the function invocation. The transaction's total computation units, along with the storage computation units, are multiplied by the gas price to determine the gas cost of the transaction based on a tier system.

Time Order shows the callstack of function invocations from left to right in the order of invocation, while Left Heavy combines repeated sequences of nested invocations into a single combined call stack. Left Heavy displays these sequences from left to right by total incurred gas consumption per combined call stack. This is useful when there have been hundreds of repeated calls to the same function to quickly observe the total gas consumption over all calls to that function. In both these views, you can click the top section and drag to zoom in and out over different sections of the profile.

Sandwich view shows a list of discrete values that correspond to gas consumption per function, with Total showing gas cost incurred in all the functions called by the function, and Self showing the gas cost done by only the given function.

Observing a transaction's gas consumption provides useful insight of expected gas cost usage of a smart contract. When developing a smart contract, you can run a local network and publish the package to the local network. Then create a transaction that calls your published smart contract, and finally run the profiler on the transaction to see a breakdown of the gas cost.

One of the main uses of the sui client command is to publish smart contracts on the Sui network. This example switches the current environment to the Devnet network, then builds, tests, and publishes one of the existing Move examples available in the Sui repository: sui/examples/move

This example also makes use of sui move commands. To learn more about those commands, see Sui Move CLI .

Each command has its own help section. For example, sui client call --help displays the following prompt:

Check Sui CLI installation

Before you can use the Sui CLI, you must install it. To check if the CLI exists on your system, open a terminal or console and type the following command:

If the terminal or console responds with a version number, you already have the Sui CLI installed.

If the command is not found, follow the instructions in Install Sui to get the Sui CLI on your system.

The following list itemizes all the available subcommands for the sui client command.

Append the --json flag to commands to format responses in JSON instead of the more human-friendly default Sui CLI output. This can be useful for extremely large datasets, for example, as those results can have a troublesome display on smaller screens. In these cases, the --json flag is useful.

The following examples demonstrate some of the most often used commands.

Use the sui client envs command to find the network environments set up in the CLI. The information for these environments is also stored in the client.yaml file in the Sui configuration directory (~/.sui/sui_config).

Use client new-env to add details for a new network environment. This example creates an environment pointer to Sui Mainnet. Setting the alias value makes referencing the environment less prone to typographical errors. After running this command, Sui updates your client.yaml file in ~/.sui/sui_config with the new information.

Use the sui client switch command to change the current network. This example switches the current network to mainnet .

If you run sui client envs after this command, you see the asterisk in the active column on the mainnet row of the table.

Use the sui client active-address command to reveal the current address. The CLI uses the current active address to execute address-specific CLI commands (like sui client objects) when you don't provide them with a Sui address value.

If you use one of the standard public RPCs (for example, fullnode.devnet.sui.io:443), you can use the faucet command to request gas coins. If you use a custom faucet service, then pass in the URL to the faucet using the --url option. The faucet command works for a local network, as well. If you start your network with a custom faucet port, include the --url option.

Use sui client objects to list summary information about the objects the current active address owns. You can provide a Sui address value to the command to list objects for a particular address. This example lists objects for the current active address.

Use sui client object to list object information for the ID you provide. This example displays the information for a Coin object.

Use the sui client dynamic-field command to list the details of the dynamic field with the ID you provide.

In this example, let's see how to transfer SUI or transfer an object from one address to another. First of all, there two main commands for sending SUI or transferring objects: pay and transfer . Both pay and transfer have a few sister commands: pay-sui , pay-all-sui , transfer-sui .

The differences between these commands are:

Assume you have two addresses:

Address hungry-spodumene has a few coins:

You want to send 0.5 SUI to eloquent-amber . Given that you have a few gas coins, you can use pay . If only one gas coin exists, then you need to use transfer-sui or pay-sui , or you would need to split the coin first to have another coin to use for paying gas. In this case, let's use the pay-sui command as you do not need to provide a separate gas coin to be used for the gas fees. In the command below, you set the recipient to be eloquent-amber , which coin to use to transfer SUI from, and the amount of SUI to transfer.

Beginning with the Sui v1.24.1 release , the --gas-budget option is no longer required for CLI commands.

The result is:

Check the gas for the sender and the recipient's coins using sui client gas

. The sender now has 199.49 SUI for that gas coin that was used. 0.5 SUI was transferred, and the remaining 0.01 SUI paid the gas fees.

If you want to transfer the whole object, you can use sui client pay-all-sui or sui client transfer-sui (without passing the amount):

or

Then check the gas for eloquent-amber again:

Use the sui client replay-transaction --tx-digest to re-execute a transaction locally and show the transaction effects. This command will fetch the transaction dependencies from the Full node specified in the client env. For transactions that happened quite far in the past, it is advised to set the client Full node to one that has non-pruned chain data for that transaction. This will also verify that the resulting effects from the locally executed transaction match the effects of the transaction stored on-chain.

You can add additional flags --gas-info and --ptb-info to this command to see more information about the transaction.

Use sui client replay-batch --path to replay several transactions listed in a newline-separated file. This will verify that all transactions local execution results match the effects on-chain.

Use the sui client profile-transaction --tx-digest command to re-execute a transaction locally and produce a gas profile. Similar to the replay command, this command fetches the transaction dependencies from the Full node specified in the client environment that are needed to execute the transaction. During the local execution of the transaction, this command records all the Move function invocations and the gas cost breakdown for each invocation.

To enable the profiler, you must either install or build the Sui Client binary locally with the --features tracing flag if it has not already been built with that feature enabled.

The command outputs a profile to the current working directory in the format gas_profile_{tx_digest}{unix_timestamp}.json . You can include the optional flag --profile-output to write the profile to /PATH/OUTPUT{tx_digest}_{unix_timestamp}.json instead. Use speedscope to inspect the profile.

To install speedscope run

To open a profile in speedscope run

When looking at the profile in speedscope, there are three different views available from the top left menu: Timer Order , Left Heavy , and Sandwich . In each view, each bar's vertical width corresponds to the percentage of gas consumption incurred by the function, and you can hover your mouse over a bar or click a bar to see the computation units accrued by the function invocation. The transaction's total computation units, along with the storage computation units, are multiplied by the gas price to determine the gas cost of the transaction based on a tier system.

Time Order shows the callstack of function invocations from left to right in the order of invocation, while Left Heavy combines repeated sequences of nested invocations into a single combined call stack. Left Heavy displays these sequences from left to right by total incurred gas consumption per combined call stack. This is useful when there have been hundreds of repeated calls to the same function to quickly observe the total gas consumption over all calls to that function. In both these views, you can click the top section and drag to zoom in and out over different sections of the profile.

Sandwich view shows a list of discrete values that correspond to gas consumption per function, with Total showing gas cost incurred in all the functions called by the function, and Self showing the gas cost done by only the given function.

Observing a transaction's gas consumption provides useful insight of expected gas cost usage of a smart contract. When developing a smart contract, you can run a local network and publish the package to the local network. Then create a transaction that calls your published smart contract, and finally run the profiler on the transaction to see a breakdown of the gas cost.

One of the main uses of the sui client command is to publish smart contracts on the Sui network. This example switches the current environment to the Devnet network, then builds, tests, and publishes one of the existing Move examples available in the Sui repository: sui/examples/move

This example also makes use of sui move commands. To learn more about those commands, see Sui Move CLI .

Each command has its own help section. For example, sui client call --help displays the following prompt:

Commands

The following list itemizes all the available subcommands for the sui client command.

Append the --json flag to commands to format responses in JSON instead of the more human-friendly default Sui CLI output. This can be useful for extremely large datasets, for example, as those results can have a troublesome display on smaller screens. In these cases, the --json flag is useful.

The following examples demonstrate some of the most often used commands.

Use the sui client envs command to find the network environments set up in the CLI. The information for these environments is also stored in the client.yaml file in the Sui configuration directory (~/.sui/sui_config).

Use client new-env to add details for a new network environment. This example creates an environment pointer to Sui Mainnet. Setting the alias value makes referencing the environment less prone to typographical errors. After running this command, Sui updates your client.yaml file in ~/.sui/sui_config with the new information.

Use the sui client switch command to change the current network. This example switches the current network to mainnet .

If you run sui client envs after this command, you see the asterisk in the active column on the mainnet row of the table.

Use the sui client active-address command to reveal the current address. The CLI uses the current active address to execute address-specific CLI commands (like sui client objects) when you don't provide them with a Sui address value.

If you use one of the standard public RPCs (for example, fullnode.devnet.sui.io:443), you can use the faucet command to request gas coins. If you use a custom faucet service, then pass in the URL to the faucet using the --url option. The faucet command works for a local network, as well. If you start your network with a custom faucet port, include the --url option.

Use sui client objects to list summary information about the objects the current active address owns. You can provide a Sui address value to the command to list objects for a particular address. This example lists objects for the current active address.

Use sui client object to list object information for the ID you provide. This example displays the information for a Coin object.

Use the sui client dynamic-field command to list the details of the dynamic field with the ID you provide.

In this example, let's see how to transfer SUI or transfer an object from one address to another. First of all, there two main commands for sending SUI or transferring objects: pay and transfer . Both pay and transfer have a few sister commands: pay-sui , pay-all-sui , transfer-sui .

The differences between these commands are:

Assume you have two addresses:

Address hungry-spodumene has a few coins:

You want to send 0.5 SUI to eloquent-amber . Given that you have a few gas coins, you can use pay . If only one gas coin exists, then you need to use transfer-sui or pay-sui , or you would need to split the coin first to have another coin to use for paying gas. In this case, let's use the pay-sui command as you do not need to provide a separate gas coin to be used for the gas fees. In the command below, you set the recipient to be eloquent-amber , which coin to use to transfer SUI from, and the amount of SUI to transfer.

Beginning with the Sui v1.24.1 [release](), the --gas-budget option is no longer required for CLI commands.

The result is:

Check the gas for the sender and the recipient's coins using sui client gas

. The sender now has 199.49 SUI for that gas coin that was used. 0.5 SUI was transferred, and the remaining 0.01 SUI paid the gas fees.

If you want to transfer the whole object, you can use sui client pay-all-sui or sui client transfer-sui (without passing the *amount):*

or

Then check the gas for eloquent-amber again:

Use the sui client replay-transaction --tx-digest to re-execute a transaction locally and show the transaction effects. This command will fetch the transaction dependencies from the Full node specified in the client env. For transactions that happened quite far in the past, it is advised to set the client Full node to one that has non-pruned chain data for that transaction. This will also verify that the resulting effects from the locally executed transaction match the effects of the transaction stored on-chain.

You can add additional flags --gas-info and --ptb-info to this command to see more information about the transaction.

Use sui client replay-batch --path to replay several transactions listed in a newline-separated file. This will verify that all transactions local execution results match the effects on-chain.

Use the sui client profile-transaction --tx-digest command to re-execute a transaction locally and produce a gas profile. Similar to the replay command, this command fetches the transaction dependencies from the Full node specified in the client environment that are needed to execute the transaction. During the local execution of the transaction, this command records all the Move function invocations and the gas cost breakdown for each invocation.

To enable the profiler, you must either install or build the Sui Client binary locally with the --features tracing flag if it has not already been built with that feature enabled.

The command outputs a profile to the current working directory in the format gas_profile_{tx_digest}{unix_timestamp}.json . You can include the optional flag --profile-output to write the profile to /PATH/OUTPUT{tx_digest}_{unix_timestamp}.json instead. Use speedscope to inspect the profile.

To install speedscope run

To open a profile in speedscope run

When looking at the profile in speedscope, there are three different views available from the top left menu: Timer Order , Left Heavy , and Sandwich . In each view, each bar's vertical width corresponds to the percentage of gas consumption incurred by the function, and you can hover your mouse over a bar or click a bar to see the computation units accrued by the function invocation. The transaction's total computation units, along with the storage computation units, are multiplied by the gas price to determine the gas cost of the transaction based on a tier system.

Time Order shows the callstack of function invocations from left to right in the order of invocation, while Left Heavy combines repeated sequences of nested invocations into a single combined call stack. Left Heavy displays these sequences from left to right by total incurred gas consumption per combined call stack. This is useful when there have been hundreds of repeated calls to the same function to quickly observe the total gas consumption over all calls to that function. In both these views, you can click the top section and drag to zoom in and out over different sections of the profile.

Sandwich view shows a list of discrete values that correspond to gas consumption per function, with Total showing gas cost incurred in all the functions called by the function, and Self showing the gas cost done by only the given function.

Observing a transaction's gas consumption provides useful insight of expected gas cost usage of a smart contract. When developing a smart contract, you can run a local network and publish the package to the local network. Then create a transaction that calls your published smart contract, and finally run the profiler on the transaction to see a breakdown of the gas cost.

One of the main uses of the sui client command is to publish smart contracts on the Sui network. This example switches the current environment to the Devnet network, then builds, tests, and publishes one of the existing Move examples available in the Sui repository: sui/examples/move

This example also makes use of sui move commands. To learn more about those commands, see Sui Move CLI .

Each command has its own help section. For example, sui client call --help displays the following prompt:

JSON output

Append the --json flag to commands to format responses in JSON instead of the more human-friendly default Sui CLI *output. This can be useful for extremely large datasets, for example, as those results can have a troublesome display on smaller screens. In these cases, the --json flag is useful.*

The following examples demonstrate some of the most often used commands.

Use the sui client envs command to find the network environments set up in the CLI. The information for these environments is also stored in the client.yaml file in the Sui configuration directory (~/.sui/sui_config).

Use client new-env to add details for a new network environment. This example creates an environment pointer to Sui Mainnet. Setting the alias value makes referencing the environment less prone to typographical errors. After running this command, Sui updates your client.yaml file in ~/.sui/sui_config with the new information.

Use the sui client switch command to change the current network. This example switches the current network to mainnet .

If you run sui client envs after this command, you see the asterisk in the active column on the mainnet row of the table.

Use the sui client active-address command to reveal the current address. The CLI uses the current active address to execute address-specific CLI commands (like sui client objects) when you don't provide them with a Sui address value.

If you use one of the standard public RPCs (for example, fullnode.devnet.sui.io:443), you can use the faucet command to request gas coins. If you use a custom faucet service, then pass in the URL to the faucet using the --url option. The faucet command works for a local network, as well. If you start your network with a custom faucet port, include the --url option.

Use sui client objects to list summary information about the objects the current active address owns. You can provide a Sui address value to the command to list objects for a particular address. This example lists objects for the current active address.

Use sui client object to list object information for the ID you provide. This example displays the information for a Coin object.

Use the sui client dynamic-field command to list the details of the dynamic field with the ID you provide.

In this example, let's see how to transfer SUI or transfer an object from one address to another. First of all, there two main commands for sending SUI or transferring objects: pay and transfer . Both pay and transfer have a few sister commands: pay-sui , pay-all-sui , transfer-sui .

The differences between these commands are:

Assume you have two addresses:

Address hungry-spodumene has a few coins:

You want to send 0.5 SUI to eloquent-amber . Given that you have a few gas coins, you can use pay . If only one gas coin exists, then you need to use transfer-sui or pay-sui , or you would need to split the coin first to have another coin to use for paying gas. In this case, let's use the pay-sui command as you do not need to provide a separate gas coin to be used for the gas fees. In the command below, you set the recipient to be eloquent-amber , which coin to use to transfer SUI from, and the amount of SUI to transfer.

Beginning with the Sui v1.24.1 [release](), the --gas-budget option is no longer required for CLI commands.

The result is:

Check the gas for the sender and the recipient's coins using sui client gas

. The sender now has 199.49 SUI for that gas coin that was used. 0.5 SUI was transferred, and the remaining 0.01 SUI paid the gas fees.

If you want to transfer the whole object, you can use sui client pay-all-sui or sui client transfer-sui (without passing the amount):

or

Then check the gas for eloquent-amber again:

Use the sui client replay-transaction --tx-digest to re-execute a transaction locally and show the transaction effects. This *command will fetch the transaction dependencies from the Full node specified in the client env. For transactions that happened quite far in the past, it is advised to set the client Full node to one that has non-pruned chain data for that transaction. This will also verify that the resulting effects from the locally executed transaction match the effects of the transaction stored on-chain.*

You can add additional flags --gas-info and --ptb-info to this command to see more information about the transaction.

Use sui client replay-batch --path to replay several transactions listed in a newline-separated file. This will verify that all transactions local execution results match the effects on-chain.

Use the sui client profile-transaction --tx-digest command to re-execute a transaction locally and produce a gas profile. Similar to the replay command, this command fetches the transaction dependencies from the Full node specified in the client environment that are needed to execute the transaction. During the local execution of the transaction, this command records all the Move function invocations and the gas cost breakdown for each invocation.

To enable the profiler, you must either install or build the Sui Client binary locally with the --features tracing flag if it has not already been built with that feature enabled.

The command outputs a profile to the current working directory in the format gas_profile_{tx_digest}{unix_timestamp}.json . You can include the optional flag --profile-output to write the profile to /PATH/OUTPUT{tx_digest}_{unix_timestamp}.json instead. Use speedscope to inspect the profile.

To install speedscope run

To open a profile in speedscope run

When looking at the profile in speedscope, there are three different views available from the top left menu: Timer Order , Left Heavy , and Sandwich . In each view, each bar's vertical width corresponds to the percentage of gas consumption incurred by the function, and you can hover your mouse over a bar or click a bar to see the computation units accrued by the function invocation. The transaction's total computation units, along with the storage computation units, are multiplied by the gas price to determine the gas cost of the transaction based on a tier system.

Time Order shows the callstack of function invocations from left to right in the order of invocation, while Left Heavy combines repeated sequences of nested invocations into a single combined call stack. Left Heavy displays these sequences from left to right by total incurred gas consumption per combined call stack. This is useful when there have been hundreds of repeated calls to the same function to quickly observe the total gas consumption over all calls to that function. In both these views, you can click the top section and drag to zoom in and out over different sections of the profile.

Sandwich view shows a list of discrete values that correspond to gas consumption per function, with Total showing gas cost incurred in all the functions called by the function, and Self showing the gas cost done by only the given function.

Observing a transaction's gas consumption provides useful insight of expected gas cost usage of a smart contract. When developing a smart contract, you can run a local network and publish the package to the local network. Then create a transaction that calls your published smart contract, and finally run the profiler on the transaction to see a breakdown of the gas cost.

One of the main uses of the sui client command is to publish smart contracts on the Sui network. This example switches the current environment to the Devnet network, then builds, tests, and publishes one of the existing Move examples available in the Sui repository: sui/examples/move

This example also makes use of sui move commands. To learn more about those commands, see Sui Move CLI .

Each command has its own help section. For example, sui client call --help displays the following prompt:

Examples

The following examples demonstrate some of the most often used commands.

Use the sui client envs command to find the network environments set up in the CLI. The information for these environments is also stored in the client.yaml file in the Sui configuration directory (~/.sui/sui_config).

Use client new-env to add details for a new network environment. This example creates an environment pointer to Sui Mainnet. Setting the alias value makes referencing the environment less prone to typographical errors. After running this command, Sui updates your client.yaml file in ~/.sui/sui_config with the new information.

Use the sui client switch command to change the current network. This example switches the current network to mainnet .

If you run sui client envs after this command, you see the asterisk in the active column on the mainnet row of the table.

Use the sui client active-address command to reveal the current address. The CLI uses the current active address to execute address-specific CLI commands (like sui client objects) when you don't provide them with a Sui address value.

If you use one of the standard public RPCs (for example, fullnode.devnet.sui.io:443), you can use the faucet command to request gas coins. If you use a custom faucet service, then pass in the URL to the faucet using the --url option. The faucet command works for a local network, as well. If you start your network with a custom faucet port, include the --url option.

Use sui client objects to list summary information about the objects the current active address owns. You can provide a Sui address value to the command to list objects for a particular address. This example lists objects for the current active address.

Use sui client object to list object information for the ID you provide. This example displays the information for a Coin object.

Use the sui client dynamic-field command to list the details of the dynamic field with the ID you provide.

In this example, let's see how to transfer SUI or transfer an object from one address to another. First of all, there two main commands for sending SUI or transferring objects: pay and transfer . Both pay and transfer have a few sister commands: pay-sui , pay-all-sui , transfer-sui .

The differences between these commands are:

Assume you have two addresses:

Address hungry-spodumene has a few coins:

You want to send 0.5 SUI to eloquent-amber . Given that you have a few gas coins, you can use pay . If only one gas coin exists, then you need to use transfer-sui or pay-sui , or you would need to split the coin first to have another coin to use for paying gas. In this case, let's use the pay-sui command as you do not need to provide a separate gas coin to be used for the gas fees. In the command below, you set the recipient to be eloquent-amber , which coin to use to transfer SUI from, and the amount of SUI to transfer.

Beginning with the Sui v1.24.1 release , the --gas-budget option is no longer required for CLI commands.

The result is:

Check the gas for the sender and the recipient's coins using sui client gas

. The sender now has 199.49 SUI for that gas coin that was used. 0.5 SUI was transferred, and the remaining 0.01 SUI paid the gas fees.

If you want to transfer the whole object, you can use sui client pay-all-sui or sui client transfer-sui (without passing the amount):

or

Then check the gas for eloquent-amber again:

Use the sui client replay-transaction --tx-digest to re-execute a transaction locally and show the transaction effects. This command will fetch the transaction dependencies from the Full node specified in the client env. For transactions that happened quite far in the past, it is advised to set the client Full node to one that has non-pruned chain data for that transaction. This will also verify that the resulting effects from the locally executed transaction match the effects of the transaction stored on-chain.

You can add additional flags --gas-info and --ptb-info to this command to see more information about the transaction.

Use sui client replay-batch --path to replay several transactions listed in a newline-separated file. This will verify that all transactions local execution results match the effects on-chain.

Use the sui client profile-transaction --tx-digest command to re-execute a transaction locally and produce a gas profile. Similar to the replay command, this command fetches the transaction dependencies from the Full node specified in the client environment that are needed to execute the transaction. During the local execution of the transaction, this command records all the Move function invocations and the gas cost breakdown for each invocation.

To enable the profiler, you must either install or build the Sui Client binary locally with the --features tracing flag if it has not already been built with that feature enabled.

The command outputs a profile to the current working directory in the format gas_profile_{tx_digest}{unix_timestamp}.json . You can include the optional flag --profile-output to write the profile to /PATH/OUTPUT{tx_digest}_{unix_timestamp}.json instead. Use [speedscope] to inspect the profile.

To install speedscope run

To open a profile in speedscope run

When looking at the profile in speedscope, there are three different views available from the top left menu: Timer Order , Left Heavy , and Sandwich . In each view, each bar's vertical width corresponds to the percentage of gas consumption incurred by the function, and you can hover your mouse over a bar or click a bar to see the computation units accrued by the function invocation. The transaction's total computation units, along with the storage computation units, are multiplied by the gas price to determine the gas cost of the transaction based on a tier system.

Time Order shows the callstack of function invocations from left to right in the order of invocation, while Left Heavy combines repeated sequences of nested invocations into a single combined call stack. Left Heavy displays these sequences from left to right by total incurred gas consumption per combined call stack. This is useful when there have been hundreds of repeated calls to the same function to quickly observe the total gas consumption over all calls to that function. In both these views, you can click the top section and drag to zoom in and out over different sections of the profile.

Sandwich view shows a list of discrete values that correspond to gas consumption per function, with Total showing gas cost incurred in all the functions called by the function, and Self showing the gas cost done by only the given function.

Observing a transaction's gas consumption provides useful insight of expected gas cost usage of a smart contract. When developing a smart contract, you can run a local network and publish the package to the local network. Then create a transaction that calls your published smart contract, and finally run the profiler on the transaction to see a breakdown of the gas cost.

One of the main uses of the sui client command is to publish smart contracts on the Sui network. This example switches the current environment to the Devnet network, then builds, tests, and publishes one of the existing Move examples available in the Sui repository: sui/examples/move

This example also makes use of sui move commands. To learn more about those commands, see Sui Move CLI .

Each command has its own help section. For example, sui client call --help displays the following prompt:

Publish a Move package

One of the main uses of the sui client command is to publish smart contracts on the Sui network. This example switches the current environment to the Devnet network, then builds, tests, and publishes one of the existing Move examples available in the Sui repository: sui/examples/move

This example also makes use of sui move commands. To learn more about those commands, see Sui Move CLI .

Each command has its own help section. For example, sui client call --help displays the following prompt:

Help

Each command has its own help section. For example, sui client call --help displays the following prompt: