Multisig Authentication

The following steps demonstrate how to create a multisig transaction and then submit it against a network using the Sui CLI . A transaction can be the transfer of an object, the publish or upgrade of a package, the payment of SUI, and so on.

To learn more about how to create multisig addresses and create multisig transactions using the TypeScript SDK, see the SDK documentation for details.

This topic assumes you are somewhat familiar with the Sui CLI, specifically the sui client and sui keytool commands. Consequently, a command might be introduced without any context. If you are unsure about the details of a referenced command, see the Sui CLI documentation for more information.

You need an existing address on the network you are working on to receive an object. The topic assumes that this address is the current active address (sui client active-address), but any address you have access to is fine.

The topic also assumes that your active environment is Testnet (sui client active-env). You can perform these steps on Devnet or a local network as well, but you must adjust the instructions appropriately.

To demonstrate multisig, this topic guides you through setting up and executing a multisig transaction using the Sui CLI.

To begin, create three addresses that will act as the signers for the transaction you perform later in the instruction. Use the sui client new-address command to generate a Sui address and public key for three supported key schemes.

The console displays a response to each successful call that is similar to the following:

When working with blockchain data, addresses and hashed values create large strings that can be cumbersome to work with in a CLI environment. To make referencing values easier in subsequent commands (and facilitate copy and paste), this topic uses shell variables. Use the values you receive from the console responses to set shell variables for these addresses, replacing the variables with the appropriate address.

Create one more shell variable assigned to your active address.

You can set the shell variables to the alias values instead of addresses if you want.

Use sui keytool to list the addresses you created in the previous section.

The response resembles the following, but displays actual alias names, addresses, keys, and peer IDs:

The output includes public key data that you use later, so create shell variables to store the information. Don't forget to replace with the actual values you receive from the previous console response.

To sign a transaction using multisig, you need to create a multisig address using sui keytool multi-sig-address . The multisig address is created using the public keys from each individual participating address. Each address is also assigned a weight value that determines how many are needed to create a valid signature. When summed, the weight of the included signatures must be greater than or equal to the threshold value you also set with the command. For this example, use the following command, which states that the first two addresses require at least one more signature to create a valid multisig. The last address has a weight of 3 , which is equal to the threshold value, so its owner can create a valid signature without the others.

The response resembles the following:

Before getting SUI, set a MULTISIG shell variable to the multisig address provided at the top of the previous response (substituting the actual address for).

If you use the sui client objects $MULTISIG command, you can see that the newly created multisig address has no objects. This means you need to get SUI before you can perform any transactions. To get SUI for your multisig account, use the sui client faucet command and provide the multisig address using the --address flag. Run this command twice so that the multisig address owns at least two SUI. This example uses two SUI so that one can be transferred and the other can pay for gas.

Use the sui client gas command to verify the address now has at least two SUI.

It's now time to transfer an object from the multisig address. For simplicity, this example uses one of the coins your multisig address owns as the transfer object. Copy the object ID for one of the address' coins and use it to set a shell variable value.

Use the sui client transfer command to set up the transfer. The --serialize-unsigned-transaction flag outputs the Base64-encoded

transaction bytes.

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

The console displays the result ( ), which you can assign to another shell variable.

Use the sui keytool sign command to sign the transaction using two of the addresses you created previously.

You can create the signature with other tools, as well, as long as you serialize it to flag || sig || pk .

Each successful call to the command receives a response similar to the following.

Create two more shell variables to store the signatures, replacing with the values from the previous command responses.

As mentioned, the multisig must be composed of enough individual signatures such that the sum of the participating signer weights is greater than the threshold value. Use the sui keytool multi-sig-combine-partial-sig command to combine the ed25519 signature (weight: 1) and the secp256k1 (weight: 2). To complete the command, you must provide all public keys, their weights, and the threshold that defined the multisig address.

If successful, the console responds with a message similar to the following.

Use sui client execute-signed-tx to execute the multisig transaction. Set a shell variable equal to the multisigSerialized value you receive from the previous response, then use it to build the execute-signed-tx command.

If successful, the console responds with transaction details.

Transaction response

This response contains the actual values from a test signing. Your response should be formatted the same but the values you receive are going to be different.

Prerequisites

This topic assumes you are somewhat familiar with the Sui CLI, specifically the sui client and sui keytool commands. Consequently, a command might be introduced without any context. If you are unsure about the details of a referenced command, see the Sui CLI documentation for more information.

You need an existing address on the network you are working on to receive an object. The topic assumes that this address is the current active address (sui client active-address), but any address you have access to is fine.

The topic also assumes that your active environment is Testnet (sui client active-env). You can perform these steps on Devnet or a local network as well, but you must adjust the instructions appropriately.

To demonstrate multisig, this topic guides you through setting up and executing a multisig transaction using the Sui CLI.

To begin, create three addresses that will act as the signers for the transaction you perform later in the instruction. Use the sui client new-address command to generate a Sui address and public key for three supported key schemes.

The console displays a response to each successful call that is similar to the following:

When working with blockchain data, addresses and hashed values create large strings that can be cumbersome to work with in a CLI environment. To make referencing values easier in subsequent commands (and facilitate copy and paste), this topic uses shell variables. Use the values you receive from the console responses to set shell variables for these addresses, replacing the variables with the appropriate address.

Create one more shell variable assigned to your active address.

You can set the shell variables to the alias values instead of addresses if you want.

Use sui keytool to list the addresses you created in the previous section.

The response resembles the following, but displays actual alias names, addresses, keys, and peer IDs:

The output includes public key data that you use later, so create shell variables to store the information. Don't forget to replace with the actual values you receive from the previous console response.

To sign a transaction using multisig, you need to create a multisig address using sui keytool multi-sig-address . The multisig address is created using the public keys from each individual participating address. Each address is also assigned a weight value that determines how many are needed to create a valid signature. When summed, the weight of the included signatures must be greater than or equal to the threshold value you also set with the command. For this example, use the following command, which states that the first two addresses require at least one more signature to create a valid multisig. The last address has a weight of 3 , which is equal to the threshold value, so its owner can create a valid signature without the others.

The response resembles the following:

Before getting SUI, set a MULTISIG shell variable to the multisig address provided at the top of the previous response (substituting the actual address for).

If you use the sui client objects $MULTISIG command, you can see that the newly created multisig address has no objects. This means you need to get SUI before you can perform any transactions. To get SUI for your multisig account, use the sui client faucet command and provide the multisig address using the --address flag. Run this command twice so that the multisig address owns at least two SUI. This example uses two SUI so that one can be transferred and the other can pay for gas.

Use the sui client gas command to verify the address now has at least two SUI.

It's now time to transfer an object from the multisig address. For simplicity, this example uses one of the coins your multisig address owns as the transfer object. Copy the object ID for one of the address' coins and use it to set a shell variable value.

Use the sui client transfer command to set up the transfer. The --serialize-unsigned-transaction flag outputs the Base64-encoded transaction bytes.

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

The console displays the result ( ), which you can assign to another shell variable.

Use the sui keytool sign command to sign the transaction using two of the addresses you created previously.

You can create the signature with other tools, as well, as long as you serialize it to flag || sig || pk .

Each successful call to the command receives a response similar to the following.

Create two more shell variables to store the signatures, replacing with the values from the previous command responses.

As mentioned, the multisig must be composed of enough individual signatures such that the sum of the participating signer weights is greater than the threshold value. Use the sui keytool multi-sig-combine-partial-sig command to combine the ed25519 signature (weight: 1) and the secp256k1 (weight: 2). To complete the command, you must provide all public keys, their weights, and the threshold that defined the multisig address.

If successful, the console responds with a message similar to the following.

Use sui client execute-signed-tx to execute the multisig transaction. Set a shell variable equal to the multisigSerialized value you receive from the previous response, then use it to build the execute-signed-tx command.

If successful, the console responds with transaction details.

Transaction response

This response contains the actual values from a test signing. Your response should be formatted the same but the values you receive are going to be different.

Executing multisig transactions

To demonstrate multisig, this topic guides you through setting up and executing a multisig transaction using the Sui CLI.

To begin, create three addresses that will act as the signers for the transaction you perform later in the instruction. Use the sui client new-address command to generate a Sui address and public key for three supported key schemes.

The console displays a response to each successful call that is similar to the following:

When working with blockchain data, addresses and hashed values create large strings that can be cumbersome to work with in a CLI environment. To make referencing values easier in subsequent commands (and facilitate copy and paste), this topic uses shell

variables. Use the values you receive from the console responses to set shell variables for these addresses, replacing the variables with the appropriate address.

Create one more shell variable assigned to your active address.

You can set the shell variables to the alias values instead of addresses if you want.

Use sui keytool to list the addresses you created in the previous section.

The response resembles the following, but displays actual alias names, addresses, keys, and peer IDs:

The output includes public key data that you use later, so create shell variables to store the information. Don't forget to replace with the actual values you receive from the previous console response.

To sign a transaction using multisig, you need to create a multisig address using sui keytool multi-sig-address . The multisig address is created using the public keys from each individual participating address. Each address is also assigned a weight value that determines how many are needed to create a valid signature. When summed, the weight of the included signatures must be greater than or equal to the threshold value you also set with the command. For this example, use the following command, which states that the first two addresses require at least one more signature to create a valid multisig. The last address has a weight of 3 , which is equal to the threshold value, so its owner can create a valid signature without the others.

The response resembles the following:

Before getting SUI, set a MULTISIG shell variable to the multisig address provided at the top of the previous response (substituting the actual address for).

If you use the sui client objects $MULTISIG command, you can see that the newly created multisig address has no objects. This means you need to get SUI before you can perform any transactions. To get SUI for your multisig account, use the sui client faucet command and provide the multisig address using the --address flag. Run this command twice so that the multisig address owns at least two SUI. This example uses two SUI so that one can be transferred and the other can pay for gas.

Use the sui client gas command to verify the address now has at least two SUI.

It's now time to transfer an object from the multisig address. For simplicity, this example uses one of the coins your multisig address owns as the transfer object. Copy the object ID for one of the address' coins and use it to set a shell variable value.

Use the sui client transfer command to set up the transfer. The --serialize-unsigned-transaction flag outputs the Base64-encoded transaction bytes.

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

The console displays the result ( ), which you can assign to another shell variable.

Use the sui keytool sign command to sign the transaction using two of the addresses you created previously.

You can create the signature with other tools, as well, as long as you serialize it to flag || sig || pk .

Each successful call to the command receives a response similar to the following.

Create two more shell variables to store the signatures, replacing with the values from the previous command responses.

As mentioned, the multisig must be composed of enough individual signatures such that the sum of the participating signer weights is greater than the threshold value. Use the sui keytool multi-sig-combine-partial-sig command to combine the ed25519 signature (weight: 1) and the secp256k1 (weight: 2). To complete the command, you must provide all public keys, their weights, and the threshold that defined the multisig address.

If successful, the console responds with a message similar to the following.

Use sui client execute-signed-tx to execute the multisig transaction. Set a shell variable equal to the multisigSerialized value you receive from the previous response, then use it to build the execute-signed-tx command.

If successful, the console responds with transaction details.

Transaction response

This response contains the actual values from a test signing. Your response should be formatted the same but the values you receive

are going to be different.

Related links