Secret Contracts

Uploading a Secret Contract

To upload a contract:

```

Copy secretcli tx compute store ./contract.wasm.gz --from mykey --source "https://github.com//tarball/" --builder "enigmampc/secret-contract-optimizer:1.0.2"

```

- --source
- : Optional tarball of the source code, so your contract will be verifiable
- .
- --builder
- : Optional docker image used to compile./contract.wasm.gz
- , so that your contract will be verifiable
- . This is important for reproducible builds so you should figure out the exact version ofenigmampc/secret-contract-optimizer
- that you were using.
-

To get the contract's code ID:

```

Copy secretcliqtx[hash]

```

This will output a long JSON output, like this:

```

Copy { // [...] "logs":[ { "msg_index":0, "log":"", "events":[ { "type":"message", "attributes":[ { "key":"action", "value":"store-code" }, { "key":"module", "value":"compute" }, { "key":"signer", "value":"your secret address" }, { "key":"code_id", "value":"your code id" } ] } ] } ], "gas_wanted":"5000000", "gas_used":"3720108", "tx":{ "@type":"/cosmos.tx.v1beta1.Tx", "body":{ "messages":[ { "@type":"/secret.compute.v1beta1.MsgStoreCode", "sender":"your secret address", "wasm_byte_code":"...base64 encoded string of your contract's bytecode ...", "source":"", "builder":"" } ], "memo":"", "timeout_height":"0", "extension_options":[], "non_critical_extension_options":[] }, }, // [...] "timestamp":"2022-06-23T13:52:35Z" }

```

You will then find the code id under the logs.events array on the object with key code_id.

Instantiating a Secret Contract

In order to instantiate a contract, simply run the following command:

```

Copy secretclitxcomputeinstantiateCODE_ID"INIT_INPUT_MSG"--frommykey--label"UNIQUE_LABEL"

```

WhereCODE_ID is the code id that you got from the command above andINIT_INPUT_MSG is a JSON encoded version of the init message required in your contract. This message will depend on your contract.

To get the contract's address:

```

Copy secretcliqtx[hash]

```

You will find the contract address under logs.events.array on the object with key contract_address.

Executing a Secret Contract

Executing a contract is just as simple, simply use

```

Copy secretcli tx compute execute CONTRACT_ADDRESS "EXEC_INPUT_MSG"

```

WhereCONTRACT_ADDRESS is the address you found above, andEXEC_INPUT_MSG is the message containing the handle function you're trying to execute. This message will heavily depend on your contract, but generally the format follows the following pattern:

```

Copy { "handler_name_as_snake_case":{ "argumentA":"value for argument A", "argumentB":"value for argument B" } }

```

You can also execute a function on a contract by using the contract's label over the address, like so:

```

Copy secretcli tx compute execute --label "UNIQUE_LABEL" "EXEC_INPUT_MSG"

```

Please note that this is not recommended as its easy for someone to deploy a contract with a similar enough label where you could possibly execute the wrong contract by typoing, but it's much harder for the same thing to happen with an address.

Reading the Output of a Secret Contract Tx

In order to read the output of a transaction, such as the output of a handle function called by compute execute or a contract that has just been instantiated you would run the following

```

Copy secretcliqcomputetxHASH

```

Where HASH is your transaction hash.

Querying a Secret Contract

Querying a smart contract is just as easy, you just execute the following:

```

Copy secretcliqcomputequeryCONTRACT_ADDRESS"QUERY_INPUT_MSG"

```

WhereCONTRACT_ADDRESS is the address of your contract andQUERY_INPUT_MSG is the query you're trying to run. The output will depend on your contract.

Last updated1 year ago On this page *Uploading a Secret Contract* Instantiating a Secret Contract* Executing a Secret Contract * Reading the Output of a Secret Contract Tx* Querying a Secret Contract

Was this helpful? Edit on GitHub Export as PDF